The Xeros Data Model: Tracking Interpretations of Archaeological Finds

Michael O. Jewell[1], Enrico Costanza[1], Tom Frankland[2], Graeme Earl[2], and Luc Moreau[1]

[1] School of Electronics & Computer Science, University of Southampton, Southampton, United Kingdom, SO17 1BJ
[2] Faculty of Humanities, University of Southampton, Southampton, United Kingdom, SO17 1BF

Abstract. At an archaeological dig, interpretations are built around discovered artifacts based on measurements and informed intuition. These interpretations are semi-structured and organic, yet existing tools do not capture their creation or evolution. Patina of Notes (PoN) is an application designed to tackle this, and is underpinned by the Xeros data model. Xeros is a graph structure and a set of operations that can deal with the addition, edition, and removal of interpretations. This data model is a specialisation of the W3C PROV provenance data model, tracking the evolution of interpretations. The model is presented, with operations defined formally, and characteristics of the representation that are beneficial to implementations are discussed.

1 Introduction

Archaeological practice is focused on the aggregation and interpretation of knowledge. The process begins with the excavation of multiple regions within a trench, known as 'contexts'. The finds discovered during excavation are tagged with an ID and placed into find bags, thus grouping them according to the context in which they were found. These 'Find IDs' are usually unique to sites, with larger sites sometimes prefixing a non-unique ID with an area code to ensure uniqueness. Archaeologists also use symbols for different purposes: contexts are circled, smaller finds are in triangles, soil samples are in diamonds, etc.

The find bags and their contents then become the subject of interpretation by specialists of different areas of archaeology. For example, skeletal material may be examined by an osteoarchaeologist, while an environmental archaeologist may glean information from charcoal and plant remains found at the site. Measurements by each specialist are recorded and associated with the individual finds by use of a recording sheet, which is associated with the unique ID on the find bag. Finally, an expert will examine the aggregated data for the site, and produce a report based on an interpretation of the data.

During the whole process, individual archaeologists also produce personal interpretations of their work. These may be in the form of handwritten notes,

P. Groth and J. Frew (Eds.): IPAW 2012, LNCS 7525, pp. 139–151, 2012.

diary entries, photographs, or multimedia recordings. At present, there is a divide between this 'unstructured data' and the 'structured data' recorded by the specialists on recording sheets. While both types of data may inform the excavation process, the unstructured data is not usually included in the dissemination of the findings from the site, whereas the structured data is recorded directly and preserved for analysis. At commercial sites, structured data is of a higher priority, but the interpretation is still dependent on the prior experience of the archaeologist.

Structured and unstructured data may influence the recording of structured data: if, for example, an excavator posits or determines via measurement that a shard may be part of a larger item, they are likely to take this into consideration when analysing further shards found in the same context. The excavation approach is also strongly influenced by preliminary examination techniques, including geophysics, survey methods, field walking, and the digging of trial pits.

Given the amount of recording that takes place at a dig, it is valuable to preserve both the structured and unstructured data as interpretations of finds. This is of use both to students, who could explore how conclusions were reached, and to other archaeologists, who may reach hypotheses that were not previously considered. By opening this data up to all of the archaeologists at a dig site, multiple viewpoints can be created and knowledge accumulated.

Some technologies already exist with the ability to capture finds and notes, but these have shortcomings. Existing 'find databases' (such as ARK[5] and IADB[9]) let archaeologists record the structured data mentioned earlier, but do not allow for the creation of interpretations of this data and do not visualise how knowledge has been built up or altered over time. Wikis provide for the creation of the unstructured data, and preserve edits, but do not model the fact that a note may be expanding on knowledge from another note. Users would have to write the structure into the wiki pages using wiki markup, but this is not readily exploitable as it is not explicitly designed into the software. There are also parallels to version control systems, such as SVN and CVS, but these operate on a per-directory level: several files in a folder that is then committed to a repository would be seen as having been created at the same time as the directory.

PoN (Patina of Notes), a web application that allows for the creation and organisation of structured and unstructured data about archaeological finds, was designed to address the above issues. Finds are extended with notes in a manner akin to attaching Post-it notes to an item. This results in a new 'state of knowledge': the original find has had extra information added, and the overall information in the system has grown. Notes can be stacked, thereby extending prior states of knowledge; alternative stacks can be created; and notes can be placed bridging multiple entities. By preserving the state of the system as notes are added, it is possible to see how knowledge has accrued over time, how structures have grown, and how and why edits have taken place.

This paper describes the Xeros data model that underpins PoN. The model consists of a graph structure that extends the PROV[8] Provenance Data Model;

and a set of operations defined to act upon it. This research is detailed in three contributions:

1. A formal specification of the fundamental operations: extension, edition, and reduction. Extension allows for the 'stacking' of knowledge onto an existing state of knowledge, or onto a find itself. Edition models a change of content between two versions of a note or a find. Reduction removes a state from the data model, but ensures that entities are still preserved when exploring prior system states.
2. Several properties are required for the PoN system, including the ability to record asynchronously and the avoidance of locks (to allow a single entity to be edited by more than one party). These properties are effected in the data model by commutativity operations (cross-entity completion, post-fact merging, and pre-fact recall) and idempotence rules.
3. The commutativity operations incur some storage and computation costs. These are in part unavoidable, to ensure the integrity of the model, but some may be avoided in order to ease efficiency. An approach to optimisation is disussed.

2 Related Work

As mentioned previously, some popular archaeological systems already exist. IADB[9], the Integrated Archaeological Database, manages data throughout the lifespan of excavation projects, including recording, analysis, and dissemination. Unique URIs are provided for Finds, Contexts, etc, and these are stored in the system's database. ARK[5] provides similar facilities for the collection of archaeological data, but allows for more flexible interface control. As the PoN implementation of Xeros uses URIs, it can augment the IADB software very easily, while leaving the more specialised data entry to this purpose-built software.

There are also some approaches to add meaning to wikis: Semantic MediaWiki[6] lets users embed triples into wiki pages, which is especially useful with templated pages (e.g. a Country template may contain a hasCapital predicate). Alternatively, DBWiki[1] combines the schemas present in existing databases with wiki functionality, providing versioning on the data entry process. It is possible to query for information on a country, and then retrieve the history and provenance of that data. Neither of these approaches address the issue of interpretation building, however: to create a new note in these systems would require the user to explicitly add links to the states of knowledge to which they were referring.

Some ontologies already address areas of this research: CIDOC CRM[3], an ontology for concepts and relationships used in cultural heritage documentation, has been extended to capture the modeling and query requirements regarding the provenance of digital objects[10]; and the Annotation Ontology[2] is a vocabulary for annotating electronic documents with various forms of annotations. Xeros does not intend to replace these approaches: CIDOC artifacts could be treated as entities, and Annotation Ontology annotations could be used to extend them.

Existing provenance models, such as OPM[7] and PROV, offer a very generic model of provenance. While these are powerful due to their versatility, the models must be specialised if they are to fit the archaeology representation.

Finally, there are also existing annotation systems for other domains: The Distributed Annotation System (DAS)[4] allows for the exchange of biological sequence annotations, and many bioinformatic applications and websites support the DAS communication protocol.

3 Xeros: Representation and Operations

Xeros allows for the building of interpretations, the edition of entities, and for the non-destructive removal of states of knowledge using a reduction process. Users can navigate through the evolution of this knowledge, using completion operations to suggest interpretations that have not been explicitly created. The non-destructive nature of the operations means that the processes that have led to a state of knowledge can always be seen: hence the extension of the PROV data model.

The data model is defined as a graph structure, consisting of a set of entities (V) connected by edges (E). All entities have both a positional co-ordinate $\bar{c}$ with components (x, e, r) and an index i, allowing them to be uniquely identified in the model. The positional co-ordinate places the entity in a 3 dimensional space, with x, e, and r respectively corresponding to the eXtension, Edition, and Reduction operations that the entity has undergone: hence the name 'Xeros'. The displacement vectors for the three operations are shown in Figure 1.

$$\bar{x} = (1, 0, 0)$$
$$\bar{e} = (0, 1, 0)$$
$$\bar{r} = (0, 0, 1)$$

Fig. 1. Displacement vectors for extension, edition, and reduction.

$$new(V, \bar{c}_n) = m : \forall p, 0 \leq p < m,$$
$$V(\bar{c}_n, p) \neq \perp$$
$$V(\bar{c}_n, m) = \perp$$

Fig. 2. $new(V, \bar{c}_n)$. Produces a valid index i for a co-ordinate that does not conflict with an existing co-ordinate.

The index i is required when multiple entities occupy the same positional co-ordinate. For example, if an entity at position $\bar{a}$ is edited twice, the two resultant entities will have the same position $\bar{a} + \bar{e}$. As a result, the index i is incremented: the first entity would be at $(\bar{a} + \bar{e}, 0)$ and the second at $(\bar{a} + \bar{e}, 1)$. A formalization of this, given a set of vertices V, is provided in Figure 2.

Three Xeros-specific edges may be created by operations on the data model: $isX(\bar{a})$, $isE(\bar{a})$, and $isR(\bar{a})$. These correspond to the three main operations that can be performed on entities within the data model: extension, edition, and reduction ($\bar{a}$ being a displacement vector). These edges are subproperties of wasDerivedFrom in the PROV data model. An s edge is also used, which

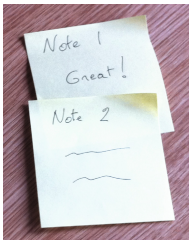
indicates that there is some other relationship between one entity and another (e.g. 'hasNote' between a find entity and a note entity). s must not be one of the Xeros edges, and it follows that for all s, $source(s) \neq dest(s)$. The following operations focus on the Xeros-specific edges, rather than the s relationship.

3.1 Extension

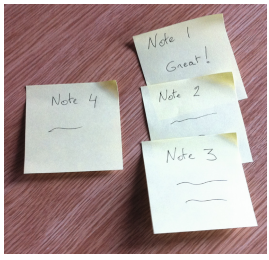
Extension, denoted by $\xrightarrow{isX}$, suggests an addition of information to the system: the accumulation of knowledge. Figure 3 shows the building up of knowledge via extension: an entity (in this case, a surface) is extended with a note; this state of knowledge is then extended with a further note; and later an alternative is added via an extension to the original entity.

Given an initial entity (e_0) and the entity by which it should be extended (e_1), the operation creates an extension entity e_n that represents the state of knowledge in which e_0 has relationship s with e_1. An $isX(\bar{x})$ edge is added from e_n to e_0, indicating that e_n is an extension of e_0 with displacement vector $\bar{x}$, and the s edge is added from e_n to e_1. The case described above would be achieved by extending the find entity with a note; extending the resultant e_n with another note; then later extending the find entity with a note.

(a) An entity is extended with Note 1.



(b) The state of knowledge is extended with Note 2.



(c) An alternate interpretation is added.

Fig. 3. The evolution of a state of knowledge via extensions

3.2 Edition

Edition, denoted by $\xrightarrow{isE}$, indicates that there has been an alteration of an entity's content. In an archaeological context this could be a correction to a find's weight, or an alteration to a note's content. The *edit* operation (Figure 5) creates an edition entity e_n that has an $isE(\bar{e})$ edge to the edited e_0.

When a sequence of operations could indicate that the resultant entity is the same as the original (such as an edit followed by a reversal of that edit), the fact that the entity has been through two processes is preserved. A link is not created between the two entities: partly as the more recent entity was created via

$$
\begin{array}{r}
V(\bar{c}_0, i_0) = e_0 \\
V(\bar{c}_1, i_1) = e_1 \\
Type(e_0) = source(s) \\
Type(e_1) = dest(s) \\
\hline
\bar{c}_n = \bar{c}_0 + \bar{x} \\
i_n = new(V, \bar{c}_n) \\
V' = V[(\bar{c}_n, i_n) \rightarrow e_n] \\
E' = E[[((\bar{c}_n, i_n), (\bar{c}_0, i_0)) \rightarrow isX(\bar{x})]] \\
[((\bar{c}_n, i_n), (\bar{c}_1, i_1)) \rightarrow s] \\
Type' = Type[e_n \rightarrow Type(e_0)]
\end{array}
$$

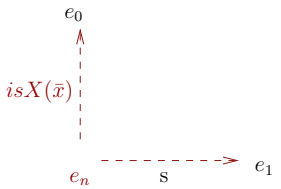


Fig. 4. $\langle V', E', e_n \rangle = extend(V, E, e_0, e_1, s, \bar{x})$. Extend e_0 with e_1, creating e_n. Dashed edges are created as a consequence of this operation.

$$
\begin{array}{r}
V(\bar{c}_0, i_0) = e_0 \\
\hline
\bar{c}_n = \bar{c}_0 + \bar{e} \\
i_n = new(V, \bar{c}_n) \\
V' = V[(\bar{c}_n, i_n) \rightarrow e_n] \\
E' = E[[((\bar{c}_n, i_n), (\bar{c}_0, i_0)) \rightarrow isE(\bar{e})]] \\
Type' = Type[e_n \rightarrow Type(e_0)]
\end{array}
$$

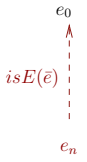


Fig. 5. $\langle V', E', e_n \rangle = edit(V, E, e_0, \bar{e})$. Edit e_0 to e_n.

a different process, and partly as detecting the match is not a simple automatic operation.

The edit operation only takes place on a non-extension entity: the x position in its positional co-ordinate must be zero. Edits of extension entities indicate that the target of either its s or isX edges have been altered, and so the original must be edited to point this edge to a new version. This results in an 'internal edit': these indicate that an edge has been retargeted. For example, cross-entity completion uses this to show that the s edge has to be retargeted; post-fact merging and pre-fact recall use this to show that the isX edge has to be retargeted. Internal edits therefore occur as a side-effect of completion operations, rather than directly via an edit operation.

3.3 Reduction

Reduction, denoted by $\xrightarrow{isR}$, indicates the removal of a state of knowledge (see Figure 6). Thus, if a measurement is found to be unnecessary or incorrect, the state of knowledge indicating that it extended the find can be removed. The state of knowledge is not deleted from the model: instead, new entities are created to omit the reduced entity.

$$
\begin{aligned}
V(\bar{c}_x, i_x) &= e_x \\
V(\bar{c}_y, i_y) &= e_y \\
V(\bar{c}_z, i_z) &= e_z \\
V(\bar{c}_0, i_0) &= e_0 \\
V(\bar{c}_1, i_1) &= e_1 \\
E((\bar{c}_y, i_y), (\bar{c}_x, i_x)) &= isX(\delta\bar{a}) \\
E((\bar{c}_z, i_z), (\bar{c}_y, i_y)) &= isX(\delta\bar{b}) \\
E((\bar{c}_y, i_y), (\bar{c}_0, i_0)) &= s \\
E((\bar{c}_z, i_z), (\bar{c}_1, i_1)) &= t \\
\hline
\bar{c}_n &= \bar{c}_z + \bar{r} \\
i_n &= new(V, \bar{c}_n) \\
\bar{d} &= \delta\bar{a} + \delta\bar{b} + \bar{r} \\
V' &= V[(\bar{c}_n, i_n) \to e_n] \\
E' &= E[[((\bar{c}_n, i_n), (\bar{c}_z, i_z)) \to isR(\bar{r})]] \\
&\quad [((\bar{c}_n, i_n), (\bar{c}_x, i_x)) \to isX(\bar{d})] \\
&\quad [((\bar{c}_n, i_n), (\bar{c}_1, i_1)) \to t] \\
Type' &= Type[e_n \to Type(e_z)]
\end{aligned}
$$

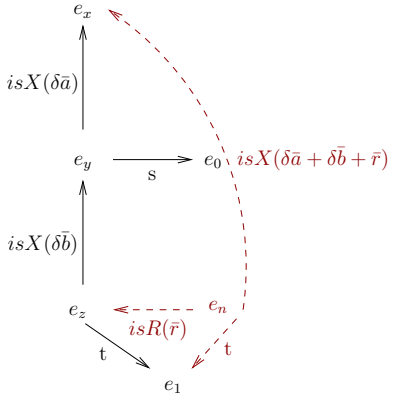


Fig. 6. $\langle V', E', e_n \rangle = reduce(V, E, e_y, s, e_0, e_z, t, e_1, e_x, \bar{x}, \bar{r}, \delta\bar{a}, \delta\bar{b})$. Removes the state of knowledge e_y and its associated entity e_0 from the knowledge graph.

To achieve this, the reduction operation requires two consecutive extension entities: the first of these (e_y) is removed by reducing the second (e_z). After the operation has been performed the state of knowledge of e_y is skipped, as the reduction of e_z (e_n) extends the root e_x directly. The isX weighting is a sum of the two original extension edges plus a reduction.

Due to the requirement of a second extension, reduction guarantees that the state of knowledge e_y is removed non-destructively. To remove e_z needs a slightly different approach, discussed as future work in Section 6.

3.4 Idempotence

While other systems make use of locking to prevent simultaneous operations on an entity, these represent alternatives in the Xeros model. If two archaeologists re-measure a vase, they may get two different measurements, so the two editions should be preserved. However, there are instances where an extension or edition may duplicate knowledge already present. Extension and edition idempotence rules (shown in Figure 7 and 8 respectively) are used to detect these situations and resolve them. The former holds when two extensions of an entity refer to the same entity via the same relation: two users adding the same note to the same find; the latter when two edits of the same entity have the same value: two users fixing the same spelling mistake in a note.

$$\begin{array}{rl}
V(\bar{c}_0, i_0) & = e_0 \\
V(\bar{c}_1, i_1) & = e_1 \\
V(\bar{c}_x, i_x) & = e_x \\
V(\bar{c}_y, i_y) & = e_y \\
E((\bar{c}_x, i_x), (\bar{c}_0, i_0)) & = isX(\bar{x}) \\
E((\bar{c}_y, i_y), (\bar{c}_0, i_0)) & = isX(\bar{x}) \\
E((\bar{c}_x, i_x), (\bar{c}_1, i_1)) & = s \\
E((\bar{c}_y, i_y), (\bar{c}_1, i_1)) & = s \\
s \notin \{isX, isE, isR\} & \\
\hline
c_x & = c_y
\end{array}$$

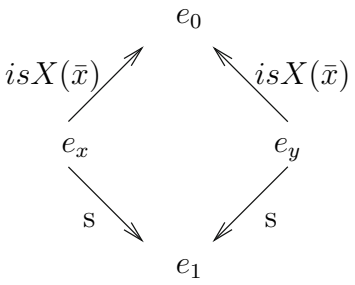


Fig. 7. Extension Idempotence: e_x and e_y are states of knowledge with the same entity e_1.

$$\begin{array}{rl}
V(\bar{c}_0, i_0) & = e_0 \\
V(\bar{c}_x, i_x) & = e_x \\
V(\bar{c}_y, i_y) & = e_y \\
E((\bar{c}_x, i_x), (\bar{c}_0, i_0)) & = isE(\bar{e}) \\
E((\bar{c}_y, i_y), (\bar{c}_0, i_0)) & = isE(\bar{e}) \\
value(e_x) & = value(e_y) \\
\hline
c_x & = c_y
\end{array}$$

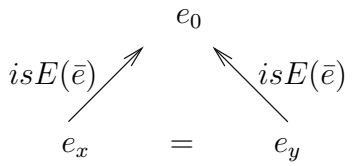


Fig. 8. Edition Idempotence: e_x and e_y are edits of e_0 that have the same value

4 Completion

The operations formalized above can be used as is, especially in a single-user scenario. However, knowledge structures are not shared: if a find has a note added and that note is then edited, the find will still refer to the old version of the note; if a find with a note is edited, the updated entity will no longer have its extension.

To address this, three 'completion' operations are defined: cross-entity completion ensures that an entity is brought up to date if an attached entity is edited; post-fact merging ensures that prior operations are performed on a newly-edited entity; pre-fact recall provides a way to infer states of knowledge that may not have been explicitly stored. These operations can be applied iteratively to the graph structure.

4.1 Cross-Entity Completion

It is possible that an edited entity may originally have been associated with another entity via extension. If a find is extended with a note, and that note is edited, the find should be updated so that it is extended with the new entity

$$
\begin{aligned}
V(\bar{c}_a, i_a) &= e_a \\
V(\bar{c}_b, i_b) &= e_b \\
V(\bar{c}_d, i_d) &= e_d \\
V(\bar{c}_1, i_1) &= e_1 \\
E((\bar{c}_b, i_b), (\bar{c}_a, i_a)) &= isE \\
E((\bar{c}_d, i_d), (\bar{c}_a, i_a)) &= s \\
E((\bar{c}_d, i_d), (\bar{c}_1, i_1)) &= isX \\
\hline
V', E', e_c &= extend(V, E, e_1, e_b, s, \bar{x}) \\
V'' &= V'[(\bar{c}_c, i_c) \rightarrow e_c] \\
E'' &= E'[[((\bar{c}_c, i_c), (\bar{c}_d, i_d)) \rightarrow isE]]
\end{aligned}
$$

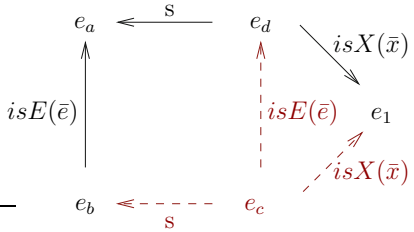


Fig. 9. $\langle V'', E'', e_c \rangle = cross(V, E, e_a, e_b, e_d, e_1, s, \bar{x}, \bar{e})$. Given the edit of e_a to e_b, where e_a is referred to by extension entity e_d, create e_c to bring the extension up to date.

rather than the original. Semantic and syntactic edits are treated as equal in this situation: in both cases, the associated entities must be updated to refer to the edited entity.

Cross-entity completion (see Figure 9) performs this process: assume that there is an entity e_1 that was extended with e_a via relationship s, resulting in an extension entity e_d, and that e_a was then edited to e_b. A completion entity e_c is created via an extension operation on e_1 that refers to the edited entity e_b via the same relationship s. Finally, an internal edit is created between the new extension entity e_c and the old e_d. As such it is shown that e_c is the state of knowledge in which e_1 is extended with the edited entity e_b.

4.2 Post-fact Merging

Post-fact merging ensures that the structures in the system are always representative of the latest operations to have occurred. A simple case is where a find has had some notes added, and is then edited: without a post-fact merge, the edited find would not retain the extensions performed earlier. The merge replicates the extensions onto the edited entity, thus bringing the graph up to date.

The process is also essential for asynchronous operations: it cannot be assumed that a user will perform an operation on the latest edition of an entity. A find may have been edited while a user was adding a note, or conversely a note may have been added while the user was editing the find. As such, any changes need to be performed on the updates that have occurred between the retrieval of the graph and the execution of an operation.

Given an entity e_a that is edited to e_b: if e_a is then extended, it follows that e_b may also be extended with the same extension entity. Alternatively, given an entity e_a that is extended and subsequently also edited, it follows that the edited version should also be extended with the same extension entity. Figure 10 shows the merging operation performing square completion given that an edit

$$
\begin{aligned}
V(\bar{c}_a, i_a) &= e_a \\
V(\bar{c}_b, i_b) &= e_b \\
V(\bar{c}_d, i_d) &= e_d \\
V(\bar{c}_1, i_1) &= e_1 \\
E((\bar{c}_b, i_b), (\bar{c}_a, i_a)) &= isE \\
E((\bar{c}_d, i_d), (\bar{c}_a, i_a)) &= isX \\
E((\bar{c}_d, i_d), (\bar{c}_1, i_1)) &= s \\
\hline
V', E', e_c &= extend(V, E, e_b, e_1, s, \bar{x}) \\
V'' &= V'[(\bar{c}_c, i_c) \rightarrow e_c] \\
E'' &= E'[[((\bar{c}_c, i_c), (\bar{c}_d, i_d)) \rightarrow isE]]
\end{aligned}
$$

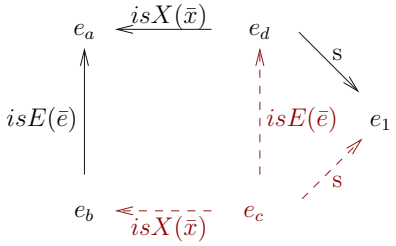


Fig. 10. $\langle V'', E'', e_c \rangle = merge(V, E, e_a, e_b, e_d, e_1, s, \bar{x}, \bar{e})$. Given the extension of e_a, where e_a has previously been edited, apply the extension to the edited entity e_b.

and extension have occurred on e_a (in any order). e_c is created as an extension of the edit entity e_b and an edit of the extension entity e_d. The merge performs an extension operation, ensuring that e_c also has the correct relation s to e_1.

Post-fact merging is most appropriate after edits where semantics are not altered, as the extensions will likely still apply. In cases where the meaning of an entity is altered, it may be prudent to either skip the post-fact merging step (i.e. requiring the user to re-extend the find with any notes) or to allow for the user to select any entities that should be added via the merge operation. It may, however, be more interesting to apply all existing extensions, so any interpretations that are no longer valid become apparent and provoke further annotations.

4.3 Pre-fact Recall

Pre-fact recall is complementary to post-fact merging: Where post-fact merging operates on the outer edges of the graph, pre-fact recall works on the inner structure. If a find is edited and then extended with a note, the structure suggests that there could be a state where the find was extended but not edited. This allows for the navigation of the graph through different permutations of operations - valuable for gaining new insights. In contrast to post-fact merging, pre-fact recall is a non-essential process and so can be determined post hoc. Pre-fact recall and post-fact merging result in the same graph: only the antecedent and consequent differ.

Given an entity e_c that has been produced as a result of an extension of e_b, which is in turn an edit of e_a, a completion entity e_d is created that is an extension of e_a. An internal edit edge is also created from e_c to e_d, hence completing the square. The formalization and visualisation of this is shown in Figure 11.

$$
\begin{aligned}
V(\bar{c}_a, i_a) &= e_a \\
V(\bar{c}_b, i_b) &= e_b \\
V(\bar{c}_c, i_c) &= e_c \\
V(\bar{c}_1, i_1) &= e_1 \\
E((\bar{c}_b, i_b), (\bar{c}_a, i_a)) &= isE \\
E((\bar{c}_c, i_c), (\bar{c}_b, i_b)) &= isX \\
E((\bar{c}_c, i_c), (\bar{c}_1, i_1)) &= s \\
\hline
V', E', e_d &= extend(V, E, e_a, e_1, s, \bar{x}) \\
V'' &= V'[(\bar{c}_d, i_d) \to e_d] \\
E'' &= E'[[((\bar{c}_c, i_c), (\bar{c}_d, i_d)) \to isE]]
\end{aligned}
$$

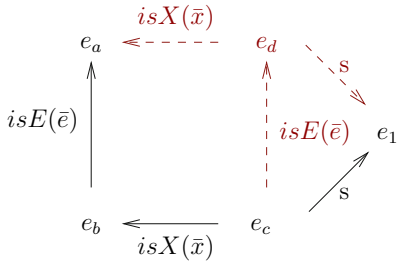


Fig. 11. $\langle V'', E'', e_d \rangle = recall(V, E, e_a, e_b, e_c, e_1, s, \bar{x}, \bar{e})$. Given the extension of e_b, where e_b is an edit of e_a, the extension can also be applied to e_a.

Naturally, extensions might only be applicable to the edited entity. Pre-fact recall does not guarantee that recalled entities are semantically valid, so it is suggested that they only be brought into the persisted state of the system if approved by a user.

5 Scalability

Completion operations can have a significant impact on graph size and, as interpretations build, an edit may result in many vertices and edges being created. Figure 12 shows a worked example: A find e_a is edited and extended with a note e_c, then the find is edited again. Finally, the note is edited. With core operations, 6 vertices and 5 edges are needed. With post-fact merging and cross-entity completion this grows to 9 vertices and 14 edges. Adding pre-fact recall, 11 vertices and 20 edges are used. This requires three cross-entity completion operations, due to the entity e_n created in the recall operation: optimised completion uses only two.

The number of references to an entity is significant: for every reference, 3 edges and an entity are created during cross-entity completion. If pre-fact recall is dynamically performed, there are fewer references and the overhead is thus reduced. Similarly, given an extension propagated over a chain of edits via pre-fact recall or post-fact merging, 2n edges and n vertices are created, where n is the number of entities preceding (for pre-fact) or following (for post-fact). If the system only operates on the most recent changes, post-fact has minimal overhead. The most costly post-fact merging operation would occur due to an extension at the oldest point in an edit chain.

Xeros therefore provides a flexible approach to building a scalable system: eager post-fact merging and cross-entity completion minimise storage while ensuring asynchronous operations are handled, and dynamic pre-fact recall provides for the navigation of potential entities in the system.

$$
\begin{aligned}
V_1, E_1, e_b &= edit(V_0, E_0, e_a, \bar{e}) && \diamond \\
V_2, E_2, e_d &= extend(V_1, E_1, e_b, e_c, hasNote, \bar{x}) && \diamond \\
V_3, E_3, e_n &= recall(V_2, E_2, e_a, e_b, e_d, e_c, s, \bar{x}, \bar{e}) && \\
V_4, E_4, e_e &= edit(V_3, E_3, e_b, \bar{e}) && \diamond \\
V_5, E_5, e_o &= merge(V_4, E_4, e_b, e_e, e_d, e_c, s, \bar{x}, \bar{e}) && \square \\
V_6, E_6, e_f &= edit(V_5, E_5, e_c, \bar{e}) && \diamond \\
V_7, E_7, e_g &= cross(V_6, E_6, e_c, e_f, e_d, e_b, s, \bar{x}, \bar{e}) && \square \\
V_8, E_8, e_h &= cross(V_6, E_6, e_c, e_f, e_n, e_a, s, \bar{x}, \bar{e}) && \\
V_9, E_9, e_i &= cross(V_6, E_6, e_c, e_f, e_o, e_e, s, \bar{x}, \bar{e}) && \square
\end{aligned}
$$

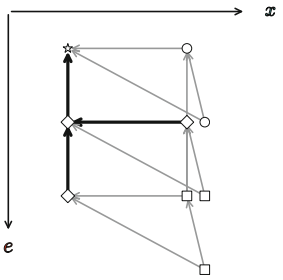


Fig. 12. Operations during a set of edits and extensions. Lines with ◇ are performed when no completion is used; lines with □ are added when post-fact and cross-entity completion are used; all lines are performed when every completion operation is used. The diagram shows only the actions on e_a for simplicity.

6 Conclusions

This paper has introduced the Xeros Data Model, its operations, and characteristics. It has been shown that the structure of the model allows for a variety of efficient storage approaches, and that it is robust against asynchronous operations.

Future work on Xeros will provide formalisations for the creation of groups to allow the aggregation of entities, and of the universe: an entity that refers to the leaves of the graph at a single point in time. Editions of the universe allow for these 'snapshots' to be navigated, and users can roll back to any edition of the universe. This approach also caters for the deletion of entities that cannot be removed via reduction: by removing a leaf reference from the universe, the entity can be omitted from visualisations but still exist in the graph.

An HCI study is also being performed, by way of the PoN web application. This is underpinned by the Xeros data model, and the system allows archaeologists to capture their finds and notes in an intuitive manner. Work on PoN will be further informed by the Xeros model, and requirements in the system will feed back into the model's development.

Acknowledgements. This research is funded in part by the EPSRC and AHRC PATINA project through the RCUK Digital Economy programme (EP/H042806/1).

References

[1] Buneman, P., Cheney, J., Lindley, S., Müller, H.: DBWiki: a structured wiki for curated data and collaborative data management. In: SIGMOD Conference 2011, pp. 1335–1338 (2011)

[2] Castro, L.J.G., Giraldo, O.X., Castro, A.G.: Using the Annotation Ontology in semantic digital libraries. In: 9th International Semantic Web Conference, ISWC 2010 (November 2010)
[3] Doerr, M., Ore, C.-E., Stead, S.: The CIDOC conceptual reference model: a new standard for knowledge sharing. In: Tutorials, Posters, Panels and Industrial Contributions at the 26th International Conference on Conceptual Modeling, ER 2007, vol. 83, pp. 51–56. Australian Computer Society, Inc., Darlinghurst (2007)
[4] Dowell, R.D., Jokerst, R.M., Day, A., Eddy, S.R., Stein, L.: The distributed annotation system. BMC Bioinformatics 2, 7 (2001)
[5] Eve, S., Hunt, G.: ARK: A development framework for archaeological recording. In: Layers of Perception. Proceedings of the 35th International Conference on Computer Applications and Quantitative Methods in Archaeology, CAA, pp. 1–5 (April 2007)
[6] Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H., Studer, R.: Semantic wikipedia. Journal of Web Semantics 5, 251–261 (2007)
[7] Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E., Van den Bussche, J.: The Open Provenance Model core specification (v1.1). Future Generation Computer Systems (July 2010)
[8] Moreau, L., Missier, P. (eds.) Belhajjame, K., Cresswell, S., Golden, R., Groth, P., Klyne, G., McCusker, J., Miles, S., Myers, J., Sahoo, S.: The PROV Data Model and Abstract Syntax Notation. W3c first public working draft, World Wide Web Consortium (October 2011)
[9] Rains, M.: Towards a computerised desktop: the integrated archaeological database system. In: Computer Applications and Quantitative Methods in Archaeology, pp. 207–210 (March 1994)
[10] Theodoridou, M., Tzitzikas, Y., Doerr, M., Marketakis, Y., Melessanakis, V.: Modeling and querying provenance by extending CIDOC CRM. Distributed and Parallel Databases 27, 169–210 (2010), doi:10.1007/s10619-009-7059-2