Dear Editor,

Thank you for the through and careful comments from the reviewers, and the opportunity to revise and resubmit our article to Future Generation Computer Systems. We have revised and modified the article in the light of these comments.

The comments from the second reviewer were particularly perceptive, and after consideration of these we have made some more substantial changes. It was clear that our approach to explaining the grouping operator by beginning with an example of the pitfalls faced in designing the operator was misleading. As a consequence we have taken a much more direct approach in Section 4. We have also removed our reliance on pre-existing minima and maxima in section on events, and so have re-written section 5. We have also removed the section discussing the extension of our work to agents, acknowledging with reviewer two that substantial extra work would have been required to make this section consistent with the rest of the paper. We believe the paper stands without this extension.

Below we reproduce the reviewer comments. Our responses to them and the changes we have made are inline in blue.

Comments from the editors and reviewers:

Reviewer 1

The authors have addressed the concerns, but have indeed raised some others:

1. The authors use the following words interchangeably: abstraction, obfuscation, and hiding of information. These terms are not similar, and have been used carelessly in the title and the Introduction. For review:

   * Obfuscation is typically applied to programs to make it "harder to understand". Christian Collberg, Clark Thomborson, and Douglas Low. A taxonomy of obfuscating transformations. Technical Report 148, Department of Computer Science, University of Auckland, July 1997.

   * Abstraction is to provide a conceptual view, typically of data structures. "An abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of object and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer."

   * Hiding of information is similar to encapsulation. From OO design:"The process of hiding all the details of an object that do not contribute to its essential characteristics; typically, the structure of an object is hidden, as well as the implementation of its methods. The terms information hiding and encapsulation are usually interchangeable."

   From OO design the correct term in this work to use is encapsulation and not obfuscation or abstraction.

A consequence of this verbiage are the title (Abstracting PROV...) which and the incorrect sentence such as following: "As we will see in the rest of the paper, replacing the three selected nodes with a single abstract node (an activity in this case) while preserving the validity of the document, requires that all other nodes that lie on the directed paths that connect these nodes are also removed." The nodes are technically not removed but hidden or encapsulated.

It is strongly recommended that authors review the draft from the perspective of these definitions.

We argue here that abstraction is the usual word in the PROV context for this work. In Zoom [1] an abstraction mechanism is a mechanism that presents the most relevant provenance information to a user. It is used in [5], in which the author discusses a variety of means of abstracting provenance graphs, including [4], in which the term carries the same meaning as in this work. Also discussed are the Abstract Provenance Graphs from [6]. In these, abstraction is performed using static analysis of a configured workflow before the provenance is created, and a homomorphism between the abstract and the concrete graph is demonstrated. Although they create their abstraction in a different way, the same effect, of removing unnecessary information, is achieved. Finally, [5] discusses [3], in which the authors show how the user views that they present allow provenance to be reasoned about at different levels of abstraction. This understanding of abstraction, and of the ability to move between different levels of abstraction, is key to our understanding of the term here.

We have therefore made no change to the term abstraction, but we have replaced the word obfuscation when it occurred.

2. The introduction does not mention anything about the trust model between PA and IP on obtained provenance. Specifically if PA does not trust the obtained report, what guarantee does it have for trusting the obtained provenance. The authors must provide explanation as to why they think provided provenance will be trusted by PA.

   We have added a discussion in the first paragraph in section 1.1 to make the trust model more explicit.

3. Related Work

   1. There are many differences that this work has with Zoom, but to assert that in Zoom the user requires knowledge about the underlying graph topology is incorrect. In Zoom user knows the relevant modules (which are resented as nodes) and chooses amongst the internal nodes to displayit does not substitute the entire subgraph with another abstract node. A view is typically to improve readability of large graphsand that is the notion used in Zoom, but is different than an encapsulation which is specifically for hiding information. Thus, Zoom does not prevent a

reader to expand the view and see the original graph, but the system being proposed in the article will never divulge underlying information.

2. There is confusion when the actions are user-defined and when dictated by the underlying topology. In comparing Zoom the authors state In Zoom, the user is aware of underlying topology and and in their case the user need not be. In comparing [19], they say the aggregation operator is entirely driven by graph topology and not by a user choice. The latter sentence by inference implies that their approach is driven by user choice, and since any form of aggregation/grouping requires knowledge of graph topology, by consequence, the user is aware of the topology. This conflicts with their statement contrasting Zoom.

The above sentence must be made more specific.

We have clarified the relation of this work to Zoom in the relatedWork section.

4. The paper introduces three operators: path closure, extend, and replace, and provides either a extension and replace or just closure and replace. It is not clear when the IP or the system will chose one operator over the other. The time complexity appears to be the same. So why will extend and replace ever be applied over path closure and replace?

   The key contribution of the paper is the grouping operator. The definition uses the three auxiliary operators path closure, extend and replace. We have added text to better clarify this in the first paragraph of sec 1.2 and in the introduction text of section 4.

5. There are several issues with Fig 6.

   Fig 6. was an example of the problems that a naive approach would face. It is clear to us now that the approach itself was problematic for more than one reviewer, and so section 4 has been substantially revised to remove the misleading early emphasis on wrong solutions, and move more quickly to the solution that we present (the grouping operator). This, we trust, deals with the fundamental issue behind many of the reviewer's points below.

   (a) The basis for the position of e in Fig 6 is not clear. Why in between a1, a3 and a2, a4? Why not, for instance, as a substitute for e1? If this choice of representation is one chosen by authors, then it should be mentioned.

   We have dealt with this by the removal of Fig 6.

   (b) In group, the authors use the term rewired but do not explain what that means. If rewired means ALL incoming and outgoing edges are connected back again, then what happened to edge a1-¿e1 and a3-¿e3? The explanations are confusing: it seems the direction of the edge is introduced but this edge is also relabeled wgBy. Why?

   "Rewired" is a poor choice of term and has been removed. The labels for edges in each of the diagrams in the paper are now clear.

It might be helpful in that Figure to show how each edge in the left hand side of the diagram is mapped to each edge in the right hand side of the diagram.

Dealt with by the removal of Fig 6.

The definition of Group operator is a node grouping operator; the definition does not include edges. But group operator is stated as an alternative to path closure, replace, extend.

'Group' returns a PROV graph, and is a rewriting operator. We have clarified this at the start of section 1.2. Rather than being an alternative to path closure, replace and extend, group is defined using these operators. The rewiring is part of the definition of group, defined in the subordinate operator replace.

Why are edges a1-¿e1 and a-¿e3 not left unlabeled because used edge label will lead to a false dependency. Again the group operator definition does not imply a false dependency may result as a result of it.

The group operator as defined later in section 4 will not produce a false dependency here.

(c) There is something wrong with the mathematical definition: $PG_{gu/ea}\ X\ P(V) \rightarrow PG_{gu/ea}$

The result is the same graph? As a side, the equations are not numbered, and P(V) in this equation is not defined.

We've clarified that this equation is the type signature of the group operator, rather than it's definition.

The powerset operator has been named when it is used, and the equations in the paper have been numbered throughout.

(d) The definition of Group operator is a node grouping operator; the definition does not include edges.

The definition of Group is in terms of extend, closure and replace, and the edges are dealt with by the replace operator. We have clarified this at the start of section 1.2. Group is thus a rewriting operator and the definition includes edges.

(e) Explain why these cycles cannot be broken by simply keeping timing information or happens before relationship.

We deal with the cycles by including nodes that induce cycles in the set of nodes to be removed. We discuss the implications of timing in section 5.

6. refers to issue. Which one? Previous para states two issues.

This is referring to the paragraph before section 4.1 begins. Both problems are caused by the nodes a_1 and a_3. We have reworked the text in Section 4 to introduce the issues (the introduction of cycles and badly typed relations) separately, and to deal with them in turn.

7. it is not clear why the type of v_d and v_s are necessary in definition when G is an element of $PG_{gu/ea}$.

   t is the type of the nodes on the boundary that Extend creates. We have added text just above the definition of Extend to clarify this.

8. Figure 8 uses a new keyword absorb. Why isnt it extend now?

   The figure has been corrected to remove the misleading word.

9. How is Definition 6 related to the grouping operator? Is it related? The significance of this definition is not clear.

   Defn 6 is the definition of the group operator under the assumption that all the nodes removed are of the same type (homogeneous grouping). We make this clear at the start of section 4.1, and again just before definition 6. Section 4.2 generalises to full grouping.

10. In Section 4.2 the word propagation is used. Is it similar to extension?

    We have avoided the word propagation and clarified that strict *e_grouping* involves a further step.

Reviewer 2

the reviewer kindly summarised the detailed comments here towards the end ("Overall...") so responses are primarily given there.

Summary

This paper is about abstracting provenance graphs, using the PROV model promulgated by the W3C. PROV graphs include nodes that can be entities, activities or agents, and edge relationships indicating usage, generation, invalidation, association, attribution, delegation, and other relationships between them. Given such a graph, describing a history or trace of a process, a common concern is to hide information according to some policy. Provenance abstraction is a subproblem arising when we have identified a set of nodes we wish to abstract (i.e. collapse into a single node). This paper considers the problem: if the input graph is a "valid" one according to the PROV standard (specifically, the validity constraints in the PROV-CONSTRAINTS standard), what other nodes/edges need to be collapsed, or other changes to the graph made, so that the result is still valid? There are several difficulties: collapsing nodes of different types into a single node (of a given type) may result in edge relationships needing to be adjusted, additional nodes may also need to be collapsed in order to avoid cycles, etc.

The paper first considers the subcase where there are only two node types (entity/activity) and two edge relationships (usage/generation). Thus, the graph is bipartite. Formal definitions of the problem and proposed algorithm are given: first the set of nodes is closed-off to include all nodes "between" selected nodes, then additional nodes on the frontier of the selected set are added

so that the outside of the selected set matches the intended collapsed node type. The paper considers the constraints and argues that abstracting a valid graph results in another valid graph. Next the paper shows how to extend the results to consider PROV graphs with agents, with an unstated assumption that agent nodes are disjoint from entity and activity nodes, so that the agent nodes only interact with the rest of the graph in fairly limited ways and the underlying bipartite abstraction approach still works.

One complication in this setting is that edges from agents to abstracted nodes may no longer make sense, and in this case they are removed.

Evaluation

The problem being considered seems challenging and I think a good solution would be worthy of publication. However, I'm not yet persuaded that the approach presented in the paper is sufficiently motivated and justified.

1. The problem seems under-specified. The abstract states two goals for abstraction: first, the abstract version of the document should be valid if the original version is, and second, the dependencies (edges) in the abstract version should be justified by those in the original. (Section 1.2 and the conclusion likewise asserts that abstraction will not introduce false information, but I don't see where this is actually proved.) However, the second goal is not captured formally, and I think there is a third important goal that is not mentioned at all: that the abstracted document should not "abstract away too much".

   For example, one (unserious) solution could be to abstract the whole graph to a single node, discarding all edges. This would be correct (the internal details of the abstracted subgraph are definitely hidden, all right) but is not "optimal" in some sense among the correct solutions. Another slightly less silly solution could be to discard the entire abstracted set (and all incident edges), resulting in a PROV graph that is still valid, hides the intended information, but still abstracts "too much" in some sense since there is nothing represented the abstracted nodes. What is missing to rule out both silly solutions] is some condition that says you can't throw "too much" of the PROV graph away, for example, saying that parts of the graph that have nothing to do with (i.e. not neighbors of) an abstracted node should not be affected.

   We have made explicit our the goals of the group operator (validity of the abstracted document and that all relationships are justified), and introduced minimality as a guiding principle of the group operator, as the reviewer suggests.

   In addition, I think the notion of correctness of abstractions in this paper needs to be compared with the notion of "sound" workflow views explored for ZOOM by Liu et al. [A].

   See our answer to point 5 below.

2. I did not really follow the discussion of preservation of constraints in section 5. I consider myself fairly familiar with the constraints specification. The specification (and the accompanying semantics) does not assume the events are linearly ordered, and the authors correctly point this out. However, the argument that the constraints are satisfied by the abstracted document seems to hinge on being able to compute maxima and minima among (finite) sets of events. This may not be possible in a general preorder, so using the notation for "max" and "min" is not necessarily meaningful, since the max or min of a set may not exist. As a simple example, suppose we have two parallel chains e1 wgb a1 used e2 and e3 wgb a2 used e4. Then the events related to e1/a1/e2 are unrelated to those for e3/a2/e4, for example the max and min of the two concurrent usage events doesn't exist.

   See answer to point 2 below.

   Another (more minor but relevant) nitpick is that the ordering in a PROV graph is a preorder (aka quasiorder), so that there can be nontrivial cycles in the $\leq$ relation (for example for "simultaneous" generation events). This means that if an upper bound or lower bound exists for a set of events, it may not be unique. This also seems likely to lead to strange situations, for example if abstraction leads to two generation events becoming simultaneous, it will seem strange if they have different timestamps (though there is nothing wrong with this vis a vis validity since the timestamps are just uninterpreted data as far as PROV validity is concerned.) Yet in several places the paper assumes that simultaneous events are necessarily equal, which is not the case in PROV.

   I think this difficulty can be overcome by, for example, assuming that you are given an event preorder consistent with the original PROV instance and constructing from that another preorder consistent with the abstracted instance in which the needed maximal or minimal events exist, adding them if necessary, and choosing the maximum or minimum of a set of events arbitrarily if there is more than one possibility.

   This is dealt with in our answer to point 2 below.

3. It is fine and reasonable that the paper mostly focuses on the entity/activity/generation/use subset of PROV. The later section on extending to include agents, however, implicitly relies on some simplifying assumptions. In PROV, entity and activity are disjoint classes, but agent is not assumed to be disjoint from either entity or activity. So, a node could be both entity and agent, or both entity and activity (just not entity and activity or all three). I think this possibility would lead to some complications compared to the simple case discussed in the paper, and this simplifying assumption (that agents, entities and activities are disjoint) should be made explicit.

   Another related point is that by assuming that nodes/edges have exactly one PROV type, the paper may miss an opportunity for using flexibility that already exists in PROV to aid abstraction. For example, nothing in

PROV says that a node has to have any type at all; and though edges are expected to have a property type, PROV provides a generic edge type "wasInfluencedBy" that superclasses the other edge types. Thus, the problem with edges becoming isolated discussed in section 6.3 could be avoided by using wasInfluencedBy edges to represent the problematic edge relationships, instead of deleting such edges and removing the resulting disconnected nodes. Alternatively, the issue highlighted in figure 15 could be avoided by instead allowing eN to be both entity and agent, so that the actedOnBehalfOf edge still makes sense. (However, I'm not sure this works in general and would result in graphs that no longer have agents distinct from entity /activity nodes.)

We have explicitly restricted our scope throughout to consider only bipartite graphs, and removed the section on agents.

4. A relatively minor point is that there is some further uncited work on provenance sanitization/abstraction/redaction [A,B,C,D]. The notion of convexity mentioned here appears also in [B] for example, which only considers activity abstraction for a simplified form of OPM. There is also a short survey [E] which already discusses/compares work up to circa 2014.

   Please note that only references A and B are resolved below. C,D are not clear, while we believe E to be [2] which we now have considered.

Overall, I think the paper presents work that is worth publishing: I started out thinking the problem and solution seemed a bit simplistic, but the more I thought about (and read) the paper the more I realized that there are some nontrivial subtleties. However, the above issues need to be addressed. Given that I've been asked to review this after one round of reviewing and revision has already been performed, I hesitate to require major revision, so instead I'll suggest the following changes that could be made in a minor revision round:

1. Clarify what part of the paper proves that that relations in the abstracted graph are "justified", and discuss the issue of how to avoid "abstracting too much".

   In Definition 10 we have clarified what we mean by "justified" relations. Essentially, nodes are justified if they are unchanged or represent a set of nodes where the boundary nodes are of the same type. Relations are justified if the nodes are justified and the type is unchanged.

   We address the issue of abstracting too much in Section 4.5. Essentially, our approach in the paper is to ensure the PROV-compliance of the result of the group operator. In section 4.5 we outline an approach to the issue of abstracting too much. We propose subdividing the nodes selected and applying the group operator to each subdivision in turn. Identifying the "right" subdivision would take substantial care and evaluation of alternatives, and would detract from the focus of the paper.

2. Clarify the discussion of max/min and avoid apparent reliance on the maxima/minima already existing in the event structure for the original document.

   We acknowledge that the objection raised by the reviewer regarding the difficulty with max{} and min{} in common cases such as those in Fig 1 in this document is substantial. While it could have been circumvented as the reviewer themselves suggested, we found a simpler and more natural formalisation. We have entirely rewritten Section 5 as a consequence. Also, the lengthy Appendix is no longer needed and has been removed. In essence, in the new formalisation we propose that abstract events be defined in terms of 'ground' events in the original graph, through a function that maps abstract events to *sets* of ground events. Also, a preorder is defined on these events so that PROV-constraint validity can be established. We further introduce a new order relationship on sets of ground events, defined in terms of the original PROV-constraints preorder, and suggest that the mapping should be order-preserving. Finally, we indicate and justify a specific definition for the mapping function.

3. Clarify assumptions needed for the existing algorithm to work in the presence of agents.

   In view of the reviewers' comments we acknowledge that the agents formalisation section requires substantial additional work. In the interest of time and indeed to ensure proper focus on the paper, and considering that the abstraction model including events still stands without mentioning agents, we decided to remove the section altogether.

4. Discuss alternative abstraction strategies such as using wasInfluencedBy edges.

   We have now included an example of how the influence relationship may be used for abstraction and a brief discussion in Sec. 4.5, along with a new figure

5. Cite/discuss uncited related work and compare with other formal notions of abstraction correctness such as "sound workflow views"

   A reference to [A] along with a discussion as suggested here has been added to Sec.2.1 (Related work). Also [B] is now discussed quite extensively in the same section. E is cited with brief comments. As mentioned above, B and C which the reviewer refers to are undefined in the review.

These changes would help to circumscribe the contribution of the paper more clearly, and identify possible areas for further work building on it.

Of course, I would also be happy with more substantial changes that address the concerns above more thoroughly.

The conclusion has also been reworked to take into account the changes in the rest of the paper.

Detailed comments

The paper is overall in fairly good shape, there were few typos or difficult-to-follow points.

p10: "Further extensions... wasDerivedFrom and wasInformedBy" - while wasInformedBy in Prov is just an abbreviation for a 2-step generation/use path, wasDerivedFrom is a little more complicated (it can correspond to a chain of use/generation steps) and has a special status in PROV-CONSTRAINTS: it is the only relationship where event ordering cycles are strictly forbidden. Incidentally, this means that in the absence of wasDerivedFrom, it's easy in some sense for a document to be valid, as long as it is well-formed, all of the events can be simultaneous.

We have re-worded text in sec 3.1 to say that those two relations can be removed using inference 5 and 11 from PROV-CONSTRAINTS

p12. I think it would help discussion to clarify that there might be many preorders $(Ev, \preceq)$ that are consistent with a given, valid PROV document, and throughout this and the next section you are fixing one such preorder. I think it would also be helpful (if you do as I suggest above) to distinguish between the preorder $(Ev, \preceq)$ that is given for the original document, and a new one that you construct $(Ev', \preceq')$, that extends $(Ev, \preceq)$ with any new events needed to describe the new edge relationships introduced for the abstracted node. Likewise, it would be helpful to state the constraint-preservation result somewhere explicitly as a theorem so that it is clear what formal property corresponds to the informal description given earlier.

We have rephrased sec 3.3 to clarify

p12. In the formula $\preceq \subset Ex \times Ev$, there should be space between the preorder and subset symbols

Done

p12. Missing colon/punctuation after "in [9])"

Done with full stop.

p13, In C7, I think the RHS should be end(a), not ev(a).

Done

p13, in C8, you say that two invalidation events for the same activity are equal, but (as for C2)they need not be equal, just simultaneous.

Done

p15 "subgraph given by pclos" - since pclos returns a set of nodes, not a graph, did you mean to say "subgraph induced by pclos"?

Done

p16. The figures fall off the edge of the page.

Done

p17. In definition 3 and following discussion, I had to think for a minute to convince myself that extend correctly ensures that boundary nodes are of type t. I think this happens to be the case for bipartite $PG_{gu/ea}$ graphs. The definition ensures that all added nodes are of type t, but if more edges or node

types are present then it seems possible that this definition might leave some boundary nodes of other types. Please say a little more here since this fact is relied on later and then when agents are considered I think the correctness of extend when t = e or a needs to be revisited.

We have chosen to drop the (less mature) work on agents

p21. In the "if $|V_{gen}| \leq 1$" side condition, there is unnecessary spacing between "—" and "V" and missing spacing between "—" and "≤".

The spacing has been corrected.

p21. "v2 is reachable from v2" - I think one of these was meant to be v1.

Corrected.

p22. "And secondly, in a provenance graph ... the graph is bipartite" - true for $PG_{gu/eq}$ but not generally. Also, "with respect of" -¿ "with respect to"

Corrected.

p22. "the theoretical max" -¿ "... maximum"

Corrected.

p22. "will have to hidden" - missing "be"

Corrected.

p23. In section 5 it is implied that we can reason about ordering constraints (i.e. a preorder) by reasoning about all the possible linear orderings consistent with the preorder, and perhaps this is the rationale for using min and max. However, I am not sure that introducing discussion of linear orderings is necessary or helpful here; I think it would be clearer to present the content of this section as a recipe for constructing an event preorder for the abstracted graph G' from any such preorder consistent with the original graph G, adding any necessary min/max events.

We have taken this approach.

p24. In figure 10, I was uncertain whether your model allows multiple edges of the same kind between two nodes. The formal definition of the model doesn't appear to allow this; PROV does, but does so by assigning identifiers to use/generation/etc relations, which are not modeled explicitly here. (but then on page 24 you say you are going to use such identifiers for convenience anyway).

We have now made clear in the revised section 5 that using identifiers is required at this point since the abstraction mechanism can merge relations and we require a preorder that is consistent with all original relations.

p24. "wgby" was elsewhere called genBy as well as wgBy (in figures)

Fixed

p27 "Let V* to" - remove "to"

Fixed

p27. "abstracted prov graph" -¿ "...PROV..."

Fixed

p27. "following equalities, define" remove comma

Fixed

p28. Most of the definitions in this section have "e-grouping" in parentheses except definition 11. This seems inconsistent (though I think you could get rid

of all of them). Also, why isn't definition 11 exactly like definition 9 up to symmetry?

Fixed

p30. In definition 15, the generation events are simultaneous but not necessarily equal.

Fixed

p30. "the event of the invalidation event" - there seem to be extra words

Fixed

p34: "agent nodes are always the targets of directed edges" - this relies on the unstated assumption that agent nodes are disjoint from entity/activity nodes. In general, an agent node that is also an entity or activity could be the source of a directed edge (use, generation etc.)

Fixed

p34. "or its symmetric," -¿ "or its symmetric form,"

Fixed

p35. There are occurrences of both "obo" and "abo" as abbreviations for "actedOnBehalfOf", please make consistent. Fixed

p36. Removing edges and removing isolated nodes seems undesirably lossy to me. For example if there was already an isolated agent node in the graph you would discard it even though it was not disconnected as a result of abstraction. Edges that are no longer well-typed could be replaced with wasInfluencedBy edges which can link any entity/activity/agent nodes. This would avoid the disconnection problem. Anyway, if you solve the disconnection problem by discarding isolated nodes, you might still wind up with a less connected graph (e.g. what if ag4 in figure 15 was also acting on behalf of some ag5). These won't be discarded because they are not isolated.

Fixed by restriction our attention to Entities and Activities.

p37. "will not introduce false information" - as mentioned above, I'm not sure this claim has been supported in the paper; in any case, I think it should be clearer that the claim means "no new edges/nodes are added", but some edges/nodes present in the graph could be deleted, and this might create false impressions that two things are not connected even though they were connected in the original graph. For example, suppose ag42 has agents acting on behalf of it in two subgraphs. I abstract subgraph 1 into activity 1 and subgraph 2 into activity 2, so the actedOnBehalfOf edges no longer typecheck and go away. I also discard ag42 since it is now isolated. The resulting graph is now two disconnected abstract nodes, and the fact that there was a single agent involved somehow in both of them is no longer present in the graph. If that was information I wanted to know, then I might indeed think the abstracted graph is misleading unless it is clear that abstraction can discard this kind of information. (That in turn is a reason, to me at least, to consider retaining such edges as influences if no more specific relationship is appropriate after abstraction).

We have clarified here using the concept of "justified" relations, and restricted our attention to bipartite graphs to avoid this problem.

p37. "PROV-CONSTR" -¿ "PROV-CONSTRAINTS"?

Fixed.

p38. Citation 9: include URL

Done.

p41. The discussion at the beginning of the section seems to conflate times and events (e.g. start(a) is referred to as a start time).

This section has been removed

p41. In C2, the goal is to show that generations are equal, not just simultaneous, but this is stronger than needed (which is fine). This happens again in discussion of C2 for a-grouping on page 44.

Removed

p43-46. There are several chains of equational/relational reasoning that are not typeset very readably. I suggest reformatting as:

expression 1
= reason 1
expression 2
≤
...
= reason n-1
expression n

The section has been removed

References [A] Ziyang Liu, Susan B. Davidson, and Yi Chen. 2011. Generating sound workflow views for correct provenance analysis. ACM Trans. Database Syst. 36, 1, Article 6 (March 2011), 35 pages. DOI: https://doi.org/10.1145/1929934.1929940

[B] Barbara T. Blaustein, Adriane Chapman, Len Seligman, M. David Allen, Arnon Rosenthal: Surrogate Parenthood: Protected and Informative Graphs. PVLDB 4(8): 518-527 (2011)

References

[1] 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.

[2] James Cheney and Roly Perera. An analytical survey of provenance sanitization. In Bertram Ludäscher and Beth Plale, editors, *Provenance and Annotation of Data and Processes*, pages 113–126, Cham, 2015. Springer International Publishing.

[3] Sarah Cohen-Boulakia, Olivier Biton, Shirley Cohen, and Susan Davidson. Addressing the provenance challenge using zoom. *Concurrency and Computation: Practice and Experience*, 20(5):497–506, 2008.

[4] Paolo Missier, Jeremy Bryans, Carl Gamble, Vasa Curcin, and Roxana Dánger Mercaderes. Provabs: model, policy, and tooling for abstracting PROV graphs. In *IPAW 2014: Provenance and Annotation of Data and Processes*, volume 8628 of *Lecture Notes in Computer Science*. Springer, 2014.

[5] Luc Moreau. Aggregation by provenance types: A technique for summarising provenance graphs, 2015.

[6] Daniel Zinn and Bertram Ludäscher. Abstract provenance graphs: Anticipating and exploiting schema-level data provenance. In Deborah L. McGuinness, James R. Michaelis, and Luc Moreau, editors, *Provenance and Annotation of Data and Processes*, pages 206–215, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.