Provenance in Databases

Maxime Debosschere

Master Thesis

School of Information Technology
Hasselt University / transnational University Limburg

2012–2013

Abstract

An overview of the concept of *provenance* across various disciplines is followed by an analysis of the different facets related to provenance in databases. Two important models for exchanging provenance information, OPM and PROV, are discussed and compared with an emphasis on their temporal aspects. A reasoning tool for PROV instances is presented.

Contents

Introduction

The subject of this thesis is *Provenance in Databases*, an interesting and useful topic which deservedly receives a lot of research attention. Provenance, in essence, deals with the storage and retrieval of data about data (metadata) which helps to explain its origin. This, in turn, has many useful applications, ranging from data quality and trustworthiness assessment to license verification. The research question is *"In what ways can provenance information be represented and reasoned about?"* It is deliberately open to allow for a domain study which encompasses the various aspects related to this subject, both from a theoretical as from a practical point of view.

The thesis consists of three chapters, the first of which explores the notion of *provenance* across various fields (art and antiques, archival science, archaeology, geology) before arriving at databases. Different definitions and types of provenance in the context of databases are investigated. Furthermore, the importance of this subject is illustrated by means of examples and usage benefits, without neglecting possible downsides.

The second chapter is devoted to two popular models for the storage of provenance information: OPM and PROV. These models are analysed and compared in detail, with an emphasis on their temporal aspects.

A part of this thesis was putting the theory into practice by building a reasoning tool which, given a PROV instance, is able to reflect on its temporal properties. The final chapter provides insights on how this tool was developed.

A summary in Dutch is included as an appendix.

I wish to thank my family, promotor prof. dr. Jan Van den Bussche, and assessors prof. dr. Bart Kuijpers and dr. Tom Ameloot for their support and advice.

Maxime Debosschere
Diepenbeek, June 2013

Chapter 1

What is Provenance?

The word *provenance* stems from the Middle French *provenir*, meaning "come forth, arise". Its modern meaning is "(1) origin, source; (2) the history of ownership of a valued object or work of art or literature". This chapter aims to provide an overview of the different interpretations of provenance across those fields in which it is a prominent concept.

1.1 Art and antiques

The roots of provenance research lie in the world of art and antiques. Relevant information may vary from purchase receipts, exhibition catalogues and private correspondence to expert certifications and the chemical composition of paint.[1] A large number of art institutes, including The Metropolitan Museum of Art[2] (New York) and The Cleveland Museum of Art,[3] have research groups devoted to the investigation of ownership history.

The International Foundation for Art Research (IFAR) defines provenance as follows:[4]

> *The provenance of a work of art is a historical record of its ownership, although a work's provenance comprehends far more than its pedigree. The provenance is also an account of changing artistic tastes and collecting priorities, a record of social and political alliances, and an indicator of economic and market conditions influencing the sale or transfer of the work of art.*

A detailed history of a work of art yields several significant benefits:

- Evidence of authenticity. Provenance information should be seen as a contributing factor rather than solid proof. Forgers might falsify the provenance of a work by reproducing accompanying documents such as receipts of sale. The infamous counterfeiter John Drewe, for example, managed to

[1] `http://www.xamou-art.co.uk/provenance-art-appraisal-and-authentication/` [Retrieved February 20 2013]

[2] `http://www.metmuseum.org/research/provenance-research-project` [Retrieved January 28 2013]

[3] `http://www.clevelandart.org/provenance-research` [Retrieved January 28 2013]

[4] `http://www.ifar.org/provenance_guide.php` [Retrieved January 21 2013]

sell numerous forged paintings to various high-profile art connoisseurs by creating false accompanying provenance documents (Landesman, 1999).

- Verified additional information. Accurate documentation may shed light on the artist, creation date or portrayed subject.
- Increased value. Art collectors are willing to pay considerably more for art items of which the history is confirmed (Sullivan, 2005).
- Proof of legitimate ownership. The declassification of World War II records in the 1990's sparked a sudden increase in public awareness regarding stolen art in Europe during the Nazi era (1933–1945) and consequently many restitution claims by rightful owners emerged. This newfound awareness was further augmented by The Washington Conference on Holocaust-Era Assets,[5] in which a desire was stated for widespread efforts to identify unrestituted Nazi-confiscated art and to make relevant provenance records and archives accessible to researchers (Washington Conference on Holocaust-Era Assets, 1998). Database portals related to this issue are The Project for the Study of Collecting and Provenance (PSCP)[6] by The Getty Research Institute, Nazi-Era Provenance Internet Portal,[7] The Central Registry on Information on Looted Cultural Property 1933–1945,[8] and The Art Loss Register.[9] Several detailed examples of resolved Nazi-era restitution claims on the Museum of Fine Arts[10] (Boston) website demonstrate how a complete history of a work of art can help to determine its legitimate owner.

Provenance enjoys a widespread attention in the world of music instruments. For, inter alia, the benefits given above, classical music aficionados are interested in knowing previous owners and players of particular instruments. Research has also been conducted to determine the age and geographical origin of wood from stringed instruments (Bernabei and Bontadi, 2011).

1.2 Achival Science

In its *Glossary of Archival and Records Terminology*, the Society of American Archivists (SAA) defines provenance as "1. The origin or source of something. - 2. Information regarding the origins, custody, and ownership of an item or collection."[11] Clearly, only the second meaning is tailored to the archival community. The glossary further states:

> *Provenance is a fundamental principle of archives, referring to the individual, family, or organization that created or received the items*

[5] http://www.state.gov/www/regions/eur/wash_conf_material.html [Retrieved January 28 2013]
[6] http://www.getty.edu/research/tools/provenance/index.html [Retrieved January 31 2013]
[7] http://nepip.org [Retrieved February 21 2013]
[8] http://www.lootedart.com [Retrieved January 31 2013]
[9] http://www.artloss.com/en [Retrieved January 31 2013]
[10] http://www.mfa.org/collections/provenance [Retrieved February 21 2013]
[11] http://www2.archivists.org/glossary/terms/p/provenance [Retrieved February 21 2013]

in a collection. The principle of provenance or respect des fonds *dictates that records of different origins (provenance) be kept separate to preserve their context.*

The following fragment (Gilliland-Swetland, 2000) makes clear that the concepts of "principle of provenance" and "respect des fonds" are similar but not identical:

> *The principle of* respect des fonds *was first codified in 1839 in regulations issued by the French minister of public instruction. The principle stated that records should be grouped according to the nature of the institution that accumulated them. In 1881, the Prussian State Archives issued more precise regulations on arrangement that defined* Provenienzprinzip*, or the principle of provenance. The principle of provenance has two components: records of the same provenance should not be mixed with those of a different provenance, and the archivist should maintain the original order in which the records were created and kept.*

1.3 Archaeology

In the field of Archaeology the words *provenance* and *provenience* are seemingly used interchangeably. The Program on Ancient Technologies and Archaeological Materials (ATAM), a division of the Illinois State Archaeological Survey (ISAS) at the University of Illinois at Urbana-Champaign, mentions three published definitions:[12]

- *Provenience: the place of origin of a specimen (Jolly and White, Physical Anthropology and Archaeology 1995:468)*
- *Provenience (provenance): the geographical or geological origin or source of an artifact (Rice, Pottery Analysis 1987:480)*
- *Provenience: archaeological context, or the "horizontal and vertical position" within the surrounding sediments in an excavation (Renfrew and Bahn, Archaeology: Theories Method and Practice 1996:46)*

There are various discussions on public debate platforms about the difference of context between both words.[13] Some archaeologists suggest that the words can be used completely interchangeably and that the difference is merely linguistic between Americans and Europeans. Others claim that *provenience* specifically refers to the location and conditions in which an artifact was found whereas *provenance* relates to the history of an object in terms of ownership. More interpretations exist and it appears that there will be no consensus on this matter soon.

[12] `http://www.isas.illinois.edu/atam/research/ceramics/ceramicsprovenance.html` [Retrieved April 11 2013]

[13] `http://archaeology.about.com/b/2006/05/09/provenience-or-provenance-a-poll.htm` [Retrieved April 11 2013]

1.4 Geology

The *Encyclopedia of Science & Technology* entry for provenance in the field of (sedimentary) geology is as follows (Ingersoll, 2007):

> (...) *all characteristics of the source area from which clastic sediments and sedimentary rocks are derived, including relief, weathering, and source rocks.*

A review on novel sedimentary provenance techniques contains a more extensive definition (Haughton et al., 1991):

> *The study of sedimentary provenance interfaces several of the mainstream geological disciplines (mineralogy, geochemistry, geochronology, sedimentology, igneous and metamorphic petrology). Its remit includes the location and nature of sediment source areas, the pathways by which sediment is transferred from source to basin of deposition, and the factors that influence the composition of sedimentary rocks (e.g. relief, climate, tectonic setting).*

1.5 Databases

The most recent contextualization of the word *provenance* occurred in the field of databases. When retrieving information from a database it might be interesting to know exactly where the data came from, what changes it underwent and what processes caused those changes. Provenance aims to answer those questions, which in turn aids the user in assessing the quality and reliability of the data. Synonyms for database provenance are *lineage* or *pedigree*, although the former is often exclusively used in the context of *why*-provenance (which is discussed later). The first publication on provenance dates back from 1986, but research didn't really take off until the early 2000's (Moreau, 2010). The first provenance workshop took place in 2002 and today it is a trending topic which enjoys a considerable research interest. This is not surprising as the need for reliable information has never been higher.

It is important to note that the term *database* should be seen in a bigger context than the traditional closed systems: it encompasses all known data storage facilities, whether they are open or closed, structured or unstructured. In a way the entire World Wide Web could be regarded as a giant database. Some literature solely focuses on provenance technology in function of traditional databases, the Web or e-Science, but this distinction is not made in this thesis.

1.5.1 Types of Provenance

Multiple descriptions or definitions for provenance have been proposed, some of which are more detailed than others. Below are just a few of many interpretations.

> *Data provenance — sometimes called lineage or pedigree — is the description of the origins of a piece of data and the process by which it arrived in a database.* (Buneman et al., 2001)

> *Information concerning the origin of data.* (Buneman et al., 2007)
>
> *The details regarding the sources and origins of information (e.g., author, publisher, citations, etc.) are referred to as provenance, and they serve as a means to evaluate trust.* (Artz and Gil, 2007)
>
> *Provenance information describes the origins and the history of data in its life cycle. Such information (also called lineage) is important to many data management tasks.* (Cheney et al., 2009a)
>
> *The provenance of digital objects represents their origins.* (Gil et al., 2013)

The W3C PROV Incubator Group acknowledges that there can't be all-encompassing definition of provenance and developed an application-specific working definition instead (Gil et al., 2010):

> *Provenance of a resource is a record that describes entities and processes involved in producing and delivering or otherwise influencing that resource. Provenance provides a critical foundation for assessing authenticity, enabling trust, and allowing reproducibility. Provenance assertions are a form of contextual metadata and can themselves become important records with their own provenance.*

Several specific types of provenance exist. A first important distinction, that between *where*- and *why*-provenance, was made in 2001. The former kind asks where a given piece originated from and is important for understanding the source of errors in the data, and for carrying annotations through database queries (Buneman et al., 2001). Why-provenance, on the other hand, provides an explanation for the existence of a piece of data in a database. A third notion, *how*-provenance, was devised later and allows for a mathematical approach to determine how source tuples contributed to a result (Green et al., 2007). It has been shown that *why*-provenance can be extracted from *how*-provenance (Cheney et al., 2009a).

Depending on which data is considered useful, a few perspectives emerge. One such perspective is *process*-based, in which the history of a piece of data is viewed in terms of the process that led to that piece of data (Moreau, 2010). It captures the actions and steps taken to generate the information in question (Gil et al., 2013). This definition is very broad as it may encompass many different aspects, ranging from the executed software, input data and interacting users to the electricity used in running the system (Moreau, 2010). Other perspectives focus on *agents* (entities involved in creating or manipulating data) or *objects* (the source artifacts from which a combined artifact was made) (Gil et al., 2013).

A lesser known provenance mechanism is storage in annotations. In some ontologies, for example Dublin Core, provenance-related information, such as author, creation date and version, can be saved (Moreau, 2010). Efforts are being made in establishing a mapping between the Dublin Core terms vocabulary and the PROV-O ontology.[14,15]

[14] `http://dublincore.org/groups/provenance/` [Retrieved April 14 2013]
[15] `http://www.w3.org/TR/prov-dc/` [Retrieved April 14 2013]

1.5.2 Importance

Some of the definitions above already touched upon the advantages of provenance. Its usefulness has also been demonstrated by means of 33 use cases (Gil et al., 2010). They cover a broad range of different topics and were reviewed and assembled into three flagship scenarios. These illustrate several of the possibilities which provenance has to offer:

- A news blog gathering news from various information sources: checking license policies, integrating and versioning unstructured content such as documents and media, aggregating and tracking content usage, checking authority, verifying original sources, ...
- Disease outbreak analysis: integrating data from different sources (which may use different representations), archiving and reusing data, justifying decisions made based on data, allowing for analysis repeatability, ...
- Business contract: comparing realisations to obligations, understanding object derivations, verifying authority, ...

Besides these case-specific utilities there are numerous global benefits. One of the key advantages provided by provenance information is that it enables the evaluation of data trustworthiness. Although *trust* is a multifaceted notion, it often revolves around the reliability of the information source (Artz and Gil, 2007). Knowledge of the data origin and subsequent processing enables users to place the data in context, which in turn provides the user with the means to make a trust judgment (Jagadish and Olken, 2004).

In the specific case of the Semantic Web, provenance establishes a relation between people and information, while the web contains social network data to infer trust between people (Artz and Gil, 2007). Hence, provenance can be regarded as a platform for trust algorithms on the web (Groth et al., 2012; Carroll et al., 2005).

Very closely related to examining data trustworthiness is data quality assessment: the former focuses on the origin and processing of data while the latter has more to do with the information itself. Quality, as with trust, is no univocal notion. One publication vaguely describes it as "*fitness for use*" (Tayi and Ballou, 1998), but another study defines several quality dimensions: syntactic and semantic accuracy, completeness, timeliness, consistency, and dimension trade-offs (Scannapieco et al., 2005). Another investigation identifies 16 different data quality dimensions (Pipino et al., 2002), which illustrates the ambiguity surrounding the concept of *data quality*.

Datasets in the public domain are prone to errors and this can have grave consequences. In genome databases, for example, faulty data can cause a considerable economic and medical impact (Müller and Naumann, 2003). It is therefore important to develop techniques to determine the level of quality, and provenance has been proposed as a data quality metric (Simmhan et al., 2006). Its potential utility in the quality assessment process is well-defined (Simmhan et al., 2005):

> *Data quality of source data is important since errors introduced by faulty data tend to inflate as they propagate to data derived from*

> *them.* (...) *The level of detail included in the provenance determines the extent to which the quality of the data can be estimated. Rudimentary lineage metadata about the data* (...) *can assist the data user in establishing the authenticity of the data and avoid spurious sources.*

Provenance is a critical component in the investigation of *causality*, which determines how source tuples are involved in a query result. This, in turn, leads to useful applications such as explaining unexpected answers and handling aggregate queries (Meliou et al., 2010). Related to this is the question why a particular tuple is *not* in a result set, the so-called *Why Not?*-provenance. A framework which automates the traditional debugging process for explaining non-answers has been developed (Chapman and Jagadish, 2009).

1.5.3 Complications

The usage of provenance information comes with several downsides. Arguably the biggest concerns are related to *security* and *privacy*. Especially when dealing with sensitive information, such as medical reports or corporate financial data, security is of vital importance (Braun et al., 2008). Nonetheless traditional security systems are unable to handle provenance data for two reasons.

First, there can be multiple levels of confidentiality. Even though provenance information is metadata, it can be more sensitive than the data it is attached to. An example of this is an employee's performance review: the employee should have access to the output (the review report), but not to those elements which created the document (provenance), such as the contributors for specific parts of the text (Braun et al., 2008). Hence, the data and the provenance require different security standards.

Second, provenance focuses on relationships between items, whereas traditional security systems work with individual data items (Braun et al., 2008).

A basic security model for provenance has been developed (Braun and Shinnar, 2006).

Another complication of provenance is related to its storage: it may require many times more disk space than the base data, depending on the granularity and level of detail. However, improvements have been made: one strategy managed to reduce the size of provenance up to a factor of 20 (Chapman et al., 2008).

Other concerns related to provenance include incompleteness (missing provenance for particular needs), unreliability (unclear or forged provenance), heterogeneity (different kinds of provenance might be difficult to merge) and a lack of precise, formal and repeatable research (Cheney et al., 2009b).

Chapter 2

OPM and PROV

The first chapter outlined the basic types and benefits of provenance in databases. The remainder of this thesis focuses on the practical aspects of provenance storage and retrieval. This chapter is an analysis of two popular models for provenance information.

2.1 OPM

2.1.1 Model Description

OPM, which stands for *Open Provenance Model*, was conceived and developed during a series of public challenges and workshops. The first *Provenance Challenge* aimed to understand the expressiveness and potential of different systems (Moreau et al., 2008). The succeeding challenge was an attempt at reaching system interoperability.[1] First drafts of the OPM specification were developed during workshops in August 2007 and June 2008, and a third challenge focused on testing these new blueprints (Simmhan et al., 2011). Based on the findings of these tests a new version (*v1.1*) of the model was released. The fourth and last challenge was to apply OPM to broad end-to-end scenarios.[2]

Note that, unless otherwise indicated, all information in this subsection is extracted from *v1.1* of the *Open Provenance Model Core Specification* (Moreau et al., 2011).

OPM aims to address the following needs:

- *To allow provenance information to be exchanged between systems, by means of a compatibility layer based on a shared provenance model.*
- *To allow developers to build and share tools that operate on such provenance model.*
- *To define provenance in a precise, technology-agnostic manner.*

[1] `http://twiki.ipaw.info/bin/view/Challenge/SecondProvenanceChallenge` [Retrieved May 6 2013]

[2] `http://twiki.ipaw.info/bin/view/Challenge/FourthProvenanceChallenge` [Retrieved May 6 2013]

- *To support a digital representation of provenance for any thing, whether produced by computer systems or not.*
- *To allow multiple levels of description to coexist.*
- *To define a core set of rules that identify the valid inferences that can be made on provenance representation.*

The primary purpose of OPM is data exchange and there are no guidelines on how provenance should internally be stored in systems. Furthermore no particular digital representations are endorsed, although implementations in various formats, including XML,[3] exist.

An OPM instance can be regarded as a directed graph which depicts causal dependencies (arrows) between the various "things" (nodes). Each node is one of three types:

- **Artifacts** (denoted with ellipses): "*Immutable piece of state, which may have a physical embodiment in a physical object, or a digital representation in a computer system.*" A keyword here is *immutable*: while it is recognised that an artifact can have multiple states, the model only captures one specific snapshot from the past. Hence, multiple states require multiple artifacts. The definition further implies that OPM can be employed to represent real-world things (e.g., a car).
- **Processes** (denoted with rectangles): "*Action or series of actions performed on or caused by artifacts, and resulting in new artifacts.*" As with artifacts, these need not be digital (e.g., driving).
- **Agents** (denoted with octagons): "*Contextual entity acting as a catalyst of a process, enabling, facilitating, controlling, or affecting its execution.*" People can be agents (e.g., a race pilot).

Edges between the nodes defined above depict the causal dependencies. Source nodes are effects and target nodes are causes. Depending on the types of nodes it connects, several dependencies are possible (see Figure 2.1):

- A ***used*** edge from a process P to an artifact A depicts a causal relationship in which P required the availability of A.
- A ***wasGeneratedBy*** edge from an artifact A to a process P depicts a causal relationship in which A required the availability of P. This implies that A was generated after the begin of P.
- A ***wasTriggeredBy*** edge between two processes depicts a causal relationship in which a process P_2 required the availability of another process P_1. This implies that P_1 began before P_2 was able to complete.
- A ***wasDerivedFrom*** edge from an artifact A_2 to an artifact A_1 depicts a causal relationship in which A_2 required the availability of A_1. This implies that A_1 was generated before A_2.
- A ***wasControlledBy*** edge between a process P and an agent Ag depicts a causal relationship in which Ag required the availability of P.

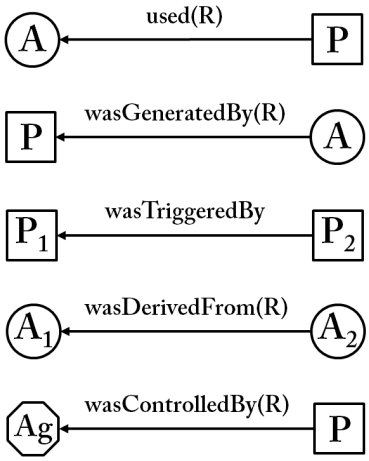

Figure 2.1: Different dependencies in OPM.

The *wasTriggeredBy* and *wasDerivedFrom* edges are actually summaries that enable process-based and an artifact-based views, respectively. The following derivations, referred to as *completion rules*, can be applied.

The occurrence of a *wasTriggeredBy* edge from a process P_2 to another process P_1 implies the existence of an artifact A which is used by P_2 and generated by P_1. Hence, it is always legal to introduce an artifact between two directly connected processes. The reverse action, that of eliminating an artifact, is allowed as well.

In the case of *wasDerivedFrom* edges only one operation is allowed: process introduction. This is because an artifact A_2 derived from another artifact A_1 implies the existence of a process P which generated A_2 and used A_1. The removal of a process in between two artifacts, however, is not always justified: a process might have connected two artifacts in different contexts, which means that a causal dependency between them might lack.

In order to find out if an object indirectly contributed to a future result, i.e., indirect causes, *multi-step inferences* can be constructed. These inferred edges may be depicted in graphs by dashed lines.

- A multi-step *wasDerivedFrom* edge $a_1 \rightarrow^* a_2$ is a transitive closure of *wasDerivedFrom* edges and expresses that artifact a_2 had a direct or an indirect effect on artifact a_1.
- A multi-step *used* edge $p \rightarrow^* a$ expresses that process p used artifact a or an artifact derived from a.
- A multi-step *wasGeneratedBy* edge $a \rightarrow^* p$ expresses that artifact a, or an artifact from which a was derived, was generated by process p.
- A multi-step *wasTriggeredBy* edge $p_1 \rightarrow^* p_2$ expresses that process p_1 used an artifact which was directly or indirectly generated by process p_2.

[3] `http://openprovenance.org/model/opmx` [Retrieved May 4 2013]

It is mandatory to define a *role* for each *used*, *wasGeneratedBy*, and *wasControlledBy* edge. This is because more than one of these edges might be connected to a process (e.g., three *used* edges between a process and an artifact indicate that the process required the availability of the same artifact three times). These roles, which are tags attached to the dependency arrows, help to distinguish multiple edges between the same components. They do not have semantic meanings, nor do they exist outside their local scopes. When a role is unknown the reserved value *undefined* can be used.

Both in the real and digital world, time-related information is ubiquitous: a document was created at x, a train arrived at y, someone launched a computer program at z, ... It is only logical that OPM offers the ability to include this information into the graph as well. This is possible by creating optional timestamps indicating the creation and use times of artifacts, and the begin and end times of processes. All the different dependencies can be extended with their time of occurrence, except for *wasControlledBy* which holds both begin and end times. Based on available time observations it is possible to perform some basic reasoning: an artifact must exist before being used, a process must begin before generating an artifact, ... Note, however, that this temporal information cannot be utilized to determine causality, as it's perfectly possible to model "impossible" scenarios which conflict with the basic reasoning assumptions.

OPM allows multiple instances, of different granularities, to overlap. Such descriptions are called *accounts* and their members are those components which refer to their unique account identifiers. Every OPM component can be a member of zero, one or multiple accounts.

The OPM *annotation framework* permits the addition of extra information, such as subtypes or descriptions, to graphs, nodes, edges, accounts, roles, or annotations.

Virtual collections of OPM instances can be created by attaching the graphs to *profiles*. These facilitate the enforcement of community-specific best practices and usage guidelines.

2.1.2 Formal Account

The lack of a formal semantics in OPM was addressed with the development of a formalized notion of an OPM graph and the addition of a temporal theory. Only a subset of OPM without agents was considered. The information in this section was extracted from *A Formal Account of the Open Provenance Model* (Kwasnikowska et al., 2010).

Edges with roles are now referred to as *precise edges*, denoted as $x \xrightarrow{r} y$, with r being the role. This "r" can be replaced by a "!" in case the role itself is irrelevant. Edges without a role are *imprecise edges*.

OPM graphs are *legal* if they meet two conditions. First, every artifact is involved in at most one precise *wasGeneratedBy* edge. Second, for every precise *wasDerivedFrom* edge $A \xrightarrow{r} B$, a process P exists such that $A \xrightarrow{!} P$ and $P \xrightarrow{r} B$. Note that the role from the derivation equals the one from the use. This construction is called a *use-generate-derive triangle* (see Figure 2.2) and it allows

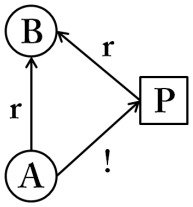

Figure 2.2: A use-generate-derive triangle (A, B, P, r).

for ascertaining that A was directly influenced by B. Its notation is (A, B, P, r).

In the formalized notion, the different timepoints (the start time of a process, the end time of a process, the creation time of an artifact, the usage time of an artifact) are written as $begin(P)$, $end(P)$, $create(A)$, and $use(P, r, A)$, respectively, where A is an artifact, P a process and r a role. They are referred to as *temporal variables*.

A *temporal interpretation* of an OPM instance is a triple $(T, \leq, \tau)$ in which:

- T is a set of OPM timepoints,
- $\leq$ is a partial order on T,
- τ maps the set of all temporal variables to T, i.e., it contains every implied time constraint.

Recall that time information is optional. If every time placeholder in an instance contains a value, only one temporal interpretation for that instance exists. The number of possible interpretations goes up as the number of omitted timestamps increases.

It is perfectly possible for a legal OPM graph with a nonsensical temporal interpretation to occur. Before being able to verify the validity of a specific interpretation, several more aspects of OPM need to be formalized.

An expression of the form $u \leq v$, where u and v are both temporal variables, is called an *inequality*. A temporal interpretation $(T, \leq, \tau)$ *satisfies* an inequality $u \leq v$ if $\tau(u) \leq \tau(v)$. The *temporal theory* of an OPM instance is the set of inequalities expressed by 8 axioms:

- AX 1: For every process: $\boldsymbol{begin(P) \leq end(P)}$.
- AX 2: For every precise *wasGeneratedBy* edge $A \xrightarrow{!} P$: $\boldsymbol{begin(P) \leq create(A) \leq end(P)}$.
- AX 3: For every precise *used* edge $P \xrightarrow{r} A$: $\boldsymbol{begin(P) \leq use(P, r, A) \leq end(P)}$ and $\boldsymbol{create(A) \leq use(P, r, A)}$.
- AX 4: For every imprecise *wasDerivedFrom* edge $A \rightarrow B$: $\boldsymbol{create(B) \leq create(A)}$.
- AX 5: For every imprecise *wasGeneratedBy* edge $A \rightarrow P$: $\boldsymbol{begin(P) \leq create(A)}$.
- AX 6: For every imprecise *used* edge $P \rightarrow A$: $\boldsymbol{create(A) \leq end(P)}$.

- AX 7: For every *wasInformedBy* edge $P \rightarrow Q$: $\boldsymbol{begin(Q) \leq end(P)}$.
- AX 8 (*triangle axiom*): For every *use-generate-derive triangle* (A, B, P, r): $\boldsymbol{use(P, r, B) \leq create(A)}$.

A temporal interpretation $(T, \leq, \tau)$ of an instance which satisfies every inequality from the temporal theory is called a *temporal model* of that instance: the ordering of timepoints is completely valid.

The inequalities which make up the temporal theory all apply to direct (*single-step*) edges. However, new inequalities can be inferred from this. As an example, consider an *imprecise used* edge $P \rightarrow A$ and an imprecise *wasGeneratedBy* edge $A \rightarrow Q$. From axiom 6 follows that $create(A) \leq end(P)$ and from axiom 5 follows that $begin(Q) \leq create(A)$. From these two inequalities a new inequality can be inferred: $begin(Q) \leq end(P)$. In general, when an inequality $u \leq v$ is satisfied in every temporal model of an instance, it is called a *logical consequence* of that instance.

An inequality can be inferred by repeatedly applying transitivity, but a much more effective way is to simply look at graph patterns. Indeed, a fixed set of rules (inequalities that make up graph patterns) allows to graphically determine whether or not an inequality belongs to the temporal theory.

The building blocks which make up the graph patterns are discussed first: *edge-inference rules*. These rules create new inferences based on existing inequalities. They are defined as *multi-step edges* in the OPM core specification and express a direct or indirect influence of one object over another. Four edge-inference rules are trivial: existing *wasDerivedFrom*, *wasGeneratedBy*, *used* and *wasInformedBy* edges in the OPM graph (regardless of the presence of roles) are also inferred edges. Four other, non-trivial, edge-inference rules are listed.

- (Fig. 2.3a): $A \dashrightarrow B$ and $B \dashrightarrow C$ infers $A \dashrightarrow C$ (*wasDerivedFrom*).
- (Fig. 2.3b): $A \dashrightarrow B$ and ($B \rightarrow P$ or $B \overset{!}{\rightarrow} P$) infers $A \dashrightarrow P$ (*wasGeneratedBy*).
- (Fig. 2.3c,d): $A \dashrightarrow B$ and ($P \rightarrow A$ or $P \overset{!}{\rightarrow} A$ or $A \overset{!}{\rightarrow} P$) infers $P \dashrightarrow B$ (*used*).
- (Fig. 2.3e,f): $A \dashrightarrow Q$ and ($P \dashrightarrow A$ or $A \overset{!}{\rightarrow} P$) infers $P \dashrightarrow Q$ (*wasInformedBy*).

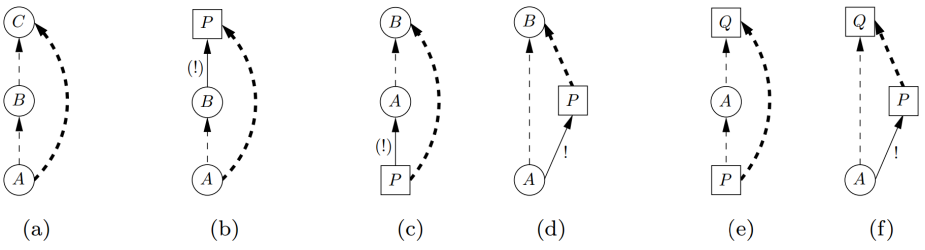

Figure 2.3: Non-trivial edge-inference rules. From (Kwasnikowska et al., 2010).

It is now possible to define graph patterns to perform temporal inferences. Formally, an inequality $u \leq v$ over the temporal variables of a legal OPM graph, where $u \neq v$, is a logical consequence of that graph if and only if (i) it already belongs to the temporal theory, i.e., the axioms, or (ii) if it matches one of the following inequalities:

1. $\boldsymbol{create(B) \leq create(A)}$, for an inferred *wasDerivedFrom* edge $A \dashrightarrow B$. This incorporates AX 4.
2. $\boldsymbol{begin(P) \leq create(A)}$, for an inferred *wasGeneratedBy* edge $A \dashrightarrow P$. This incorporates AX 5.
3. $\boldsymbol{create(A) \leq end(P)}$, for an inferred *used* edge $P \dashrightarrow A$. This incorporates AX 6.
4. $\boldsymbol{begin(Q) \leq end(P)}$, for an inferred *wasTriggeredBy* edge $P \dashrightarrow Q$. This incorporates AX 7.
5. $\boldsymbol{create(B) \leq use(P, r, A)}$, for a *used* edge $P \xrightarrow{r} A$ and an inferred *wasDerivedFrom* edge $A \dashrightarrow B$.
6. $\boldsymbol{begin(Q) \leq use(P, r, A)}$, for a *used* edge $P \xrightarrow{r} A$ and an inferred *wasGeneratedBy* edge $A \dashrightarrow Q$.
7. $\boldsymbol{use(P, r, C) \leq create(A)}$, for a *use-derive-generate triangle* (B, C, P, r) and an inferred *wasDerivedFrom* edge $A \dashrightarrow B$.
8. $\boldsymbol{use(P, r, B) \leq end(Q)}$, for a *use-derive-generate triangle* (A, B, P, r) and an inferred *used* edge $Q \dashrightarrow A$.
9. $\boldsymbol{use(P, r, B) \leq use(Q, s, A)}$, for a *use-derive-generate triangle* (C, B, P, r) and a *used* edge $Q \xrightarrow{s} A$, where (i) $A = C$ or (ii) $A \dashrightarrow C$.

These rules have been proven to be both sound (they represent valid inferences) and complete (every inequality that is a logical consequence of the axioms, but which is not included in the axioms, can be inferred by one of the patterns). Axioms 4 to 7 no longer have to be considered when deriving temporal inferences since this list incorporates them. Figure 2.4 presents a visual representation of the graph patterns.

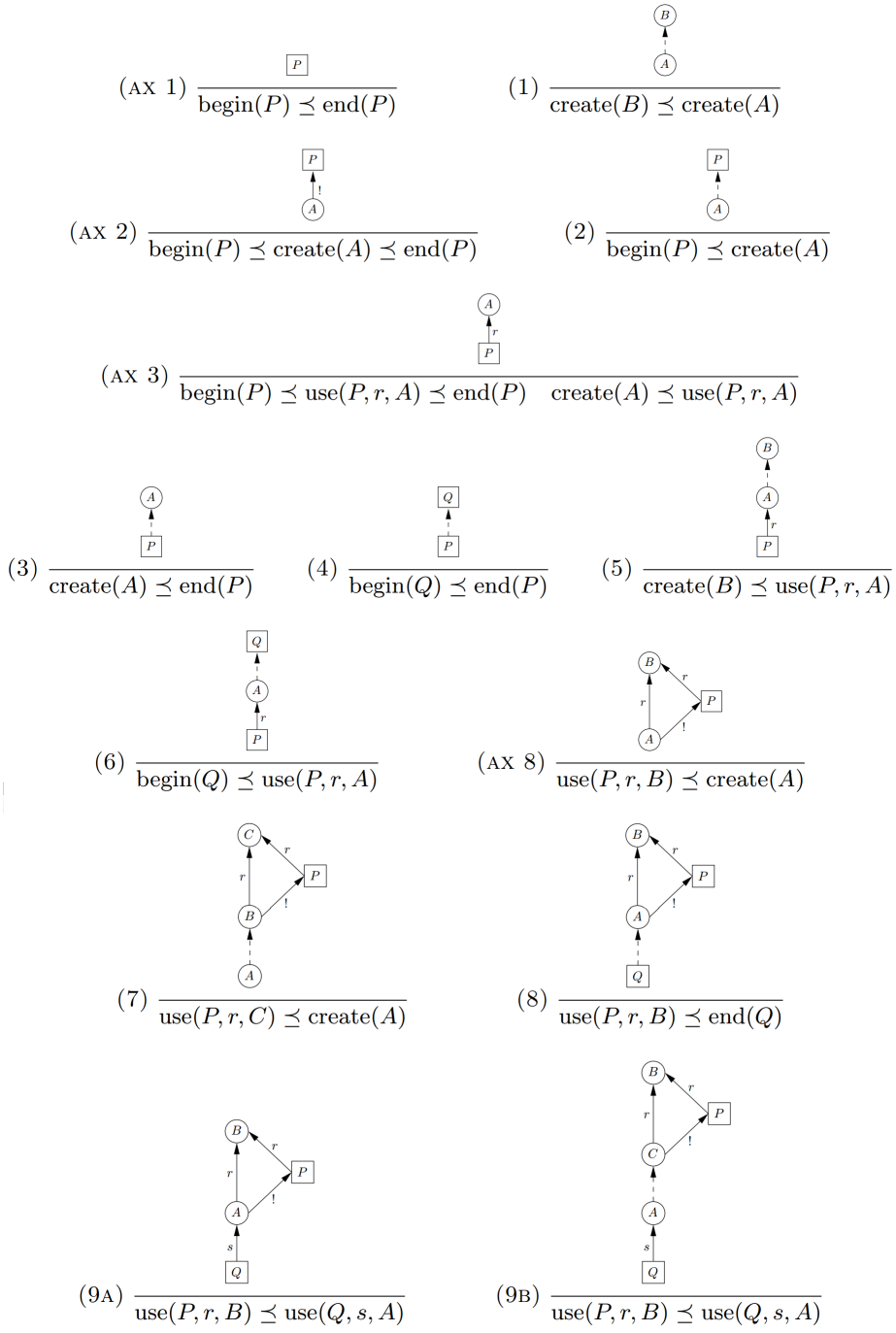

Figure 2.4: Graph patterns for deriving temporal inferences. From (Kwasnikowska et al., 2010).

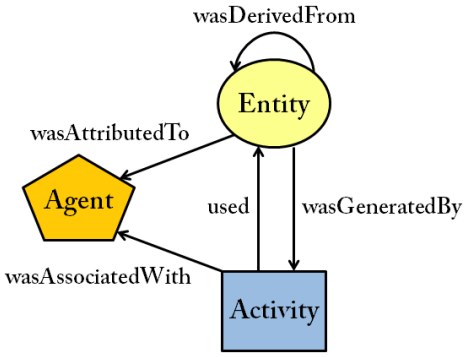

Figure 2.5: Core structures in PROV.

2.2 PROV

PROV is a World Wide Web Consortium (W3C) specification for storing provenance data. The information in this section was extracted from several W3C reports: the W3C Incubator Group XG Final Report (Gil et al., 2010) and the W3C Working Group PROV Model Primer (Gil et al., 2013), PROV Data Model (Moreau et al., 2013) and PROV Data Model Constraints (Cheney et al., 2013).

2.2.1 Model Description

The PROV Data Model comprises six components, all dealing with specific aspects of provenance. However, only the first three components contain *core structures*: structures which capture the essence of provenance (refer to Figure 2.5 for a graphical overview). The remaining components consist of *extended structures* which improve expressibility. PROV information can be modelled as a graph where nodes express *base types* (entities, activities and agents) and edges express *relations* (generation, usage, derivation, ...). In the following summary, mandatory parameters are denoted with dotted underlines.

1. **Entities and activities.** This is the only component which captures time-related information.

   - *"An **entity** is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary."* Every entity contains an identifier and a set of attribute-value pairs. The time between its generation and invalidation is called the *lifetime*. Entities are a part of the PROV core and are graphically represented by means of yellow ovals.
   - *"An **activity** is something that occurs over a period of time and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities."* Every activity contains an identifier, timestamps for start and end times, and a set of attribute-value pairs. Activities are a part of the PROV core and are graphically represented by means of blue rectangles.

- *"**Generation** is the completion of production of a new entity by an activity. This entity did not exist before generation and becomes available for usage after this generation."* Every generation is represented by a ***wasGeneratedBy*** edge which contains an identifier, an identifier of the generated entity, an identifier of the activity involved in the generation, a timestamp, and a set of attribute-value pairs. At least one of the optional attributes must be present. Generation is a part of the PROV core.
- *"**Usage** is the beginning of utilizing an entity by an activity. Before usage, the activity had not begun to utilize this entity and could not have been affected by the entity."* Every usage is represented by a ***used*** edge which contains an identifier, an identifier of the activity involved in the usage, an identifier of the entity being used, a timestamp, and a set of attribute-value pairs. At least one of the optional attributes must be present. Usage is a part of the PROV core.
- *"**Communication** is the exchange of some unspecified entity by two activities, one activity using some entity generated by the other."* All communication is represented by ***wasInformedBy*** edges which contain an identifier, identifiers of the sending and receiving activities, and a set of attribute-value pairs. Communication is a part of the PROV core.
- *"**Start/End** is when an activity is deemed to have been started/ended by an entity, known as* trigger*. The activity did not exist/no longer exists before its start/end.* (...) *A start/end may refer to a trigger entity that set off/terminated the activity, or to an activity, known as* starter/ender*, that generated the trigger."* Every start/end is represented by a ***wasStartedBy/wasEndedBy*** edge which contains an identifier, an identifier of the started/ended activity, identifiers of the trigger and starter/ender, a timestamp, and a set of attribute-value pairs. At least one of the optional attributes must be present.
- *"**Invalidation** is the start of the destruction, cessation, or expiry of an existing entity by an activity. The entity is no longer available for use (or further invalidation) after invalidation.* (...)*"* Every invalidation is represented by a ***wasInvalidatedBy*** edge which contains an identifier, an identifier of the invalidated entity, an identifier of the invalidator activity, a timestamp and a set of attribute-value pairs. At least one of the optional attributes must be present.

2. **Derivations.**

- *"A **derivation** is a transformation of an entity into another, an update of an entity resulting in a new one, or the construction of a new entity based on a pre-existing entity."* Every derivation is represented by a ***wasDerivedFrom*** edge which contains an identifier, identifiers of the generated and used entities by the derivation, an identifier of the involved activity, identifiers of the generation and the usage which the derivation expresses, and a set of attribute-value pairs. Derivation is a part of the PROV core.

- *"A **revision** is a derivation for which the resulting entity is a revised version of some original."* This relation can be denoted by adding the following attribute-value pair to a derivation: `prov:type = 'prov:Revision'`.
- *"A **quotation** is the repeat of (some or all of) an entity, such as text or image, by someone who may or may not be its original author."* This relation can be denoted by adding the following attribute-value pair to a derivation: `prov:type = 'type:Quotation'`.
- *"A **primary source** for a topic refers to something produced by some agent with direct experience and knowledge about the topic, at the time of the topic's study, without benefit from hindsight.* This relation can be denoted by adding the following attribute-value pair to a derivation: `prov:type = 'type:PrimarySource'`.

3. **Agents, Responsibility, and Influence.**

   - *"An **agent** is something that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent's activity."* Every agent contains an identifier and a set of attribute-value pairs. Three standard types of agents are available (*SoftwareAgent*, *Organization* and *Person*) and they can be defined by adding an attribute-value pair. Agents are a part of the PROV core and are graphically represented by means of orange pentagons.
   - *"**Attribution** is the ascribing of an entity to an agent."* Every attribution is represented by a ***wasAttributedTo*** edge which contains an identifier, identifiers of the involved entity and agent, and a set of attribute-value pairs. Attribution is a part of the PROV core.
   - *"An activity **association** is an assignment of responsibility to an agent for an activity, indicating that the agent had a role in the activity. It further allows for a plan to be specified, which is the plan intended by the agent to achieve some goals in the context of this activity."* Every association is represented by a ***wasAssociatedWith*** edge which contains an identifier, an identifier of the involved activity, identifiers for the agent and plan involved, and a set of attribute-value pairs. At least one of the optional attributes must be present. Association is a part of the PROV core.
   - *"**Delegation** is the assignment of authority and responsibility to an agent (by itself or by another agent) to carry out a specific activity as a delegate or representative, while the agent it acts on behalf of retains some responsibility for the outcome of the delegated work."* Every delegation is represented by a ***actedOnBehalfOf*** edge which contains an identifier, identifiers of the delegate and responsible agents, an identifier of an activity involved, and a set of attribute-value pairs. Delegation is part of the PROV core.
   - *"**Influence** is the capacity of an entity, activity, or agent to have an effect on the character, development, or behavior of another by means of usage, start, end, generation, invalidation, communication, derivation, attribution, association, or delegation."* Every influence

is represented by a ***wasInfluencedBy*** edge which contains an identifier, identifiers for the two objects involved, and a set of attribute-value pairs.

4. **Bundles.**

   - *"A **bundle** is a named set of provenance descriptions, and is itself an entity, so allowing provenance of provenance to be expressed. A **bundle constructor** allows the content and the name of a bundle to be specified; (...)"* Every bundle constructor has an identifier and a set of attribute-value pairs.

5. **Alternate Entities.**

   - *"An entity that is a **specialization** of another shares all aspects of the latter, and additionally presents more specific aspects of the same thing as the latter. In particular, the lifetime of the entity being specialized contains that of any specialization."* Every specialization is represented by a ***specializationOf*** edge which contains identifiers of the specialized and general entity.
   - *"Two **alternate** entities present aspects of the same thing. These aspects may be the same or different, and the alternate entities may or may not overlap in time."* Every alternate relation is represented by an ***alternateOf*** edge which contains identifiers of the two alternate entities.

6. **Collections.**

   - *"A **collection** is an entity that provides a structure to some constituents that must themselves be entities. These constituents are said to be **member** of the collections. (...)"*
   - *"**Membership** is the belonging of an entity to a collection."* Every member is represented by a ***membership*** edge which contains an identifier of the collection and an identifier of the member entity.

Besides these types and relations, PROV also defines several default attributes in the `prov` namespace, to be used in the attribute-value pairs: `prov:label` for object descriptions, `prov:location` to add geospatial metadata, `prov:role` to clarify the function of entities or agents towards activities,[4] `prov:type` for typing objects and `prov:value` to associate values to entities. All of these attributes are optional.

Five kinds of events can be modelled: activity start, activity end, entity generation, entity usage and entity validation. These are all *instantaneous* events in that they cannot be modelled to run longer than one single snapshot in time.

[4]Whereas in OPM roles are mandatory for use, generation and controlling relations, in PROV they are always optional. Furthermore, in PROV they can also be applied to invalidation, start and end relations, and one edge may contain multiple roles. The fact that PROV roles don't have a semantic value doesn't mean that *use-generate-derive triangles* aren't supported. Instead, these triangles can now be specified inside a derivation relation by means of optional placeholders to specify the involved usage, activity and generation.

2.2.2 Inferences and Constraints

Inferences allow the user to add new components (activities, entities or edges) based on existing data. Below are the most significant inferences (i.e., those that exclusively consist of PROV core concepts).[5]

- A *wasInformedBy* edge from an activity A to an activity B implies the existence of an intermediate entity E, a *used* edge from A to E and a *wasGeneratedBy* edge from E to B (inference 5).
- An entity E, a *used* edge from an activity A to an entity E, and a *wasGeneratedBy* edge from E to an activity B implies the existence of a *wasInformedBy* edge from A to B (inference 6).
- The existence of an entity implies the existence of a *wasGeneratedBy* edge and a *wasInvalidatedBy* edge, both connected to that entity (inference 7).
- A *wasDerivedFrom* edge from an entity E to an entity F implies the existence of an intermediate activity A, a *wasGeneratedBy* edge from E to A and a *used* edge from A to F (inference 11). This is a *use-generate-derive triangle*.

PROV instances are *valid* if they represent a consistent history to which logical reasoning can be applied. To determine its validity, an instance must comply to a set of constraints, which can be categorized into four groups. Only those constraints which affect the PROV core concepts are listed. Unless otherwise indicated, temporal inequalities are not strict, i.e., simultaneous events are allowed.

- *Uniqueness constraints*: identifiers guarantee that objects and relations are unique (constraints 22 and 23); it is not possible to have multiple *wasGeneratedBy* edges or *wasInvalidatedBy* edges between the same entity and activity (c24 and c25).
- *Event ordering constraints*:
  - Activities: the start of an activity precedes its end (c30); an activity can be started/ended by multiple other activities but only if this happens simultaneously (c31 and c32); an activity uses/generates entities after it starts but before it ends (c33 and c34); when one activity informs another activity, the start of the target precedes the end of the source (c35).
  - Entities: the generation of an entity precedes its invalidation (c36); an activity is used after its generation but before its invalidation (c37 and c38); an entity can be generated/invalidated by multiple activities but only if this happens simultaneously (c39 and c40); inside a derivation, the usage precedes the generation (c41); when one entity is derived from another entity, the generation of the target entity strictly precedes the generation of the source entity (c42).

[5]Based on the W3C Recommendation of April 30th 2013: `http://www.w3.org/TR/2013/REC-prov-constraints-20130430/`

- *Type constraints* make sure that all parameters are of the correct type, e.g., a *wasGeneratedBy* edge has an entity as source and an activity as target (c50).

- *Impossibility constraints* guarantee that certain specific patterns never occur. A derivation without a specified activity cannot include a generation or usage (c51); it is impossible for two items to have the same identifier (c53 and c54).

To finish this analysis, the author of this thesis presents his personal comments on some aspects of PROV.

- Many of the new relations that are not a part of the PROV core increase the complexity without adding expressibility. More precisely, the relations *revision*, *quotation*, *primary source*, *delegation*, *influence*, *alternate*, *specialization*, and *membership* could be modelled by means of attribute-value pairs instead of distinct relations.

- The generation and invalidation timestamps of entities are stored inside relations (*wasGeneratedBy* and *wasInvalidatedBy*) and not inside the entities themselves.

  Activities differ from this in that their start and end times can be stored inside relations (*wasStartedBy* and *wasEndedBy*), but also inside the activities themselves. This increases the complexity of the model: two constraints (constraint 28, called *unique-startTime*, and constraint 29, called *unique-endTime*) demand that time values in activities match those in the relations. Furthermore, querying time information from activities has to take into account four distinct possibilities of time annotation. Finally, although the PROV core model doesn't include the *wasStartedBy* and *wasEndedBy* relations, they often have to be considered when reasoning with core concepts.

  It would benefit the conciseness and clarity of the model to remove the timestamps from the activities, thereby creating a mechanism of storing time similar to that of entities. This removal would not require any other modifications than abolishing constraints 28 and 29, since no other constraints rely on timestamps stored inside activities.

- PROV constraint 42 (*derivation-generation-generation-ordering*) requires the used entity in every derivation to be created *strictly* before the generated entity. This is based on the definition of derivation, which requires entities derived from other entities to be "newer" than their source. However, this interpretation doesn't anticipate different levels of time granularity: if an entity is derived from another entity within the same time interval, the PROV instance will incorrectly be considered invalid. The strictness of this constraint seems to yield no benefits and is unique in the sense that no other constraints demand strict inequalities.

OPM		PROV core	
○	artifact	entity	○
□	process	activity	□
⬠	agent	agent	⬠
□⟶○	used	used	□⟶○
○⟶□	wasGeneratedBy	wasGeneratedBy	○⟶□
□⟶□	wasTriggeredBy	wasInformedBy	□⟶□
○⟶○	wasDerivedFrom	wasDerivedFrom	○⟶○
□⟶⬠	wasControlledBy	wasAttributedTo	□⟶⬠
		wasAssociatedWith	○⟶⬠
		actedOnBehalfOf	⬠⟶⬠

Figure 2.6: Comparison between objects and relations in OPM and PROV core.

2.3 OPM vs. PROV

Now that OPM and PROV have been investigated, parallels and differences between these two models can be observed.

It is quite obvious that the PROV core has been deeply inspired by OPM. One only has to compare the nature and naming of the objects and relations to discover many striking similarities (see Figure 2.6). The PROV core, however, has two more relations than OPM, both related to agents.

More noteworthy differences exist. OPM roles, for example, don't have a PROV counterpart, although they can be simulated by means of attribute-value pairs. Hence, PROV by default makes no distinction between *precise* or *imprecise* edges. Another difference is that PROV supports the representation of entity removal by means of *wasInvalidatedBy* edges, whereas OPM does not.

A natural question to ask is whether a valid OPM instance is compatible with a valid PROV instance, i.e., how do the axioms in the formal account of OPM relate to the constraints of PROV? By comparing, and sometimes combining, constraints it is possible to create a mapping. Note that for PROV, unless otherwise indicated, temporal inequalities are not strict, i.e., simultaneous events are allowed.

- **OPM AX 1 = PROV c30** (*start-precedes-end*)
  - OPM: For every process: $begin(P) \leq end(P)$.
  - PROV: An activity starts before it ends.
- **OPM AX 2 = PROV c34** (*generation-within-activity*)
  - OPM: For every precise *wasGeneratedBy* edge $A \xrightarrow{!} P$: $begin(P) \leq create(A) \leq end(P)$.
  - PROV: An activity generates entities after it starts but before it ends.
- **OPM AX 3 = PROV c33 + c37** (*usage-within-activity* + *generation-precedes-usage*)
  - OPM: For every precise *used* edge $P \xrightarrow{r} A$: $begin(P) \leq use(P, r, A) \leq end(P)$ and $create(A) \leq use(P, r, A)$.

  - PROV: An activity uses entities after it starts but before it ends; an entity is used after its generation. From this follows that an entity is used after it is generated, after the process begins, and before the process ends.

- **OPM AX 4 ≈ PROV c42** (*derivation-generation-generation-ordering*). There is no complete match because OPM allows for equal creation timestamps whereas PROV doesn't. This means that valid OPM graphs are not compliant with the PROV constraints if they contain *wasDerivedFrom* edges between simultaneously created entities.
  - OPM: For every imprecise *wasDerivedFrom* edge $A \rightarrow B$: $create(B) \leq create(A)$.
  - PROV: The generation of the target entity strictly precedes the generation of the source entity.

- **OPM AX 5 = PROV c34(1)** (*generation-within-activity*). Since PROV does not distinguish between *precise* and *imprecise* edges, this axiom becomes subsumed by OPM AX 2.
  - OPM: For every imprecise *wasGeneratedBy* edge $A \rightarrow P$: $begin(P) \leq create(A)$.
  - PROV: An activity generates entities after it starts.

- **OPM AX 6 = PROV c33(2) + c37** (*usage-within-activity + generation-precedes-usage*). Since PROV does not distinguish between *precise* and *imprecise* edges, this axiom becomes subsumed by OPM AX 3.
  - OPM: For every imprecise *used* edge $P \rightarrow A$: $create(A) \leq end(P)$.
  - PROV: An activity uses entities before it ends; an entity is used after its generation. From this follows that an entity is generated before the process ends.

- **OPM AX 7 = PROV c35** (*wasInformedBy-ordering*)
  - OPM: For every *wasInformedBy* edge $P \rightarrow Q$: $begin(Q) \leq end(P)$.
  - PROV: When one activity informs another activity, the start of the target precedes the end of the source.

- **OPM AX 8** (*triangle axiom*) **= PROV c41** (*derivation-usage-generation-ordering*)
  - OPM: For every *use-generate-derive triangle* (A, B, P, r): $use(P, r, B) \leq create(A)$.
  - PROV: Inside a derivation, the usage relation precedes the generation relation.

Chapter 3

Reasoning Application

A part of this thesis consists of a reasoning application which allows the user to extract additional information from PROV core instances.

3.1 Front-end

The software enables the verification of which axioms apply to a given PROV core instance. The instance is valid if none of the axioms are violated. Furthermore it is possible to inspect whether two temporal variables occur in a temporal inference.

The front-end is made in the programming language *C# 4.0* in *.NET Framework 4* with a *Winforms* user interface (see Figure 3.1). The open-source package *QuickGraph*[1] is used for the internal representation of PROV graphs. This, in combination with the open-source graph visualization software *Graphviz*,[2] allows the rendering of images of graphs (see Figure 3.2), although several bugfixes and method overrides were required for this.

Upon launching the program, the names and descriptions of stored PROV instances can be browsed. It is also possible to import a new instance to the database from an XML file compliant with the PROV XML Schema.[3] Once a specific instance is selected or imported, it is loaded from the database into the program. An internal representation is created and options to work with axioms and variables become visible. The user is given the choice to show an image of the graph. At any time it is possible to switch to another PROV instance.

Validating if a PROV instance complies to axioms is done by selecting one or more axioms and pressing the *Update* button. The log displays for each selected axiom if conflicts were found, with additional information (involved objects, relations) if this is indeed the case. These results are also written to a text file. If the PROV graph is being visually shown, those objects and relations involved in conflicts are instantaneously coloured red.

[1] `http://quickgraph.codeplex.com` [Retrieved June 3, 2013]
[2] `http://www.graphviz.org` [Retrieved June 3, 2013]
[3] `http://www.w3.org/TR/prov-xml` [Retrieved June 3, 2013]

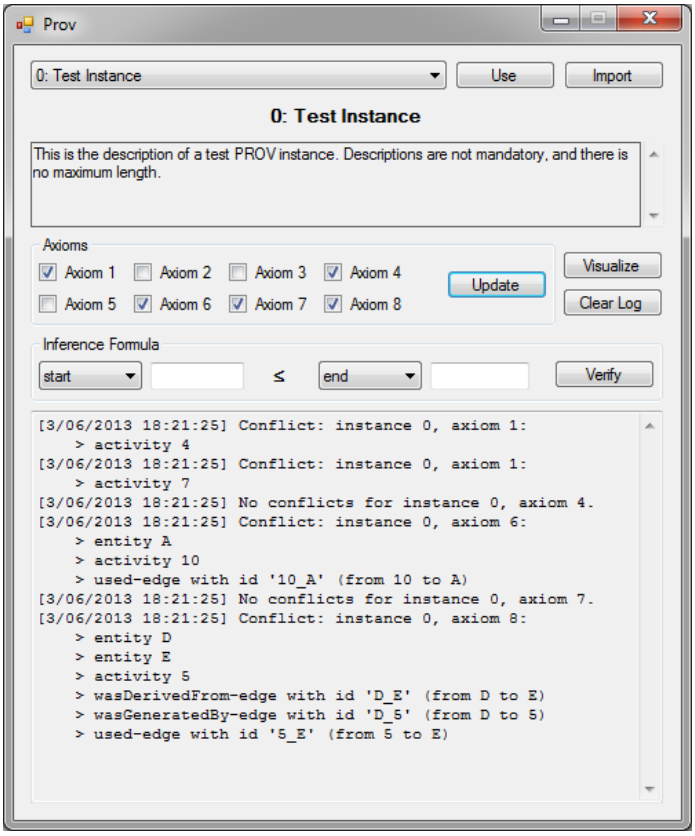

Figure 3.1: User interface of the application. Five axioms are being tested.

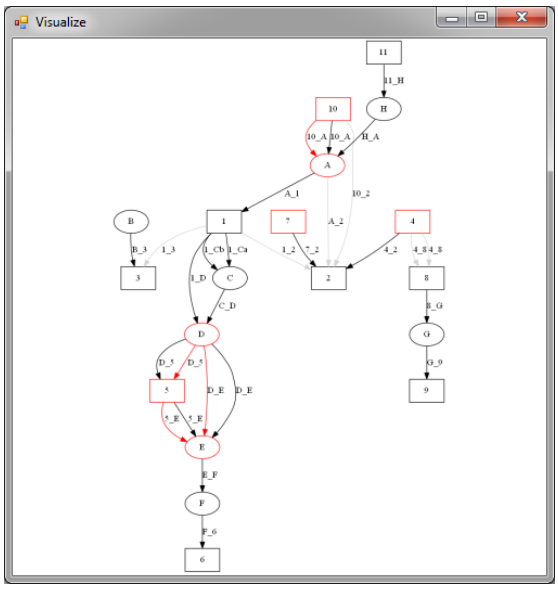

Figure 3.2: Visualization of a PROV instance. Red objects and relations are involved in at least one axiom conflict. The window can be resized and the image adjusts to it.

It is possible to check if an inference exists between two temporal variables. The user can construct a formula with actions and identifiers, e.g., *start 5* $\leq$ *end 6*. Every possible formula is guaranteed to match with at least one of the inference rules. In this example, if an inference can be made between the start of activity *5* and the end of activity *6*, the log mentions this, along with the path taken.

3.2 Back-end

Whereas the front-end takes care of the graphical elements (user interface, visualization), the back-end takes care of the storage and calculation of provenance information. The database management system used is *Microsoft SQL Server 2012*.

The database has one table for every kind of PROV core object or relation, along with the *wasStartedBy* and *wasEndedBy* relations (see Figure 3.3). Every parameter from the data model can be stored, which includes attributes even though the program never needs them. An additional table, which links to every other table, holds the names and descriptions from the PROV instances. Note that this method of differentiating between multiple instances is program-specific, i.e., it is not compliant with the PROV standard.

Information about axioms and inferences can be retrieved with database views. For axioms, some views create patterns while other views retrieve conflicts from those patterns. For inferences, some views create patterns while others verify if they hold. To illustrate how exactly axioms are verified, the views for axiom 2 (= PROV constraint 34) are presented. Other axioms are verified in a similar fashion. Inferences are easier to detect since they completely disregard temporal information.

Recall the PROV equivalent of axiom 2: *"An activity generates entities after it starts but before it ends."* The pattern needs to include those objects relevant to the axiom, which in this case are *wasGeneratedBy* edges and the entities and activities connected to those edges. The SQL code for this pattern, a view named *pattern_ax2*, is as follows:

```
SELECT Entity.instance_id,
       Entity.id AS Entity_id,
       Activity.id AS Activity_id,
       Activity.startTime AS Activity_startTime,
       Activity.endTime AS Activity_endTime,
       wasGeneratedBy.id AS wasGeneratedBy_id,
       wasGeneratedBy.time AS wasGeneratedBy_time
FROM Entity, Activity, wasGeneratedBy
WHERE Entity.instance_id = Activity.instance_id
  AND Entity.instance_id = wasGeneratedBy.instance_id
  AND wasGeneratedBy.activity_id = Activity.id
  AND wasGeneratedBy.entity_id = Entity.id
```

Note that every object in the pattern needs to have the same *instance_id* to make sure different PROV instances remain separated. This view actually retrieves

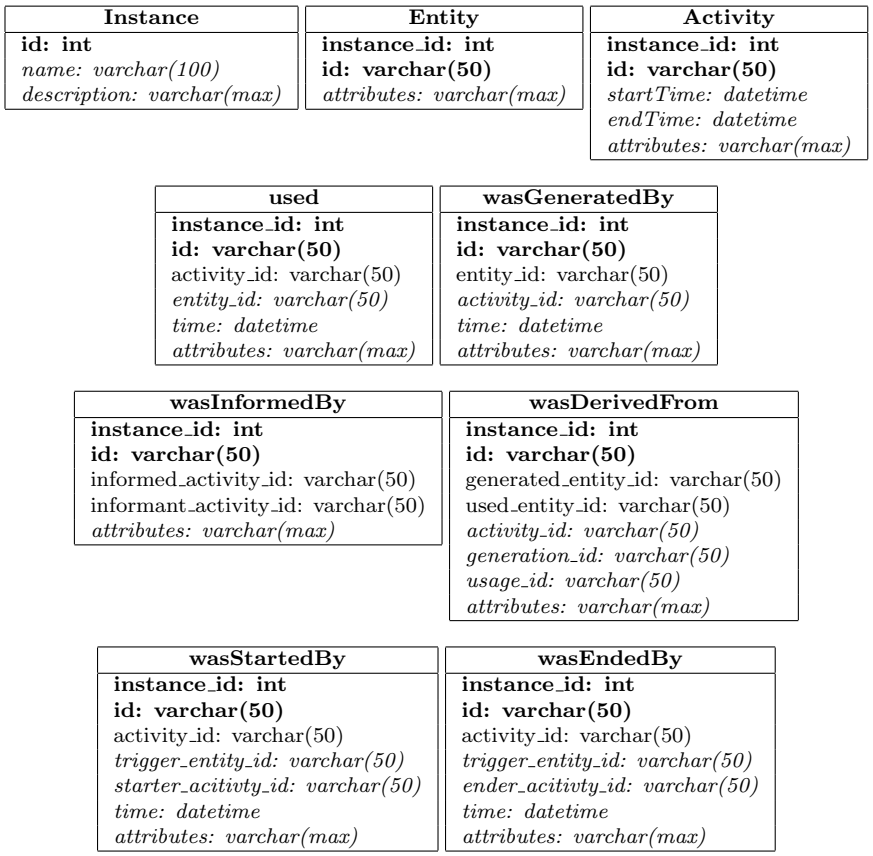

Figure 3.3: Database tables. Primary keys are in bold and fields which allow null values are in italics.

more information than is required in the next step. This is because the front-end program uses this additional knowledge to include interesting information in the log. The result from *pattern_ax2* is now used to detect conflicts against axiom 2. The SQL code to find those conflicts, a view named *invalid_ax2*, is as follows:

```
SELECT DISTINCT p_ax2.*
FROM pattern_ax2 p_ax2, invalid_ax1 i_ax1
WHERE (p_ax2.instance_id = i_ax1.instance_id
       AND p_ax2.Activity_id = i_ax1.id)
   OR p_ax2.wasGeneratedBy_time < p_ax2.Activity_startTime
   OR p_ax2.wasGeneratedBy_time > p_ax2.Activity_endTime
```

A pattern does not comply to axiom 2 if it contains an invalid activity (i.e., an activity that ended before it started), or if the generation happened before the activity started, or if the generation happened after the activity ended. The view *invalid_ax2* shows exactly those patterns which fulfil at least one of these conditions.

Appendix A

Summary in Dutch

A.1 Wat is Provenance?

Het woord *provenance* komt van het Middelfranse woord *provenir*, dat men als "voortkomst, ontstaan" kan interpreteren. De moderne betekenis is "(1) oorsprong, bron; (2) de eigendomsgeschiedenis van een waardevol object of kunstwerk, of literatuur". De betekenis van dit woord is eigenlijk nog veel ruimer en overspant vele vakgebieden. Het tweede gedeelte van de moderne betekenis is van kracht in de wereld van kunst en antiek. In de archiefwetenschappen doelt men eerder op een fundamenteel principe in de kunst van het archiveren, waarbij een logisch geheel van objecten niet met andere objecten vermengd wordt. Sommige archeologen associëren het begrip provenance met de geografische oorsprong van een artifact, maar veel andere interpretaties zijn in omloop. Geologen, ten slotte, verstaan onder provenance alle eigenschappen van een bepaald gebied waaruit sedimenten en stenen ontgonnen werden.

A.1.1 Provenance in Databanken

De meest recente invulling van het begrip provenance vond in de wereld van databanken plaats. Hier tracht men de vraag te beantwoorden waar data precies vandaan komt, welke veranderingen het onderging, en welke processen voor die veranderingen verantwoordelijk waren. Vaak gebruikte synoniemen zijn *lineage* en *pedigree*, hoewel de eerste bijna uitsluitend in de context van *why*-provenance (zie verder) gehanteerd wordt. De eerste publicatie rond dit onderwerp dateert van 1986, maar het onderzoek kwam pas in de vroege jaren 2000 goed op gang. Een eerste workshop vond in 2002 plaats en tegenwoordig is provenance een populair onderzoeksgebied dat op veel interesse mag rekenen.

Merk op dat in deze thesis het begrip *databank* ruimer opvat dat de traditionele gesloten systemen: het omvat alle manieren van dataopslag, of die nu open of gesloten zijn, gestructureerd of niet. Zo kan men ook het World Wide Web als één grote databank aanschouwen.

Welke soorten van provenance bestaan er nu concreet? Een eerste onderscheid kan men tussen *where*-, *why*-, en *how*-provenance maken. Bij de eerste

soort zoekt men waar bepaalde data vandaan komt, en dit is nuttig voor het opsporen van de bron van fouten. Bij *why*-provenance vraagt men zich af waarom een stukje data in de databank aanwezig is. Bij *how*-provenance, ten slotte, kijkt men na welke tuples in de databank tot een bepaald resultaat contribueerden.

Afhankelijk van welke informatie interessant lijkt kan men ook enkele perspectieven in acht nemen. By *procesgebaseerde* provenance focust men zich niet zozeer op de data maar wel op de processen die de data vorm gaven. Dit kan erg veel aspecten omvatten, zoals bijvoorbeeld uitgevoerde software, betrokken gebruikers en gebruikte elektriciteit om het systeem draaiende te houden. Andere perspectieven leggen de focus op de betrokken *agents* (entiteiten die data creëren of bewerken) of *objects* (artifacten die bijdroegen tot gecombineerde artifacten).

Een ander mechanisme voor provenance is opslag in annotaties. In sommige ontologiën, waaronder Dublin Core, kan provenance-gerelateerde informatie (zoals auteur, aanmaakdatum, versie) bewaard worden.

Een logische vraag is waar provenance nu eigenlijk voor kan dienen. Talrijke voorbeelden werden door de *W3C Provenance Incubator Group* verzameld en vervolgens tot drie voorbeeldscenario's samengevoegd. Deze illustreren enkele van de vele toepassingen die provenance te bieden heeft.

- Een nieuwsblog die nieuwsberichten uit talrijke verschillende bronnen verzamelt: licenties controleren, ongestructureerde gegevens zoals documenten en media integreren, het gebruik van de data op de voet volgen, de autoriteit van bronnen controleren, ...
- Analyse van een ziekte-uitbraak: data van verschillende bronnen (met mogelijk verschillende representaties) samenvoegen, data archiveren en hergebruiken, beslissingen die op data gebaseerd zijn staven, bevindingen kunnen reproduceren, ...
- Een zakelijk contract: realisaties tegenover verplichtingen vergelijken, afwijkingen in de doelstellingen begrijpen, bevoegdheden verifiëren, ...

Deze voorbeelden zijn erg concreet maar er zijn ook enkele globale voordelen. Één van de belangrijkste voordelen is de mogelijkheid om de betrouwbaarheid van data te evalueren. Hoewel *vertrouwen* een erg meerduidig begrip is, draait het erg vaak rond de betrouwbaarheid van de informatiebron. Kennis over die bron en de daaropvolgende bewerkingen stelt gebruikers in staat om data beter in zijn context te plaatsen, en van daaruit een beter gefundeerd oordeel te vellen.

Gerelateerd aan de betrouwbaarheid van data is de evaluatie van de *kwaliteit*. Ook dit is een bijzonder veelzijdig begrip, maar dat goede maatstaven om kwaliteit te meten uiterst belangrijk zijn staat buiten kijf. Provenance werd als een potentiële kwaliteitsparameter gesuggereerd. Het zou gebruikers in staat stellen om fouten in data op te sporen door te onderzoeken waar precies die fouten vandaan kwamen en hoe ze verder in verschillende datasets propageerden. Vooral databanken in het publieke domein kunnen hier voordeel uit halen. In biologische databanken, bijvoorbeeld, kunnen foutieve gegevens de oorzaak van een aanzienlijke economische en medische schade zijn.

Een andere toepassing van provenance is het verduidelijken van *causaliteit*, hetgeen bepaalt hoe tuples in een databank tot een bepaald zoekresultaat bijdragen. Met die kennis kan men onder meer onverwachte resultaten verklaren.

Erg gerelateerd met causaliteit is de vraag waarom een bepaalde tuple *niet* in een resultaat te vinden is, het zogenaamde *Why Not?*-provenance. Er bestaat al een framework dat automatisch antwoorden op deze vraag verschaft.

Het opslaan van provenance-gegevens heeft niet enkel voordelen. Zo zijn er onder meer bekommeringen met betrekking tot de vereiste opslagruimte, veiligheid en privacy.

A.2 OPM en PROV

Er bestaan verschillende modellen voor de opslag van provenance gegevens, maar de meest populaire zijn *OPM* en *PROV*. Hier volgt een analyse van beide modellen.

A.2.1 OPM

OPM staat voor *Open Provenance Model* en het is het resultaat van een reeks publieke challenges en workshops. Het tracht onder meer om een standaardmodel te verschaffen ter uitwisseling van provenance informatie tussen verschillende systemen. Daarbij is het belangrijk om provenance nauwkeurig en technologie-neutraal te definiëren. Er zijn geen richtlijnen over hoe provenance intern in systemen opgeslagen dient te worden. Bovendien worden er geen specifieke digitale representaties ter uitwisseling aanbevolen, hoewel er reeds implementaties van OPM in onder meer XML bestaan.

Een OPM instantie kan als een gerichte graaf aanzien worden die oorzakelijke verbanden (bogen) tussen verschillende "dingen" (knopen) toont. Er zijn drie types van "dingen":

- *Artifacts* (aangeduid met ellipsen): onveranderlijke momentopnamen van objecten. Deze objecten kunnen zowel fysiek als digitaal zijn.
- *Processes* (aangeduid met rechthoeken): een actie of een reeks van acties, toegepast op of veroorzaakt door artifacts. Deze resulteren in nieuwe artifacts.
- *Agents* (aangeduid met achthoeken): entiteiten die als een aanvoerder van een proces werken. Agents starten, vergemakkelijken, controleren of beïnvloeden de werking van een process.

Merk op dat zowel digitale als reële zaken gemodelleerd kunnen worden. Er bestaan vijf types van oorzakelijke verbanden:

- Een *used* boog van een process P naar een artifact A stelt een oorzakelijk verband voor waarin P de beschikbaarheid van A nodig had.
- Een *wasGeneratedBy* boog van een artifact A naar een process P stelt een oorzakelijk verband voor waarin A de beschikbaarheid van P nodig had. Hieruit volgt dat A gegenereerd werd nadat P gestart werd.
- Een *wasTriggeredBy* boog tussen twee processen stelt een oorzakelijk verband voor waarin een process P_2 de beschikbaarheid van een ander process P_1 nodig had. Hieruit volgt dat P_1 begon vooraleer P_2 kon stoppen.

- Een *wasDerivedFrom* boog van een artifact A_2 naar een artifact A_1 stelt een oorzakelijk verband voor waarbij A_2 de beschikbaarheid van A_1 nodig had. Hieruit volgt dat A_1 voor A_2 gegenereerd werd.

- Een *wasControlledBy* boog tussen een process P en een agent Ag stelt een oorzakelijk verband voor waarbij Ag de beschikbaarheid van P nodig had.

Een *wasGeneratedBy* boog is eigenlijk een afkorting van een *used* boog onmiddellijk gevolgd door een *wasGeneratedBy* boog (hetgeen een processgebaseerd perspectief mogelijk maakt), en mag erdoor vervangen worden. De omgekeerde richting is eveneens geldig.

Een *wasDerivedFrom* boog is eigenlijk een afkorting van een *wasGeneratedBy* boog onmiddellijk gevolgd door een *used* boog (hetgeen een artifactgebaseerd perspectief mogelijk maakt), en mag erdoor vervangen worden. De omgekeerde richting geldt echter niet, omdat de rol van de process in het oorzakelijk verband tussen beide artifacts niet aangetoond kan worden.

Multi-step inferences geven indirecte causale verbanden weer, bijvoorbeeld: een multi-step *used* boog $p \rightarrow^* a$ drukt uit dat een process p gebruik maakte van een artifact a of een afgeleide ervan. Enkel voor *wasControlledBy* kunnen dergelijke afleidingen niet voorkomen.

De relaties *used*, *wasGeneratedBy* en *wasControlledBy* krijgen verplicht *roles* toegekend die helpen om meerdere bogen tussen dezelfde knopen te onderscheiden.

OPM maakt het mogelijk om de volgende tijdsinformatie op te slaan: het tijdstip waarop een artifact aangemaakt of gebruikt wordt, het tijdstip waarop een process gestart of gestopt wordt, en voor iedere relatie het tijdstip waarop ze voorkomt (bij *wasControlledBy* is er zowel een start als einde). Merk op dat men geen oorzakelijke verbanden kan maken op basis van deze informatie, aangezien ook onmogelijke scenario's gemodelleerd kunnen worden.

Met behulp van beschrijvingen (*accounts*) kunnen meerdere instanties, met verschillende granulariteiten, elkaar overlappen. Bijkomende informatie (subtypes, beschrijvingen, ...) kan door het *annotation framework* toegevoegd worden. *Profiles* laten virtuele collecties van OPM instanties toe, waarop men gemakkelijk specifieke richtlijnen kan toepassen.

Het gebrek aan een formele semantiek in OPM werd aangekaart met een geformaliseerde notie. Een edge is *precise* wanneer die een role bevat, anders is die *imprecise*. Een OPM graaf is *legal* wanneer iedere artifact met hoogstens één precise *wasGeneratedBy* boog in verbinding staat, en wanneer er voor elke precise *wasDerivedFrom* boog $A \rightarrow B$ een process P bestaat zodat $A \xrightarrow{!} P$ en $A \xrightarrow{r} P$. Merk op dat de role van de derivation en de use bij deze gelijk zijn. Deze constructie noemt een *use-generate-derive triangle* en het toont aan dat artifact B een rechtstreekse invloed had op artifact A.

De starttijd van een proces, de eindtijd van een proces, het tijdstip waarop een artifact gecreëerd werd en het tijdstip waarop een artifact gebruikt werd worden nu aangeduid door respectievelijk $begin(P)$, $end(P)$, $create(A)$, $use(P, r, A)$, waarbij A een artifact is, P een process en r een role. Ze worden *temporal variables* genoemd. De formele notie introduceert ook het concept van *temporal interpretation*: een triple $(T, \leq, \tau)$ waarbij T de set alle tijdspunten in de graaf is, $\leq$ de *partial order* van T, en τ een mapping van alle temporele variabelen in

T (het bevat elke geïmpliceerde time constraint). Een *inequality* is een vergelijking van de vorm $u \leq v$, waarbij u en v temporele variabelen zijn. Wanneer een inequality in elk temporeel model van een instance geldt, noemt men het een *logical consequence*.

Om te evalueren of alle temporele informatie in een instance zinvol is dient het te voldoen aan de *temporal theory* van die instance: een verzameling van acht inequalities (axioma's) die steeds van toepassing zijn. Deze inequalities zijn als volgt:

- AX 1: Voor elk process: $begin(P) \leq end(P)$.
- AX 2: Voor elke precise *wasGeneratedBy* boog $A \xrightarrow{!} P$: $begin(P) \leq create(A) \leq end(P)$.
- AX 3: Voor elke precise *used* boog $P \xrightarrow{r} A$: $begin(P) \leq use(P, r, A) \leq end(P)$ en $create(A) \leq use(P, r, A)$.
- AX 4: Voor elke imprecise *wasDerivedFrom* boog $A \rightarrow B$: $create(B) \leq create(A)$.
- AX 5: Voor elke imprecise *wasGeneratedBy* boog $A \rightarrow P$: $begin(P) \leq create(A)$.
- AX 6: Voor elke imprecise *used* boog $P \rightarrow A$: $create(A) \leq end(P)$.
- AX 7: Voor elke *wasInformedBy* boog $P \rightarrow Q$: $begin(Q) \leq end(P)$.
- AX 8 (*triangle axioma*): Voor elke *use-generate-derive triangle* (A, B, P, r): $use(P, r, B) \leq create(A)$.

Indien elke inequality in een temporal interpretation met deze regels overeen stemt, dan is die interpretatie een *temporal model* van die specifieke instantie: alle temporele informatie is zinvol.

Een temporal theory bestaat uit inequalities die op directe (*single-step*) bogen van toepassing zijn, maar het is mogelijk om er nieuwe inequalities uit af te leiden. Beschouw bijvoorbeeld een imprecise *used* boog $P \rightarrow A$ en een imprecise *wasGeneratedBy* boog $A \rightarrow Q$. Uit de axioma's 6 ($create(A) \leq end(P)$) en 5 ($begin(Q) \leq create(A)$) volgt dan dat $begin(Q) \leq end(P)$. Dergelijke inequality kan men door middel van het herhaaldelijk toepassen van transitiviteit bekomen, maar een veel effectievere manier is om eenvoudigweg naar patronen in de OPM graaf te kijken. Inderdaad, een vast aantal *inference rules* (die patronen in de graaf voorstellen) maakt het mogelijk om grafisch af te leiden of een inequality tot de temporal theory behoort. Deze patronen berusten op *edge-inference rules*, deze zijn (A, B, en C zijn entities, P en Q zijn activities):

- Bestaande *wasDerivedFrom*, *wasGeneratedBy*, *used* en *wasInformedBy* bogen zijn ook afgeleide bogen.
- $A \dashrightarrow B$ en $B \dashrightarrow C$ leidt $A \dashrightarrow C$ af (*wasDerivedFrom*).
- $A \dashrightarrow B$ en ($B \rightarrow P$ of $B \xrightarrow{!} P$) leidt $A \dashrightarrow P$ af (*wasGeneratedBy*).
- $A \dashrightarrow B$ en ($P \rightarrow A$ of $P \xrightarrow{!} A$ of $A \xrightarrow{!} P$) leidt $P \dashrightarrow B$ af (*used*).

- $A \dashrightarrow Q$ en ($P \dashrightarrow A$ of $A \xrightarrow{!} P$) leidt $P \dashrightarrow Q$ af (*wasInformedBy*).

Met deze edge-inference rules is het mogelijk om de patronen voor temporal inferences te definiëren. Formeel gesteld is een inequality $u \leq v$ over de temporele variabelen van een OPM graaf, waarbij $u \neq v$, een logical consequence indien het reeds in een axioma vervat is of wanneer het aan één van de volgende inequalities voldoet:

1. $create(B) \leq create(A)$, voor een inferred *wasDerivedFrom* boog $A \dashrightarrow B$.
   AX 4 is hierin vervat.
2. $begin(P) \leq create(A)$, voor een inferred *wasGeneratedBy* boog $A \dashrightarrow P$.
   AX 5 is hierin vervat.
3. $create(A) \leq end(P)$, voor een inferred *used* boog $P \dashrightarrow A$.
   AX 6 is hierin vervat.
4. $begin(Q) \leq end(P)$, voor een inferred *wasTriggeredBy* boog $P \dashrightarrow Q$.
   AX 7 is hierin vervat.
5. $create(B) \leq use(P, r, A)$, voor een *used* boog $P \xrightarrow{r} A$ en een inferred *wasDerivedFrom* boog $A \dashrightarrow B$.
6. $begin(Q) \leq use(P, r, A)$, voor een *used* boog $P \xrightarrow{r} A$ en een inferred *wasGeneratedBy* boog $A \dashrightarrow Q$.
7. $use(P, r, C) \leq create(A)$, voor een *use-derive-generate triangle* (B, C, P, r) en een inferred *wasDerivedFrom* boog $A \dashrightarrow B$.
8. $use(P, r, B) \leq end(Q)$, voor een *use-derive-generate triangle* (A, B, P, r) en een inferred *used* boog $Q \dashrightarrow A$.
9. $use(P, r, B) \leq use(Q, s, A)$, voor een *use-derive-generate triangle* (C, B, P, r) en een *used* boog $Q \xrightarrow{s} A$, waarbij (i) $A = C$ or (ii) $A \dashrightarrow C$.

Deze regels zijn zowel betrouwbaar (ze stellen geldige inferences voor) als volledig (elke inequality die een logical consequence van de axioma's is, maar zelf geen axioma is, kan door een patroon afgeleid worden). Axioma's 4 tot 7 hoeven niet langer overwogen te worden bij het afleiden van temporal inferences, omdat ze in deze lijst vervat zijn.

A.2.2 PROV

PROV is een World Wide Web Consortium (W3C) specificatie voor de opslag van provenance data. Zoals bij OPM kan een instantie als een graaf voorgesteld worden. Er zijn zowel *core structures* (die de essentie van provenance opslaan), als *extended structures* (die de uitdrukbaarheid verhogen). De core objecten zijn als volgt:

- Een *entity* is een fysiek, digitaal, conceptueel of ander soort ding, met enkele vaste aspecten. Entities kunnen zowel echt als denkbeeldig zijn. Ze worden grafisch voorgesteld door middel van gele ovalen.

- Een *activity* is iets dat over een bepaalde tijdsperiode gebeurt, en met entities interageert. Mogelijke bewerkingen: gebruiken, verwerken, transformeren, wijzigen, verplaatsen, genereren, etc. Activities worden door middel van blauwe rechthoeken voorgesteld.
- Een *agent* is iets dat een bepaalde vorm van verantwoordelijkheid voor een activity heeft, of voor het bestaan van een entity, of voor de werking andere agents. Ze worden door middel van oranje vijfhoeken voorgesteld.

PROV core relations zijn:

- Generation: Een *wasGeneratedBy* boog stelt de productie van een nieuwe entity door een activity voor. De entiteit bestond eerder nog niet en kan nu door activities gebruikt worden.
- Usage: Een *used* boog stelt het gebruik van een entity door een activity voor.
- Communication: Een *wasInformedBy* boog stelt de uitwisseling van een niet-gespecificeerde entity door twee activities voor. Een activity gebruikte een entity die door de andere activity gemaakt werd. Deze relatie impliceert het bestaan van een tussenliggende entity.
- Derivation: Een *wasDerivedFrom* boog stelt de transformatie van een entity naar een andere entity voor, of de aanmaak van een nieuwe entity gebaseerd op een andere entity. Deze relatie impliceert het bestaan van een tussenliggende activity.
- Attribution: Een *wasAttributedTo* boog koppelt een entity aan een agent.
- Association: Een *wasAssociatedWith* boog koppelt de verantwoordelijkheid voor een activity aan een agent.
- Delegation: Een *actedOnBehalfOf* boog koppelt de verantwoordelijkheid om een specifieke activity uit te voeren aan een andere agent.

Voorbeelden van extended relations zijn *start*, *end*, *wasInvalidatedBy* en *wasInfluencedBy*. Vijf soorten gebeurtenissen kunnen gemodelleerd worden: het starten van een activity, het stoppen van een activity, het gebruik van een entity, en de invalidatie van een entity. Daarnaast bestaan er enkele optionele standaardattributen in de `prov` namespace voor de opslag van bijkomende informatie, waaronder `prov:label` voor objectbeschrijvingen en `prov:location` voor geospatiale gegevens. Het attribuut `prov:role` dient ter bepaling van de functie van entities of agents bij activities. In tegenstelling tot OPM zijn roles niet verplicht in PROV. Een *use-generate-derive triangle* kan rechtstreeks in de *wasDerivedFrom* relatie bepaald worden door middel van optionele placeholders voor de betrokken activity, *used* boog, en *wasGeneratedBy* boog.

De PROV specificatie bevat een ruim gamma aan inferences en constraints. Inferences maken het mogelijk om nieuwe componenten (objecten of relaties) toe te voegen op basis van bestaande data. De belangrijkste (degene met betrekking tot de PROV core) zijn als volgt:

- Een *wasInformedBy* boog tussen een activity A en een activity B impliceert het bestaan van een tussenliggende entity E, een *used* boog van A naar E en een *wasGeneratedBy* boog van E naar B (inference 5).

- Een entity E, een *used* boog van een activity A naar een entity E en een *wasGeneratedBy* boog van E naar een activity B impliceert het bestaan van een *wasInformedBy* boog van A naar B (inference 6).
- Het bestaan van een entity impliceert het bestaan van een *wasGeneratedBy* boog en een *wasInvalidatedBy* boog, beiden in relatie met die entity (inference 7).
- Een *wasDerivedFrom* boog van een entity E naar een entity F impliceert het bestaan van een tussenliggende activity A, een *wasGeneratedBy* boog van E naar A en een *used* boog van A naar F (inference 11). Dit is een *use-generate-derive triangle*.

PROV instances zijn *valid* indien ze een consistente geschiedenis hebben waarop logisch redeneren mogelijk is. Dit kan door middel van constraints geverifieerd worden. Er zijn vier soorten constraints, degene met betrekking tot de PROV core zijn hier opgesomd. Tenzij anders vermeld zijn temporal inequalities niet strikt.

- *Uniqueness constraints*: identifiers garanderen dat types en relaties uniek zijn (constraints 22 en 23); het is onmogelijk om meerdere *wasGeneratedBy* bogen of *wasInvalidatedBy* bogen tussen dezelfde entity en activity te hebben (c24 en c25).
- *Event ordering constraints*:
  - Activities: de start van een activity komt voor het einde (c30); een activity kan door meerdere andere activities gestart of gestopt worden, maar enkel als dit simultaan gebeurt (c31 en c32); een activity gebruikt of genereert entities na die start maar voor die eindigt (c33 en c34); wanneer een activity een andere activity informeert komt de start van de target voor het einde van de source (c35).
  - Entities: de generatie van een entity komt voor de invalidatie (c36); een activity wordt gebruikt na zijn generatie maar voor zijn invalidatie (c37 en c38); een entity kan door meerdere activities aangemaakt of verwijderd worden, maar enkel als dit simultaan gebeurt (c39 en c40); binnen een derivation komt de usage voor de generation (c41); wanneer een entity van een andere entity afgeleid is, komt de generatie van de target entity strikt voor de generatie van de source entity (c42).
- *Type constraints* zorgen ervoor dat alle parameters van het correcte type zijn, bijvoorbeeld, een *wasGeneratedBy* boog heeft een entity als source en een activity als target.
- *Impossibility constraints* zorgen ervoor dat bepaalde specifieke patronen nooit voorkomen. Een derivation zonder gespecificeerde activity kan geen generation of usage relatie bevatten (c51); het is onmogelijk dat twee items dezelfde identifier hebben (c53 en c54).

Ter afronding van deze analyse presenteert de auteur enkele opmerkingen over bepaalde aspecten van PROV.

OPM		PROV core	
○	artifact	entity	○
□	process	activity	□
⬠	agent	agent	⬠
□⟶○	used	used	□⟶○
○⟶□	wasGeneratedBy	wasGeneratedBy	○⟶□
□⟶□	wasTriggeredBy	wasInformedBy	□⟶□
○⟶○	wasDerivedFrom	wasDerivedFrom	○⟶○
□⟶⬠	wasControlledBy	wasAttributedTo	□⟶⬠
		wasAssociatedWith	○⟶⬠
		actedOnBehalfOf	⬠⟶⬠

Figure A.1: Vergelijking tussen objecten en relaties in OPM en PROV core.

- Sommige extended relations (*revision*, *quotation*, *primary source*, *delegation*, ...) verhogen de complexiteit maar niet de uitdrukkingskracht. Men zou ze even goed met behulp van bijkomende informatie kunnen modelleren in de core.
- Activities kunnen tijdsinformatie bevatten (start, end) die ook in relaties (*wasStartedBy*, *wasEndedBy*) voorkomt. Dit verhoogt de complexiteit: er zijn constraints om consistentie te bekomen, het is ingewikkelder om tijdsinformatie op te vragen, het is soms noodzakelijk om de start en end relaties te betrekken om de core te valideren, etc.
- PROV constraint 42 (*derivation-generation-generation-ordering*) eist dat in elke *wasDerivedFrom* relatie de source entity *strikt* voor de target entity gecreëerd werd. Deze striktheid schijnt geen rekening te houden met de verschillende niveaus van granulariteit: indien een entity binnen hetzelfde tijdsinterval van een andere entity afgeleid wordt zal de PROV instance ten onrechte als *invalid* beschouwd worden.

A.2.3 OPM vs. PROV

Het is vrij duidelijk dat OPM als basis voor de ontwikkeling van PROV diende (zie Figuur. A.1). De PROV core bevat wel twee relaties die OPM niet heeft, beide gerelateerd met agents. Een ander verschil is dat PROV standaard geen onderscheid maakt tussen strikte en niet-strikte bogen, al is er wel de mogelijkheid om rollen toe te kennen. Nog een onderscheid is dat PROV het invalideren van entities ondersteunt, terwijl dit bij OPM niet het geval is.

Men kan zich de vraag stellen of een OPM instance compatibel is met een PROV instance. Met andere woorden, hoe verhouden de axioma's in OPM zich tegenover de constraints van PROV? Door vergelijkingen te maken was het mogelijk om een mapping te bekomen:

- OPM AX 1 = PROV c30 (*start-precedes-end*)
- OPM AX 2 = PROV c34 (*generation-within-activity*)
- OPM AX 3 = PROV c33 + c37 (*usage-within-activity* + *generation-precedes-usage*)

- OPM AX 4 $\approx$ PROV c42 (*derivation-generation-generation-ordering*). Het verschil ligt in de striktheid van de constraint: deze is "$\leq$" bij OPM, maar "$<$" bij PROV.
- OPM AX 5 = PROV c34(1) (*generation-within-activity*)
- OPM AX 6 = PROV c33(2) + c37 (*usage-within-activity* + *generation-precedes-usage*)
- OPM AX 7 = PROV c35 (*wasInformedBy-ordering*)
- OPM AX 8 (*triangle axioma*) = PROV c41 (*derivation-usage-generation-ordering*)

Aangezien PROV geen fundamenteel onderscheid tussen precise en imprecise bogen maakt, worden in de context van PROV respectievelijk OPM axioma's 5 en 6 vervat in axioma's 2 en 3.

A.3 Redeneertool

Een onderdeel van de thesis bestaat uit een redeneertool die de gebruiker in staat stelt om bijkomende informatie over PROV core instanties te verkrijgen.

A.3.1 Front-end

De front-end van het programma werd in de programmeertaal *C# 4.0* in het *.NET Framework* gemaakt, met een *Winforms* user interface. De open-source packages *QuickGraph* en *Graphviz* maken interne en visuele PROV grafen mogelijk.

De gebruiker kan de namen en beschrijvingen van alle PROV core instanties in de database overlopen. Het importeren vanuit een XML dat aan het PROV XML Schema voldoet is eveneens mogelijk. Van zodra een specifieke instantie geselecteerd is, wordt die vanuit de database in het programma geladen en ontstaat er een interne representatie van de graaf. Op elk ogenblik kan de gebruiker naar een andere PROV instantie overschakelen. Bovendien is er de mogelijkheid om een afbeelding van de graaf weer te geven.

Een PROV core instance is *valid* wanneer het aan alle axioma's voldoet. Men kan echter ook specifieke axioma's controleren, en de resultaten voor elk geverifieerd axioma verschijnen simultaan in de log en in een tekstbestand. Op de afbeelding worden alle objecten en relaties die met een axioma in conflict zijn in het rood gekleurd.

Het programma staat ook toe te controleren of er een afleiding bestaat tussen twee temporele variabelen. De gebruiker kan een formule opstellen die bestaat uit acties en identifiers, zoals *start 5* $\leq$ *end 6*. Indien er in dit voorbeeld een afleiding gemaakt kan worden tussen de start van activiteit *5* en het einde van activiteit *6*, wordt dit samen met het genomen pad in de log vermeld.

A.3.2 Back-end

De database staat in voor het opslaan en berekenen van provenance informatie. Het databasebeheersysteem van dienst is *Microsoft SQL Server 2012*.

Voor elke relatie en elk object van de PROV core, samen met de *wasStartedBy* en *wasEndedBy* relaties, werd er een overeenkomstige tabel in de database aangemaakt. Een bijkomstige tabel bevat namen en beschrijvingen voor instanties en is gekoppeld aan alle elke andere tabel. Merk op dat dit een programma-specifieke aanpak is die niet in de PROV standaard vervat zit.

Uitleg over axioma's en afleidingen worden door views in de database verkregen. Sommige views creëren patronen terwijl andere views op basis van die patronen een verificatie doen of een afleiding zoeken.

Bibliography

Artz, D. and Gil, Y. (2007). A survey of trust in computer science and the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58–71.

Bernabei, M. and Bontadi, J. (2011). Determining the resonance wood provenance of stringed instruments from the Cherubini Conservatory Collection in Florence, Italy. *Journal of Cultural Heritage*, 12(2):196–204.

Braun, U. and Shinnar, A. (2006). A Security Model for Provenance. Technical Report 04-06, Harvard University.

Braun, U., Shinnar, A., and Seltzer, M. (2008). Securing Provenance. In Provos, N., editor, *3rd USENIX Workshop on Hot Topics in Security, July 29, 2008, San Jose, CA, USA, Proceedings*, HOTSEC'08, pages 4:1–4:5. USENIX Association.

Buneman, P., Cheney, J., and Vansummeren, S. (2007). On the Expressiveness of Implicit Provenance in Query and Update Languages. In Schwentick, T. and Suciu, D., editors, *Database Theory - ICDT 2007, 11th International Conference, Barcelona, Spain, January 10-12, 2007, Proceedings*, volume 4353 of *Lecture Notes in Computer Science*, pages 209–223. Springer.

Buneman, P., Khanna, S., and Wang-Chiew, T. (2001). Why and Where: A Characterization of Data Provenance. In Van den Bussche, J. and Vianu, V., editors, *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330. Springer.

Carroll, J. J., Bizer, C., Hayes, P., and Stickler, P. (2005). Named Graphs, Provenance and Trust. In Ellis, A. and Hagino, T., editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*, pages 613–622. ACM.

Chapman, A. and Jagadish, H. V. (2009). Why Not? In Çetintemel, U., Zdonik, S. B., Kossmann, D., and Tatbul, N., editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, pages 523–534. ACM.

Chapman, A. P., Jagadish, H., and Ramanan, P. (2008). Efficient Provenance Storage. In Wang, J. T.-L., editor, *Proceedings of the ACM SIGMOD In-*

ternational Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008, pages 993–1006. ACM.

Cheney, J., Chiticariu, L., and Tan, W.-C. (2009a). Provenance in Databases: Why, How, and Where. *Foundations and Trends in Databases*, 1(4):379–474.

Cheney, J., Chong, S., Foster, N., Seltzer, M., and Vansummeren, S. (2009b). Provenance: A Future History. In Arora, S. and Leavens, G. T., editors, *Proceedings of the 24th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2009, October 25-29, 2009, Orlando, Florida, USA*, pages 957–964. ACM.

Cheney, J., Missier, P., Moreau, L., and De Nies, T. (2013). Constraints of the Provenance Data Model. Technical report, W3C.

Gil, Y., Cheney, J., Groth, P., Hartig, O., Miles, S., Moreau, L., Pinheiro da Silva, P., Coppens, S., Garijo, D., Gomez, J. M., Missier, P., Myers, J., Sahoo, S., and Zhao, J. (2010). Provenance XG Final Report. Technical report, W3C.

Gil, Y., Miles, S., Belhajjame, K., Deus, H., Garijo, D., Klyne, G., Missier, P., Soiland-Reyes, S., and Zednik, S. (2013). PROV Model Primer. Technical report, W3C.

Gilliland-Swetland, A. J. (2000). Enduring Paradigm, New Opportunities: The Value of the Archival Perspective in the Digital Environment. *Council on Library and Information Resources (CLIR) Reports, pub89, February 2000.* [`http://www.clir.org/pubs/reports/pub89/reports/pub89`, Retrieved February 21 2013].

Green, T. J., Karvounarakis, G., and Tannen, V. (2007). Provenance Semirings. In Libkin, L., editor, *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pages 31–40. ACM.

Groth, P., Gil, Y., Cheney, J., and Miles, S. (2012). Requirements for Provenance on the Web. *International Journal of Digital Curation*, 7(1):39–56.

Haughton, P. D. W., Todd, S. P., and Morton, A. C. (1991). Sedimentary provenance studies. *Geological Society, London, Special Publications*, 57(1):1–11.

Ingersoll, R. V. (2007). Provenance (geology). In *Encyclopedia of Science & Technology*, volume 14, pages 546–547. McGraw-Hill, 10th edition.

Jagadish, H. V. and Olken, F. (2004). Database Management for Life Sciences Research. *ACM SIGMOD Record*, 33(2):15–20.

Kwasnikowska, N., Moreau, L., and Van den Bussche, J. (2010). A Formal Account of the Open Provenance Model. ePrint 271819, University of Southampton.

Landesman, P. (1999). A 20th-Century Master Scam. *The New York Times Magazine, July 18 1999.* [`http://www.nytimes.com/library/magazine/archive/19990718mag-art-forger.html`, Retrieved January 30 2013].

Meliou, A., Gatterbauer, W., Halpern, J. Y., Koch, C., Moore, K. F., and Suciu, D. (2010). Causality in Databases. *IEEE Data Eng. Bull.*, 33(3):59–67.

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

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., and Van den Bussche, J. (2011). The Open Provenance Model Core Specification (v1.1). *Future Generation Computer Systems*, 27(6):743–756.

Moreau, L., Ludäscher, B., Altintas, I., Barga, R. S., Bowers, S., Callahan, S., Chin, Jr., G., Clifford, B., Cohen, S., Cohen-Boulakia, S., Davidson, S., Deelman, E., Digiampietri, L., Foster, I., Freire, J., Frew, J., Futrelle, J., Gibson, T., Gil, Y., Goble, C., Golbeck, J., Groth, P., Holland, D. A., Jiang, S., Kim, J., Koop, D., Krenek, A., McPhillips, T., Mehta, G., Miles, S., Metzger, D., Munroe, S., Myers, J., Plale, B., Podhorszki, N., Ratnakar, V., Santos, E., Scheidegger, C., Schuchardt, K., Seltzer, M., Simmhan, Y. L., Silva, C., Slaughter, P., Stephan, E., Stevens, R., Turi, D., Vo, H., Wilde, M., Zhao, J., and Zhao, Y. (2008). The First Provenance Challenge. *Concurrency and Computation: Practice and Experience*, 20(5):409–418.

Moreau, L., Missier, P., Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Klyne, G., Lebo, T., McCusker, J., Miles, S., Myers, J., Sahoo, S., and Tilmes, C. (2013). PROV-DM: The PROV Data Model. Technical report, W3C.

Müller, H. and Naumann, F. (2003). Data Quality in Genome Databases. In Eppler, M. J. and Helfert, M., editors, *Eighth International Conference on Information Quality (IQ 2003), November 7-9. 2003*, pages 269–284. MIT.

Pipino, L. L., Lee, Y. W., and Wang, R. Y. (2002). Data Quality Assessment. *Communications of the ACM*, 45(4):211–218.

Scannapieco, M., Missier, P., and Batini, C. (2005). Data quality at a Glance. *Datenbank-Spektrum*, 14:6–14.

Simmhan, Y., Groth, P., and Moreau, L. (2011). Special Section: The third provenance challenge on using the open provenance model for interoperability. *Future Generation Computer Systems*, 27(6):737–742.

Simmhan, Y. L., Plale, B., and Gannon, D. (2005). A Survey of Data Provenance Techniques. Technical Report IUB-CS-TR618. Technical report, Computer Science Department, Indiana University.

Simmhan, Y. L., Plale, B., and Gannon, D. (2006). Towards a Quality Model for Effective Data Selection in Collaboratories. In Barga, R. S. and Zhou, X., editors, *Proceedings of the 22nd International Conference on Data Engineering Workshops, ICDE 2006, 3-7 April 2006, Atlanta, GA, USA*, page 72. IEEE Computer Society.

Sullivan, M. (2005). Provenance: Ignore It At Your Peril. *Forbes: The Forbes Collector, February 2 2005.* [http://www.forbes.com/2005/02/11/cz_ms_0211soapbox2_inl.html, Retrieved January 21 2013].

Tayi, G. K. and Ballou, D. P. (1998). Examining data quality. *Communications of the ACM*, 41(2):54–57.

Washington Conference on Holocaust-Era Assets (1998). Washington Conference Principles On Nazi-Confiscated Art. From the Conference on Holocaust-Era Assets, Washington, DC, December 3, 1998. [http://www.state.gov/www/regions/eur/981203_heac_art_princ.html, Retrieved January 28 2013].

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Provenance in Databases en het Web

Richting: **master in de informatica-databases**
Jaar: **2013**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Debosschere, Maxime

Datum: **19/06/2013**