University of Crete
Computer Science Department

Reasoning and Evolution of Event-based Provenance Information

Christos Strubulis
Master's Thesis

Heraklion, October, 2012

Reasoning and Evolution of Event-based Provenance Information

Christos Strubulis

Thesis submitted in partial fulfilment of the requirements for the

Masters' of Science degree in Computer Science

University of Crete
School of Sciences and Engineering
Computer Science Department
Knossou Av., P.O. Box 2208, Heraklion, GR-71409, Greece

Thesis Advisors: Prof. *Plexousakis Dimitris*

This work has been performed at the **University of Crete, School of Sciences and Engineering, Computer Science Department**

The work has been supported by the **Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS)**

University of Crete
Computer Science Department

Reasoning and Evolution of Event-based Provenance Information

Thesis submitted by
Christos Strubulis
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: _____
Christos Strubulis

Committee approvals: _____
Dimitris Plexousakis
Professor Thesis Supervisor

_____
Yannis Tzitzikas
Assistant Professor Committee Member

_____
Martin Doerr
Research-Director, Foundation for Research and Technology - Hellas (FORTH)

Departmental approval: _____
Aggelos Bilas
Professor, Director of Graduate Studies

Heraklion, October 2012

Reasoning and Evolution of Event-based Provenance Information

Christos Strubulis
Master's Thesis
Computer Science Department, University of Crete

Abstract

Provenance of a resource is a record that describes entities and processes involved in producing and delivering or otherwise influencing that resource. Generally, the above record can be considered as information that has great importance in the scientific community regarding the experiments that are conducted as part of its research. This information can be later used for the validation, interpretation or the reproduction of scientific results and is commonly stored on metadata placed in various Metadata Repositories (MRs) or Knowledge Bases (KBs).

However, in various settings it is prohibitive to store the complete provenance information because of (a) the immense space requirements needed and (b) the difficulty of controlling its quality due to the existence of possible errors.

We address the problem by introducing provenance-based inference rules as a means to reduce the amount of provenance information that has to be stored, and to ease quality control (e.g., corrections). Roughly, we show how information can be *propagated* by identifying a number of basic inference rules over a core conceptual model representing provenance. The propagation of provenance concerns fundamental modeling concepts such as *actors, activities, events, devices* and *information objects* and their associations.

However, since a KB is not static but changes over time due to several factors, a rising question is how we can satisfy change requests while still supporting the aforementioned inference rules. Towards this end, we elaborate on the specification of the required add/delete operations, and consider two different semantics for deletion of information. We describe the corresponding change algorithms, and we report on comparative results for two repository strategies regarding the derivation of new knowledge. The results allow us to understand the tradeoffs related to the use of inference rules on storage space and performance of queries and updates.

Συλλογιστική και Εξέλιξη της Πληροφορίας Προέλευσης βάσει Γεγονότων

Περίληψη

Η προέλευση (provenance) ψηφιακών αντικειμένων αποτελείται από μια καταγραφή από οντότητες και διαδικασίες οι οποίες εμπλέκονται και επηρεάζουν την παραγωγή και δημιουργία τέτοιων αντικειμένων. Γενικά, αυτή η καταγραφή εμπεριέχει πληροφορία η οποία είναι εξαιρετικής σημασίας στην επιστημονική κοινότητα και ειδικότερα στην διεξαγωγή ερευνητικών πειραμάτων. Αυτή η πληροφορία χρησιμοποιείται για την επικύρωση, ερμήνευση ή την αναπαραγωγή εκ νέου των επιστημονικών αποτελεσμάτων και συχνά καταγράφεται με μεταδεδομένα αποθηκευμένα σε διάφορα αποθετήρια μεταδεδομένων (metadata repositories MRs) ή βάσεις γνώσης (knowledge bases KBs).

Παρόλα αυτά, τέτοιου είδους συστήματα μπορούν να παράγουν τεράστια ποσοστά πληροφοριών προέλευσης, περιλαμβάνοντας επαναλαμβανόμενη πληροφορία. Επομένως, σε διάφορες ρυθμίσεις συστημάτων είναι απαγορευτικό να καταγραφεί ολόκληρη η πληροφορία προέλευσης λόγω των(α) των μεγάλες απαιτήσεων χώρου που χρειάζεται και (β) τη δυσκολία ελέγχου της ποιότητας της πληροφορίας εξαιτίας λαθών που μπορεί να εμφανιστούν.

Αντιμετωπίζουμε το παραπάνω πρόβλημα εισάγωντας λογικούς κανόνες συμπερασμάτων, με σκοπό τη μείωση της πληροφορίας προέλευσης που πρέπει να αποθηκευτεί και καλύτερη διαχείριση της ποιότητας της πληροφορίας (π.χ., σε περίπτωση διορθώσεων). Εν περιλήψει, αναλύουμε πως η πληροφορία μπορεί να *διαδοθεί* προσδιορίζοντας βασικούς κανόνες συμπερασμάτων σε ένα βασικό εννοιολογικό μοντέλο αναπαράστασης πληροφορίας προέλευσης. Η διάδοση της πληροφορίας αφορά θεμελιώδεις έννοιες αναπάρασταση όπως *υπεύθυνους φορείς, δραστηριότητες, γεγονότα, συσκευές* και *πληροφοριακά αντικείμενα*, μαζί με τις σχέσεις μεταξύ αυτών.

Ωστόσο, εφόσο μια βάση γνώσης (KB) δεν είναι στατική αλλά αλλάζει κατά τη διάρκεια του χρονου εξαιτίας αρκετών παραγόντων, ένα ζήτημα είναι πως μπορούμε να ικανοποιήσουμε αιτήσεις ενημέρωσης υποστηρίζοντας παράλληλα τους προηγούμενους κανόνες. Προς το σκοπό αυτό, προσδιορίζουμε τις λειτουργίες πρόσθεσης/αφαίρεσης πληροφορίας, λαμβάνοντας υπόψιν δυο διαφορετικές σημασιολογίες για την περίπτωση της αφαίρεσης. Περιγράφουμε τους αντίστοιχους αλγορίθμους και αναφέρουμε συγκριτικά αποτελέσματα για δυο αποθηκευτικές στρατηγικές σχετικά με την παραγωγή νέας γνώσης. Τα αποτελέσματα μας επιτρέπουν να καταλάβουμε το αντιστάθμισμα σχετικά με τη χρήση των κανόνων συμπερασμάτων οσον αφορά τον αποθηκευτικό χώρο και την απόδοση επερωτήσεων και ενημερώσεων.

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον κ. Γιάννη Τζίτζικα για την πολύτιμη καθοδήγηση και ουσιαστική συμβολή του για την ολοκλήρωση αυτής της εργασίας λειτουργώντας σαν άτυπος επόπτης. Η οργανωτική εμπειρία αλλά και η ψυχραιμία του στον τρόπο αντιμετώπισης ορισμένων θεμάτων με βοήθησαν στο να βελτιώσω τον εαυτό μου και τις ικανότητες μου.

Επιπλέον θα ήθελα να ευχαριστήσω και τον κ. Martin Doerr για τις υποδείξεις του, και τη βοήθεια του καθόλη τη διάρκεια της εργασίας. Οι φιλοσοφικές αναλυτικές συζητήσεις που είχαμε ήταν ουσιαστικές, ευχάριστες και επικοδομητικές.

Επίσης, θα ήθελα να ευχαριστήσω τον επόπτη καθηγητή μου κ. Δημήτρη Πλεξουσάκη για τις εύστοχες παρατηρήσεις του και συμβουλές για την περάτωση αυτής της εργασίας, αλλά και το ενδιαφέρον του μετά το τέλος της εργασίας σχετικά με τη μελλοντική πορεία μου.

Ακόμη, δεν θα πρέπει να παραλείψω τον Γιώργο Φλουρή. Τον ευχαριστώ για τις υποδείξεις του σε 'ασήμαντα' ζητήματα που ήταν όμως καθοριστικά. Η αναλυτικότητα του, ο επαγγελματισμός του και η υπομονή του λειτούργησαν ως πρότυπο και αξίες τόσο για αυτή την εργασία όσο και για μελλοντικές.

Παράλληλα θα ήθελα να ευχαριστήσω το Εργαστήριο Πληροφοριακών Συστημάτων του Ινστιτούτου Τεχνολογίας και Έρευνας για την ευκαιρία που μου δόθηκε στο να χρησιμοποιήσω τον εξοπλισμό και την τεχνογνωσία του.

Ένα μεγάλο ευχαριστώ θα ήθελα να δώσω σε όλους τους φίλους μου για την υποστήριξη. Θα ήθελα να ευχαριστήσω ξεχωριστά τους Μιχάλη και Γιώργο για την ηθική συμπαράσταση τους σε δύσκολες και εύκολες στιγμές, την κατανόηση τους και τα όσα περάσαμε μαζί.

Τέλος, θα ήθελα να εκφράσω την ευγνωμοσύνη μου στη μητέρα μου Κατερίνα και στον αδερφό μου Αλέξανδρο για την στήριξη, την ενθάρρυνση και την ολόψυχη αγάπη τους στις δύσκολες στιγμές που ήταν αρκετές. Σας ευχαριστώ πολύ για όλα.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

We live in a digital age. Its beginning has been marked by the invention of the personal computer and one of its characteristics is the easy and free transfer of information. However, this characteristic has a critical effect on the representation of information. Information previously written or engraved on materials such as stone, wood or paper has now obtained a digital representation. These have now been replaced by digital storage media such as hard drives, disks, DVDs, etc. However, as technology improves, their capacity increases allowing for larger amounts of information to be created and stored (RidingTheWave, 2010). In 2011 "the amount of information created and replicated will surpass 1.8 zettabytes (1.8 trillion gigabytes)" (Gantz & Reinsel, 2011).

This information explosion is also one of the effects of digital age in science. Digital technologies in various domains e.g., 3D imaging in cultural heritage, can produce vast amounts of digital data. These data are of great significance to the scientific research and its progress. However, their provenance might be of greater importance for scientists, especially, for the experimental process of scientific method. Their provenance assist scientists to interpret, validate, trust and reproduce the experimental results.

1.1 Background

The word *provenance* comes from the french verb *provenir*. According to the Oxford English Dictionary[1] and Merriam-Webster Online Dictionary[2] provenance is defined as follows.

Definition 1. (OED Provenance Definition) *(i) the fact of coming from some particular source or quarter; origin, derivation, (ii) the history or pedigree of a work of art, manuscript, rare book, etc.; concretely, a record of the ultimate derivation and passage of an item through its various owners.*

Definition 2. (MWO Provenance Definition) *(i) the origin, source, (ii) the history of ownership of a valued object or work of art or literature.*

[1] http://www.oed.com/
[2] http://www.merriam-webster.com/dictionary/

Initially the term was used for works of art. For instance, the provenance of works of fine art, antiques and antiquities often assumes great importance. Documented evidence of provenance for an object can help to establish that it has not been altered, and its is not a forgery or a reproduction. Knowledge of provenance can help to assign the work to a known artist and a documented history can be of use in helping to prove ownership.

Moreover, the quality of provenance of an important work of art can make a considerable difference to its selling price in the market; this is affected by the degree of certainty of the provenance, the status of past owners as collectors, and in many cases by the strength of evidence that an object has not been illegally excavated or exported from another country.

Scientific research is generally held to be of good provenance. Provenance of scientific results may be of greater significance than the scientific results themselves. It can be used by scientists to further understand all the factors that were involved in the derivation of the latter ones.

Workflow systems in the context of e-Science can be used as a tool to ease the management of complex scientific computational processes and record provenance information of experimental results. A workflow system is "a system that defines, creates and manages the execution of workflows" (Romano, 2008). Complex processes are organized into sequences of connected steps and each step represents a task which has to be executed. The input data each transferred through each step until the final result.

The provenance of workflow systems is in the form of metadata that contains information regarding the derivation of a workflow result. By recording this information, the result can be reproduced and determine its reliability. On the other hand, since amount of data regarding the scientific results is increasing, as discussed previously, the amount of data regarding their provenance is increasing as well. In fact, the size of the latter often exceeds the size of the former. Thus, the issue here is in what way the amount of provenance information can be minimized in order to reduce its space storage requirements. This challenge is part of the motivation for this study as discussed in the next section.

1.2 Motivation

In the context of computer systems, provenance of a data product can be defined as *"the process that led to that data product"*. As data products or data outputs are produced by the execution of computer programs, their provenance can be recorded under "process". It can include, for example, the data input used in that program, the hardware or its users (Moreau, 2010). We will be referring to these data products also as digital items or digital objects.

In a naïve view, the provenance of a digital item can be seen as a record specific to it of the events and their contexts that have causally contributed or had significant influence on its content. However, digital objects do not undergo "changes" as material items, which sum up to an accumulative effect. Each modification leaves behind the original version, which may or may not be reused in another context.

Hence, any realistic creation process of digital content gives rise to a set of digital items – temporary or permanent, connected by metadata forming a complex DAG (Directed Acyclic Graph) via the individual processes contributing to it.

The provenance "history" of a single item is the DAG of all "upstream" events until the ultimate empirical capture (measurement), simulation S/W run or human creative process. In a production environment, often controlled by a work-flow system, there are no clear a priori rules which data item will be permanent. Interactive processes of inspection of intermediate results and manual interventions or changes of processing steps may fall out of any preconceived order of events.

Therefore, the only chance to capture the complete provenance in a reliable way is by monitoring metadata of each step individually, and then concatenating these elementary metadata into the complete provenance history of an item by use of shared URIs. In cases such as the empirical 3D model generation, where ten thousands of intermediate files and processes of hundreds of individual manual actions are no rarity, it is prohibitive to register the complete history of each item its because of the immense repetition of facts between the files: On the one side, the storage space needed would be blown up by several orders of magnitude, and on the other side any correction of erroneous input would require tracing the huge proliferation graph of this input.

The major sources of redundancy are:

- transitivity of contribution: an input A to an input B to an output C is a contribution to C,
- propagation of features of digital objects preserved by certain processing steps, such as motif, shape, volume, edges, etc.,
- transitivity of part-whole relations,
- propagation of properties from wholes to parts and vice versa, be it for objects and their parts or for processes and their sub- and superprocesses.

The above notion of redundancy is not yet formally well understood and may even not be strictly logical. For instance, it is a question of convention, if we regard that persons carrying out a process carry out all of its subprocesses. Even if we accept this convention, it is impractical for the monitoring system to expand the persons to all subprocesses once we encounter that one person left too early. Therefore, the question is rather which system of propagation and exception rules would minimize redundancy for the typical statistics of processes under consideration.

If such a system has been established, we may distinguish three epistemological situations:

- The registered facts can reliably and completely be registered by a monitoring system, such as a workflow shell.
- There are facts which users need to input manually to the monitoring system but may be unwilling to do so.
- Facts come from different monitoring systems or uncontrolled human input.

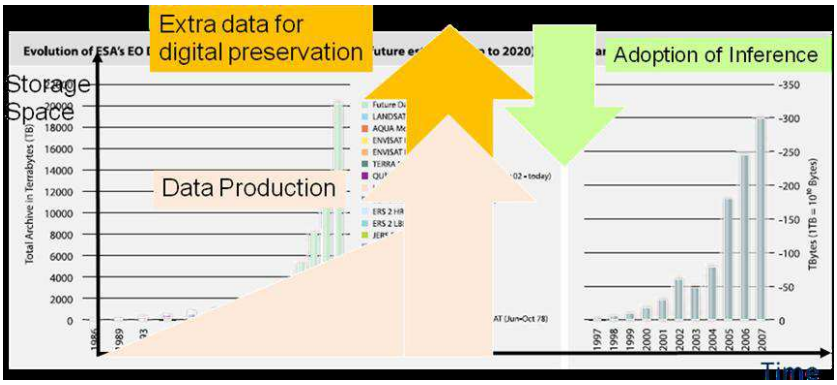


Figure 1.1: Data production and the increased space requirements

The reasoning forms we consider in this work aim at the dynamic completion (deduction) of facts from original input by resolving transitive closures and propagating the properties. This is complementary to reasoning on "data provenance", which traces dependency of individual elements of data sets between input and output, and works which recognize differences of variants of instances of complex workflows.

Deduced facts have a different epistemological status than primary input: Their truth depends directly on the truth of the primary input, whereas elements of primary input have no formal mutual dependency. This distinction is vital for effective management of error corrections in such data. Eliminating redundancy on the input side is a powerful measure to reduce input errors.

For representing provenance, we adopt a core conceptual model that contains fundamental (for provenance) modelling concepts such as *actors, activities, events, devices, information objects* and their associations. Over this model we identify three basic custom inference rules which occur frequently in practice. Of course, one could extend this set according to the details and conventions of the application at hand.

However we should consider that a knowledge base (either stored in a system or composed by various metadata files) changes over time. The rising question is how we can satisfy change requests while still supporting the aforementioned inference rules, for instance: *how one can delete provenance information that has been propagated, i.e. information that is produced by inference rules and is not explicitly stored in the repository?*

To tackle the update requirements we propose three operations, namely *Add*, *Disassociate*, and *Contract*, and discuss their semantics along with the required algorithms. The last two operations concern the deletion of information and are founded on the related philosophical viewpoints, i.e. *foundational* and *coherence* semantics. These viewpoints actually differ in the way they value the inferred knowledge in comparison to the explicitly ingested one. This analysis is not covered by the existing works, because existing approaches Konstantinidis *et al.* (2008); Magiridou *et al.* (2005); Gutierrez *et al.* (2011) consider only one of the viewpoints and only the standard RDF/S inference rules (not custom inference rules).

Having specified provenance propagation, and provenance update, the next step is to evaluate this approach. For this reason we report comparative results for two repository strategies regarding the derivation of new knowledge. In our experiments we used both real and synthetic datasets. The objective of this evaluation is to enable us to understand the trade off between space usage, query performance and update performance. To this end, we conducted experiments that measure the storage space before and after the application of the inference rules, and measure the impact of inferencing on the performance of queries and updates. In brief, the results showed that the suggested strategy significantly reduces the storage space requirements of provenance information while the performance of queries and updates is slightly higher.

1.2.1 Motivating Scenario

As an application example, in the context of the IP 3D-COFORM[3] [3D-COFORM (Tools and expertise for 3D collection formation)] , the Digital Provenance Models developed in the CASPAR IP[4] [CASPAR (Cultural, Artistic and Scientific knowledge for Preservation, Access and Retrieval)] has been enhanced to describe in any required detail the very complex data acquisition and data processing processes both on an atomic - processing step by processing step - and on an integrated level - from acquisition to data ready for publishing.

Fig. 2: The acquisition process. The string helps to keep the camera at the same distance from the chosen surface point. The Autofocus is switched off, both for greater speed and greater accuracy.

Figure 1.2: The acquisition Phase-Workflow example (3D-Coform)

Note that a single acquisition process may create thousands of images and some terabytes of data. The complex processes yielding massive intermediate data and multiple versions of final products, reprocessing with improved methods or corrected input, give raise to a need for complex generic reasoning over provenance data in order to solve digital preservation tasks, such as:

- propagation of properties from super to subprocesses (and vice versa), such as motif, shape, volume, edges, camera parameters etc.,
- propagation along processing steps,
- merging metadata of intermediate steps,

[3] www.3d-coform.eu/

[4] www.casparpreserves.eu/caspar-project/project-governance.html

- error propagation from the primary input

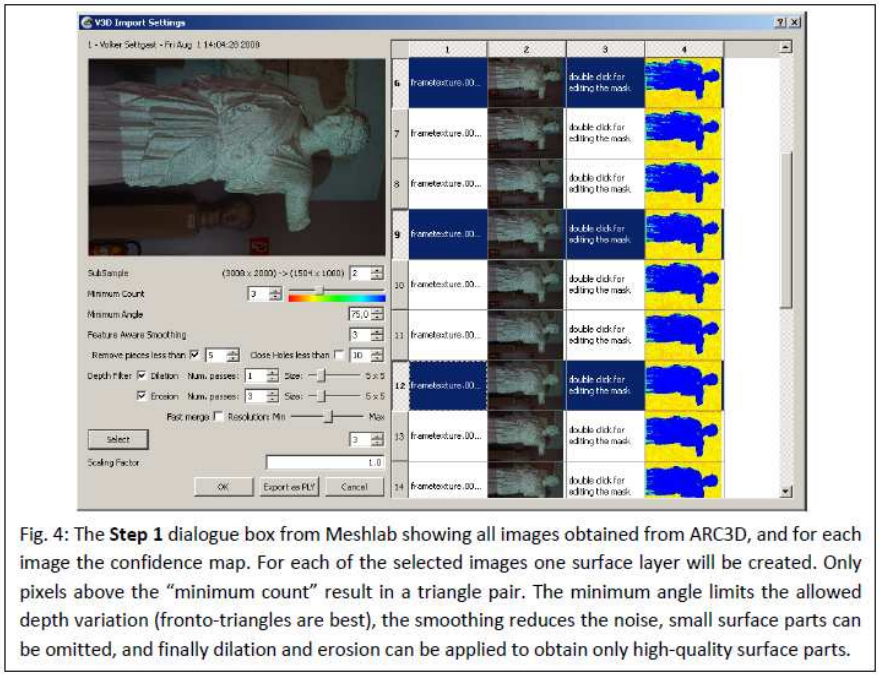


Fig. 4: The **Step 1** dialogue box from Meshlab showing all images obtained from ARC3D, and for each image the confidence map. For each of the selected images one surface layer will be created. Only pixels above the "minimum count" result in a triangle pair. The minimum angle limits the allowed depth variation (fronto-triangles are best), the smoothing reduces the noise, small surface parts can be omitted, and finally dilation and erosion can be applied to obtain only high-quality surface parts.

Figure 1.3: Intermediate images of an acquisition process (3D-Coform)

The purpose of this study is to examine and exploit the above kind of reasoning.

1.3 Thesis Contribution

In this thesis our focus concentrates on the basic reasoning and inference rules based on provenance information and the evolution of such information.

The main contributions of this thesis are:

i. We propose the introduction of inference mechanisms in order to reduce the storage space requirements of provenance information. We argue that their adoption has many advantages regarding the general management of provenance information:

   (a) They aim at the completeness of possible incomplete information regarding past events (i.e., provenance) automatically,

   (b) they reduce the search space of possible errors that can appear among inferred or non inferred facts in a metadata environment and thus allowing their easier correction,

   (c) they reduce the overhead in time and human effort in the process of ingesting new knowledge,

   (d) they reduce the space storage requirements of provenance information.

ii. We propose reasoning rules in order to eliminate redundancy in provenance information. The inference mechanisms introduced are based on a different concept than the ones proposed by other works such as (Moreau *et al.*, 2011) and (Moreau & Missier, 2011). The latter are based on the derivation of further dependencies among data, while the former are based on the propagation of properties and attributes in hierarchical structures of modelling provenance information.

iii. We elaborate on the problem of the provenance information evolution. We define the semantics of update operations of a KB taking into account our custom inference rules. Based on the philosophical debate between foundationalism and coherentism of the inferred knowledge significance we have identified two ways of deleting facts. This presents us with two novelties since (a) current works (Bechhofer *et al.*, 2001; Gabel *et al.*, 2004; Noy *et al.*, 2000; Sure *et al.*, 2003; Klein & Noy, 2003) do not consider both approaches (b) or/and take into account only the RDF/S inference rules but not any user defined custom inference rules (Magiridou *et al.*, 2005; Konstantinidis *et al.*, 2008; Gutierrez *et al.*, 2011).

iv. We have conducted several experiments measuring the query performance and the storage space requirements comparing two repository policies. Even though we focused on our approach and for one rule, the results are indicative for the problem of materializing (or not) the inferences in hierarchical RDF structures including the task of computing and storing transitive closures of DAGs.

1.4 Thesis Organization

The rest of this thesis is organized as follows:

Chapter 2 presents the basic assumptions and definitions involved in our reasoning. More specifically, it elaborates on the assumptions regarding our knowledge representation and the management of provenance information. It also introduces our application's ontology and finally some technical and working definitions adopted by in the rest of this study.

Chapter 3 presents our provenance based inference rules. Each rule is explained and encoded in first-order logic (FOL). Moreover, real examples from the cultural heritage domain are given for a better understanding of the purpose of each rule.

In Chapter 4 we tackle the challenge of how our knowledge can be changed by taking into account the inference rules which we introduced in Chapter 3. In this respect, we discuss how our KB can be updated by addition or deletion of information. More particularly for the case of deletion, we identify two ways for handling such an operation.

Chapter 5 details on the implementation considerations of this study, analysing the available repository policies and requirements with respect to the inference rules. Moreover, it presents the results of the conducted experimental evaluation using as a case study the Virtuoso triple store. We conducted experiments on real and

synthetic data comparing two repository storage policies regarding the derivation of new knowledge.

Chapter 6 is composed by three sections. The first one tackles the challenge of how a larger set of rules can be defined presenting a larger set of reasoning rules and a metamodel showing the patterns that can be identified for this purpose. The other two sections extends our reasoning on parthood relations by defining two kinds of it, the essential and the optional one. Moreover, we take into consideration some temporal aspects regarding the latter two specializations of the parthood relation.

Chapter 7 compares our approach to those from related works in the bibliography. The comparison is performed considering the tree basic parts or this study: a)the provenance storage, b) the inference mechanisms and c) the problem of knowledge evolution.

Chapter 8 elaborates on the applicability of this study, pondering on what would happen if one or more of them would be ignored. It also concludes this study summarizing its results and suggests directions for further research and work.

Chapter 2

Preliminaries

In this Chapter we analyse some of the assumptions that we will take into consideration into our reasoning and more generally throughout this study. Specifically, Section 2.1 details the assumption regarding the origin of our knowledge (Section 2.1.1) and what will be called as *explicit* and *implicit* knowledge (Section 2.1.2). Section 2.2 presents how provenance information can be modelled, introducing our application context (Section 2.2.2) and its basic concepts (Section 2.3). Finally, Section 2.4 explains some other more technical assumptions and definitions adopted in the rest of our discussion.

2.1 Knowledge Representation and Epistemological Assumptions

2.1.1 Epistemological Assumptions

The general assumption endorsed by this study is the provenance information which has been recorded empirical evidence (Mudge *et al.*, 2008). In this regard, we may distinguish three epistemological situations:

1. The registered facts can reliably and completely be registered by a monitoring system, such as a workflow shell.

2. There are facts which users need to input manually to the monitoring system and may be not willing to do so.

3. Facts come from different monitoring systems or uncontrolled human input.

One of the challenge in this respect is that in many cases that the provenance information may not even be in digital form. In this case, an expert user has to edit them in digital form for example in xml format and ingest them in a metadata environment. Many times however, because of the complexity of provenance information, the human effort required to perform this task is so high that the process might fail or result to errors in the final digitized metadata.

Moreover, because of that complexity regarding provenance information we assume that our records regarding the past are incomplete. This incompleteness may

be due to the inability of the maintainer to provide sufficient information or due to more fundamental problems of cognition in the system's domain. Such problems are characteristic of cultural information systems. This idea is related to the *Open World Assumption (OWA)*, characterizing a knowledge base system that stores incomplete knowledge. As an example, one cannot list "all Physical Objects known to the system that are not Biological Objects in the real world", but one may of course list "all items known to the system as Physical Objects but that are not known to the system as Biological Objects".

Our approach tackles this challenge by reducing the amount of provenance metadata that the user has to ingest into a metadata environment. The role of the inference rules is to complete the information, that the user would have to completed by himself, dynamically and automatically. Also as an extra benefit, since the original ingested information has been reduced, the search space for any possible errors has also been reduced. As a result, any errors appearing in the inferred facts can easily be corrected since their cause should be attributed only to the errors in the original ingested/non inferred facts. The issue of how these errors can be corrected is addressed in Chapter 4 discussing the problem of knowledge evolution.

2.1.2 Explicit and Implicit Knowledge

A Knowledge Base (KB) is a set of sentences (Russell & Norvig, 2010). We shall use the term KB to refer to a Knowledge Base in the logical sense, either stored in a system or composed by the contents of several metadata files. The knowledge of a KB is expressed by the contained facts generated by the metadata of the files. Usually the contained information is assumed to be explicit in the sense that it is a product from metadata that have been ingested (i.e., primary input) into the KB through manual or automated processes.

However, deductive information derived from the explicit one in the form of logical consequences is assumed to be implicit. More formally, we say that for a given set of sentences Σ (i.e., the KB), a sentence ϖ is a *consequence* of Σ, in symbols $\Sigma \models \varphi$, iff φ is true in Σ (Chang & Keisler, 1990). Notice that the inferred knowledge considered in this work is different than tacit knowledge (Polanyi, 1966; Smith, 2003); the former has some grounds on explicit one, whereas the latter may be the result of common sense or common knowledge.

In this work deductions are made by the application of inference rules on the primary/ingested data as discussed in Chapter 3. Derived facts have a different epistemological status than primary input: Their truth depends directly on the truth of the primary input, whereas elements of primary input have no formal mutual dependency. Based on this distinction of the origin of information we characterize knowledge as *explicit* and *implicit*. Moreover, hereafter we shall use the terms *inferred*, or *derived* equivalent to *implicit*.

2.2 Provenance Modelling and Management

There are different ways for representing provenance information depending on the respective underlying metadata environment. One of these ways is that provenance

of a data product can be represented via *annotations* comprising its derivation history. These annotations are readily usable as metadata (Simmhan *et al.*, 2005). The most usual format representing provenance metadata is XML. The data/metadata are expressed in various Semantic Web Languages (RDF/S, OWL) under schemata (ontologies) that are globally accessible via Internet and can be combined to a certain degree (Zhao *et al.*, 2004). We elaborate on ontologies in the subsequent section.

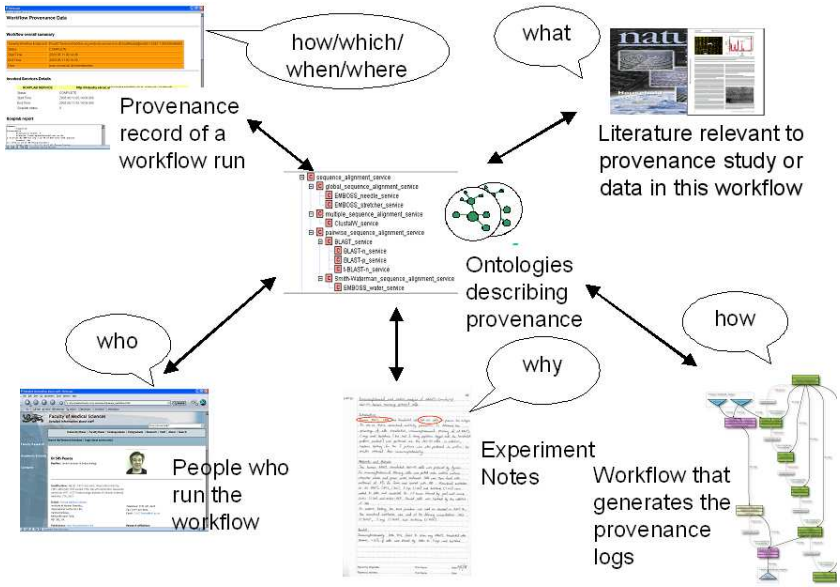


Figure 2.1: Web of Provenance (extracted from (Zhao *et al.*, 2004))

2.2.1 On Ontologies and Models

A formal definition of an ontology is thought to be given by Gruber (1995) and is the following:

Definition 3. ***(Gruber's Ontology definition)*** *An ontology is an explicit specification of a conceptualization.*

According to Gruber a conceptualization is "an abstract, simplified view of the world that we wish to represent for some purpose. Ontologies in Artificial Intelligent are explicit specifications of a conceptualization aiming at formally represent an aspect of the world. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g.,classes, relations, functions, or other objects) with human-readable text, describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms" (Gruber, 1995).

Another more recent definition has been given by Guarino (1998):

Definition 4. ***(Guarino's Ontology definition)*** *An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world. The intended models of a*

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xml:lang="en"
xml:base="http://www.ics.forth.gr/isl/rdfs/3D-COFORM_CRMdig.rdfs">
<rdfs:Class rdf:ID="D1.Digital_Object">
        <rdfs:subClassOf
        rdf:resource="http://www.ics.forth.gr/isl/rdfs/
        3D-COFORM_CIDOC-CRM.rdfs#E73.Information_Object"/>
</rdfs:Class>
<rdfs:Class rdf:ID="D2.Digitization_Process">
        <rdfs:subClassOf rdf:resource="#D11.Digital_Measurement_Event"/>
</rdfs:Class>
<rdfs:Class rdf:ID="D3.Formal_Derivation">
        <rdfs:subClassOf rdf:resource="#D10.Software_Execution"/>
</rdfs:Class>
<rdf:Property rdf:ID="L1F.digitized">
        <rdfs:domain rdf:resource="#D2.Digitization_Process"/>
        <rdfs:range
        rdf:resource="http://www.ics.forth.gr/isl/rdfs/
        3D-COFORM_CIDOC-CRM.rdfs#E18.Physical_Thing"/>
        <rdfs:subPropertyOf
        rdf:resource="http://www.ics.forth.gr/isl/rdfs
        /3D-COFORM_CIDOC-CRM.rdfs#P39F.measured"/>
</rdf:Property>
<rdf:Property rdf:ID="L1B.was_digitized_by">
        <rdfs:domain
        rdf:resource="http://www.ics.forth.gr/isl/rdfs/
        3D-COFORM_CIDOC-CRM.rdfs#E18.Physical_Thing"/>
        <rdfs:range rdf:resource="#D2.Digitization_Process"/>
        <rdfs:subPropertyOf
        rdf:resource="http://www.ics.forth.gr/isl/rdfs/
        3D-COFORM_CIDOC-CRM.rdfs#P39B.was_measured_by"/>
</rdf:Property>
<rdf:Property rdf:ID="L2F.used_as_source">
        <rdfs:domain rdf:resource="#D10.Software_Execution"/>
        <rdfs:range rdf:resource="#D1.Digital_Object"/>
        <rdfs:subPropertyOf rdf:resource="#L10F.had_input"/>
</rdf:Property>
<rdf:Property rdf:ID="L2B.was_source_for">
        <rdfs:domain rdf:resource="#D1.Digital_Object"/>
        <rdfs:range rdf:resource="#D10.Software_Execution"/>
        <rdfs:subPropertyOf rdf:resource="#L10B.was_input_of"/>
</rdf:Property>
<rdf:Property rdf:ID="L4F.has_preferred_label">
        <rdfs:domain rdf:resource=      "http://www.ics.forth.gr/isl/rdfs/
        3D-COFORM_CIDOC-CRM.rdfs#E1.CRM_Entity"/>
        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
        <rdfs:subPropertyOf
        rdf:resource="http://www.w3.org/2000/01/rdf-schema#label"/>
</rdf:Property>
</rdf:RDF>
```

Figure 2.2: An example of CRMdig in RDF/S

logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models.

As a result, ontologies or conceptual models or vocabularies have been developed in several domains of science such as in Biology or Archaeology in order to model the part of our world referring to these domains. For instance, several models have been developed for modelling provenance such as:

- Open Provenance Model
- Provenir ontology
- Provenance Vocabulary
- Proof Markup Language
- Dublin Core
- PREMIS
- WOT Schema
- SWAN Provenance Ontology
- Semantic Web Publishing Vocabulary
- Changeset Vocabulary

Among the previously referred models the most popular one is the Open Provenance Model (OPM) (Moreau *et al.*, 2011).

OPM (Open Provenance Model)

The Open Provenance Model (OPM) is a data model for provenance suggested in a workshop in August 2007 in Salt Lake City (Moreau *et al.*, 2011). In general, it is defined as an annotated causality graph encoding information about events that have happened in the past. One of the requirements of the OPM is to support a digital representation of provenance for any "thing", whether produced by computer systems or not.

The Open Provenance Model is a model of artifacts in the past, explaining how they were derived. Likewise, processes also occurred in the past, i.e. they have already completed their execution; in addition, processes can still be currently running (i.e., they may have not completed their execution yet). The main classes are the agents, the processes and the artifacts as it is shown in Fig. 2.3.

We can say that the ontology assumed by OPM is minimal. It comprises only 3 classes (Artifact, Process, Agent) and five associations among them (used, wasGeneratedBy, wasControlledBy, wasTriggeredBy, wasDerivedFrom). The associations and the involved entities are depicted in Fig. 2.4. OPM does not make any distinction between digital and physical objects and their provenance is represented by an annotated causality graph.

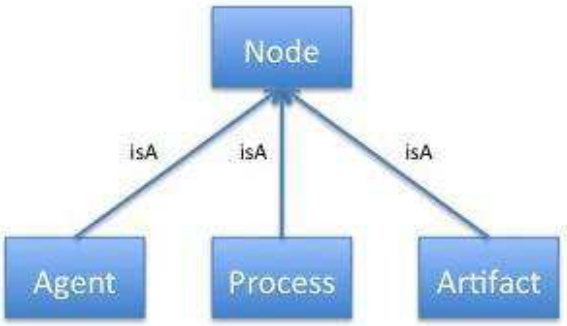


Figure 2.3: The main classes of OPM

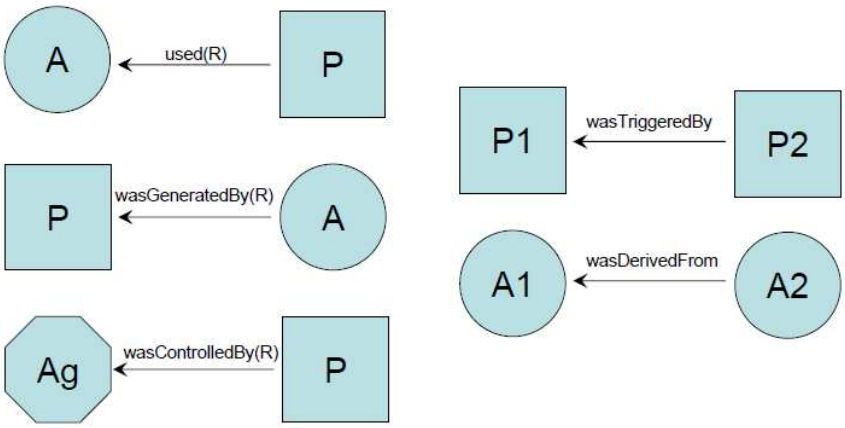


Figure 2.4: The main properties of OPM

The Open Provenance Model has been selected as a reference provenance model by the Provenance Incubator Group , part of W3C incubator Activity to provide a mapping for a core set of other provenance vocabularies and models (presented previously) which are well known in the community[1].

2.2.2 Application Context

It is thought that OPM's main advantage is its minimality and its purpose to model "anything". The result is a simple ontology that a non expert user can use it in order to model provenance information. On the other hand, that minimality could also be considered as a major disadvantage. Since the number of its classes is limited only to three, one could argue that its corresponding ontological structure is not rich enough to model the "world" of provenance. Thus, it suffers from overgeneralization and possible loss of information. In order to overcome this disadvantage, in this work

[1] http://www.w3.org/2005/Incubator/prov/wiki/Provenance_Vocabulary_Mappings

we consider CRMdig (Theodoridou *et al.*, 2010) a structurally object-oriented model which is an extension of the CIDOC CRM ontology (ISO 21127:2006) (Doerr, 2003).

In brief, CIDOC CRM is a core ontology describing the underlying semantics of data schemata and structures from all museum disciplines and archives. It is the result of a long-term interdisciplinary work and agreement and it has been derived by integrating (in a bottom-up manner) hundreds of metadata schemas. In essence, it is a generic model of recording "what has happened" in human scale.

It can generate huge, meaningful networks of knowledge by a simple abstraction: history as meetings of people, things and information. Regarding the application of CIDOC CRM for scientific data, the idea is that scientific data and metadata can be considered as historical records. Scientific observation and machine-supported processing is initiated on behalf of and controlled by human activity. Things, data, people, times and places are related by events. Other relations are either deductions from events or found by observation. Figure 2.5 depicts the main concepts of CIDOC CRM.

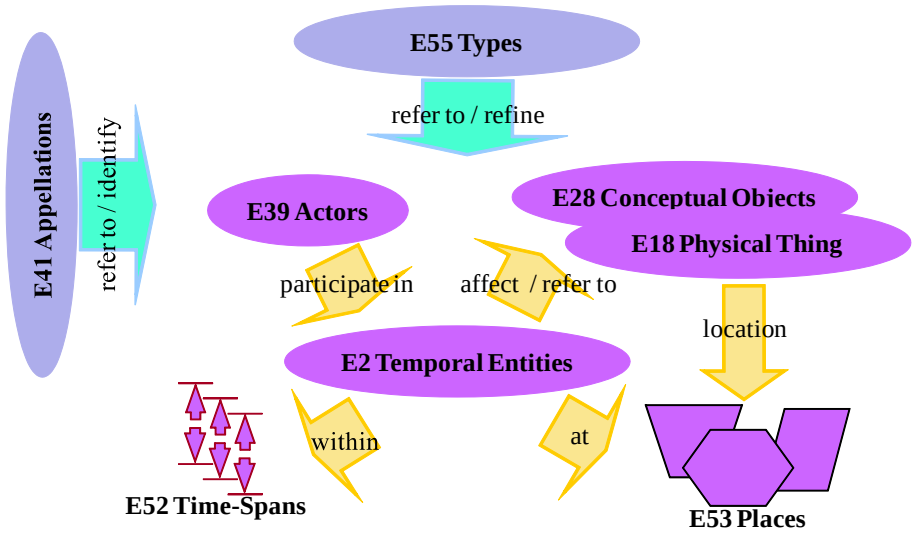


Figure 2.5: The main concepts of CIDOC CRM

CRMdig (Theodoridou *et al.*, 2010) was initially defined during the EU Project CASPAR[2] (FP6-2005-IST-033572) and its evolution continues in the context of the EU IST IP 3D-COFORM[3] project. In numbers, CIDOC CRM contains 86 classes and 137 properties, while its extension CRMdig currently contains 31 classes and 70 properties. Various mappings of CIDOC CRM are available at the the model's website[4].

They include mappings to EDM (Europeana Digital Library) which includes concepts from ORE and Dublin Core), LIDO model, UNIMARC Bibliographic for-

[2]http://www.casparpreserves.eu/
[3]http://www.3d-coform.eu/
[4]http://www.cidoc-crm.org/crm_mappings.html

mat, MIDAS Standard, Dublin Core Element Set, AMICO Data Model, EAD, and various others. Moreover the harmonization with FRBR is also discussed[5], as well at the mapping between CIDOC CRM and METS (M.Doerr, 2011).

Since Open Provenance Model (OPM) and CIDOC CRMdig have been mapped to several other models (see Fig. 2.6) we argue that they are good hubs for provenance models. This justifies the need for establishing mappings between them which has been suggested in (Salza *et al.*, 2012). Therefore, the similar concepts exploited in this thesis, might be used with OPM or/and other models.

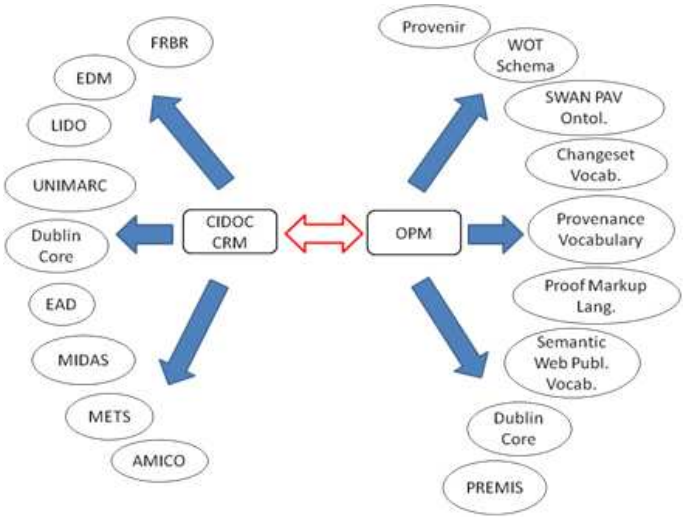


Figure 2.6: CIDOC CRM's and OPM's mappings

Fig. 2.7 shows one small part of the model, specifically the part involved in the inference rules introduced in Section 3.

The properties and classes shown in Fig. 2.7 are described in detail in CIDOC CRM's official definition.[6] In brief, the properties *P46 is composed of* and *P9 forms part of* represent the part-hood relationships of man-made objects (i.e., instances of the *E22 Man-made Object* class) and activities (i.e., instances of the *E7 Activity* class) respectively. Furthermore, the property *P14 carried out by* describes the active participation of actors (i.e. instances of the *E39 Actor* class) in activities and also implies casual or legal responsibility. Moreover, the *P16 was used for* describes the use of objects in a way essential to the performance of an activity. Finally, immaterial items (i.e., instances of the *E73 Information Object* class) are related to physical carriers (i.e., instances of the *E24 Physical Man-made Thing* class) via the *P128 carries* property and can be present at events via the *P12 was present at* property. Note that the properties *P46 is composed of* and *P9 forms part of* are transitive, reflexive and antisymmetric.

[5]http://www.cidoc-crm.org/docs/doer_le_boeuf.pdf
[6]http://www.cidoc-crm.org/docs/cidoc_crm_version_5.0.4.pdf

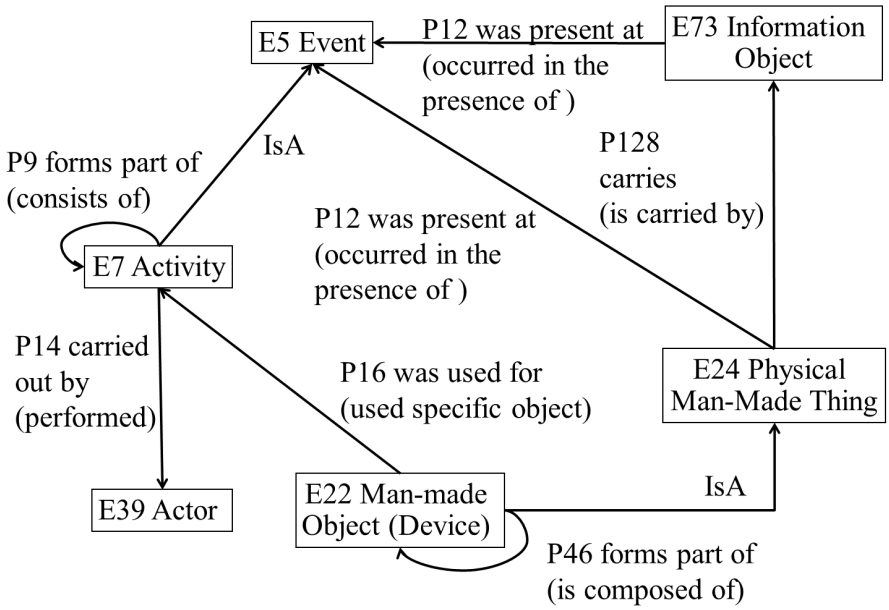


Figure 2.7: Part of CRMdig schema

2.3 Ontological and Semantic Assumptions

2.3.1 Participation and Presence at Events

According to OPM, a process is considered as the set of actions performed on or caused by artifacts and resulting in new artifacts. Processes are connected with artifacts using *used* and *wasGeneratedBy* edges. By connecting a process to several artifacts by used edges, we are not only stating the individual inputs to the process but also asserting that a causal dependency expressing that the process could take place and be complete only because all these artifacts were available. Therefore we cannot model the participation of artifacts in the derivation history of an object without implying causality. Entities modelled in CIDOC CRM can participate or denote their presence at events. The properties defined for this purpose are the *P14 carried out by* and *P12 was present at* respectively. The first one is a specialization of the second and describes the active participations of actors in activities. Note that this property implies causal or legal responsibility of the former. The second one is semantically more generic than the first one and describes not only the active but also the passive presence of entities at events without implying any specific role. For this reason, in our modelling we will consider that the actors associated to activities via the *P14 carried out by* association have a specific role for carrying out those activities. Whereas, immaterial and material objects have a passive participation at events and thus are associated with them via the *P12 present at* association.

2.3.2 Parts and Wholes

In this study we are considering reasoning forms in hierarchical structures. As depicted in Fig. 2.7 the main properties of CIDOC CRM for decomposing wholes into parts, be it for activities and subactivities or devices and parts, are the *P9 forms part of* and *P46 forms part of*. In this section, we will examine the definitions of parts and wholes that we will take into account in our reasoning.

The research area which analyses such part-hood relations known in philosophy is mereology, from the Greek word μέρος, part. More formally, mereology is "simply an attempt to set out the general principles underlying the relationships between a whole and its constituent parts" (Varzi, 1996). Its main focus is the relation expressed by the term *part of* (e.g., "X is part of Y").

Varzi (1996) gives an overview of the mereology theories regarding the study of the parthood relation. *Ground Mereology* was defined and considered as the basis of the aforementioned theories. The theory is defined by axioms using first-order theory with identity for the binary predicate P (representing the relation part-of):

- *Reflexivity*
  Everything is part of itself.
  $\forall x$ P(x,x).
- *Antisymmetry*
  Two distinct things cannot be part of each other.
  $\forall x, yP(x, y) \wedge P(y, x) \rightarrow x = y$.
- *Transitivity*
  Any part of any part of a thing is itself part of that thing.
  $\forall x, y, zP(x, y) \wedge P(y, z) \rightarrow P(x, z)$.

Other theories are defined adding further principles to the *Ground Mereology* such as the *Extensional Mereology* and the *Closed Mereology* (consult (Varzi, 1996) for a further analysis of these theories).

Nevertheless, several problems with the parthood relation have already been stated. Indicatively we mention the work of Artale *et al.* (1996). One of the most known is that of the assumption of transitivity (Johansson, 2004; Guizzardi, 2009) which does not hold in every case. One should consider the following counterexamples:

- A handle, x, can be part of a door, y, and a door can be part of a house, z, but yet the handle need not be (is not) a part of the house.
- This tree is part of the Black forest, and the Black forest is part of Germany, but yet this tree is not part of Germany.

However, in this thesis we will not attempt to find a new solution to the above problem and thus we assume that the parthood relation adheres to the axioms of the *Ground Mereology* including that of transitivity.

In our reasoning we will consider activities that can be decomposed into further subactivities (see Fig. 2.7). These activities can be also considered as events (since there is an IsA relation between the two classes), and thus events can be decomposed

into further subevents. Usually events extend to a specific timespan and that might be a condition in order for an event to be part of a much larger event. More specifically, the timespan of the latter must be larger and include the former's one. In this respect, the larger event is considered as the whole and the subevent as the part. For example, supposing that the World War II is an event lasting from 1939 to 1945, the multiple battles during this time are the subevents of the former. The same assumption and reasoning will also be followed in the case of the activities and subactivities.

With regard to devices as wholes, these can be considered as a heterogeneous complexes having different structured parts that have a functional role. This theory has been suggested by Gerstl & Pribbenow (1995) defining a the following categories of parts and wholes:

- Component/Complex
- Member/Collection
- Quantity/Mass

The above classification has been defined in accordance to the compositional structure of wholes: homogeneous, uniform and heterogeneous (Gerstl & Pribbenow, 1995, 1996). Homogeneous wholes have no compositional structure and their decomposition is based on a quantitative measure such as weight or volume (e.g., one litre of the water in a bottle). While heterogeneous have different structured parts having several roles with respect to the whole (e.g.,the engine of the car which can have multiple different parts). Uniform wholes are composed by parts that "are not distinguished according to the way they relate to the whole, although they might be distinguished with respect to each other" (e.g, two of the three apples in the basket) (Gerstl & Pribbenow, 1995).

Our modelling will be limited to heterogeneous physical objects. So wholes conceptualized as masses or collections are out of context. Thus, hereafter, we will consider that parts of a device-whole have solely a functional role type and as a result the whole is assumed to be a functional complex. According to Gerstl & Pribbenow components have "a specific relation to the whole different from that of the other components with respect to functional, spatial, temporal, or other features" (i.e., for a car such a functional part is considered to be one of its four wheels). Table 2.1 presents some examples of wholes and their functional parts.

Whole	Part
a cellphone	its battery
a computer	its screen
a camera	its lens
a car	its engine

Table 2.1: Examples of objects and their parts

In general, parts (be it for subactivities or parts of devices) can in turn be considered as wholes themselves and subdivided further into other parts. The result

is an hierarchy of components forming a partonomy (Tversky, 1986) (see Fig. 2.8 for an example of a partonomy).

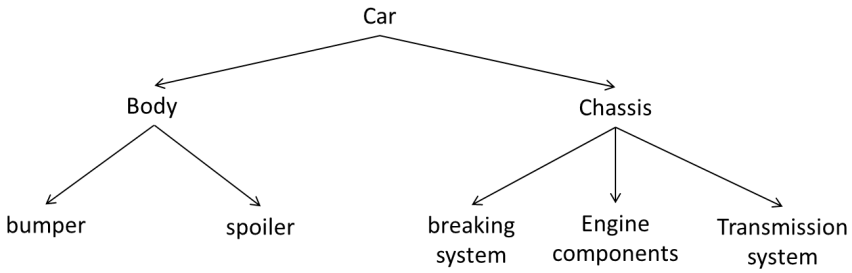


Figure 2.8: Partonomy of a car

2.4 Working Assumptions

We shall use the term KB to refer to a Knowledge Base in the logical sense, either stored in a system or composed by the contents of several metadata files. We shall use K to refer to the set of RDF/S triples of a KB. Furthermore, hereafter we assume that K denotes the set of triples as produced by the adopted metadata schemas and ontologies and the *ingested* facts ("raw data" yielded by manual or automated processes) without any post-processing.

We shall use *C(K)* to refer to the *closure* of K. Note that the closure can be defined with respect to the standard inference rules for RDF/S and/or other custom rules (e.g., like those that we introduce in this paper) and is unique. From a visual perspective, in the following examples of our rules (Section 3)*C(K)* includes both the dotted and plain associations, while K includes only the latter one. We shall also use *Red(K)* to denote the *reduction* of K. The reduction is the minimal set of facts that have the same closure as K, and it is unique if and only if the relations are acyclic (see (Zeginis *et al.*, 2011)).

For example considering the example of the car partonomy in Fig. 2.8 and that the relation among the components is a transitive one, the transitive closure of the graph is depicted in Fig. 2.9. Note that the inferred relations are denoted by dotted lines.

The elements of K is our *explicit* knowledge, while the elements of *C(K)-K* represent our *implicit/inferred* one.

A repository policy could be to keep stored either K, or *Red(K)*, or *C(K)*. Storing *Red(K)* would give optimum (i.e., minimal) space usage, but would imply an important overhead during changes, because the reduction should be recalculated after every change and its algorithmic complexity is $O(N^3)$. Moreover, in some settings, K has a different value from *C(K)* and the distinction between the two must be kept clear. On the other hand, *C(K)* is optimal with respect to efficiency, because all the information is stored and can be easily found, but has increased space requirements. For this reason, for the purposes of this paper we chose to store K as a reasonable

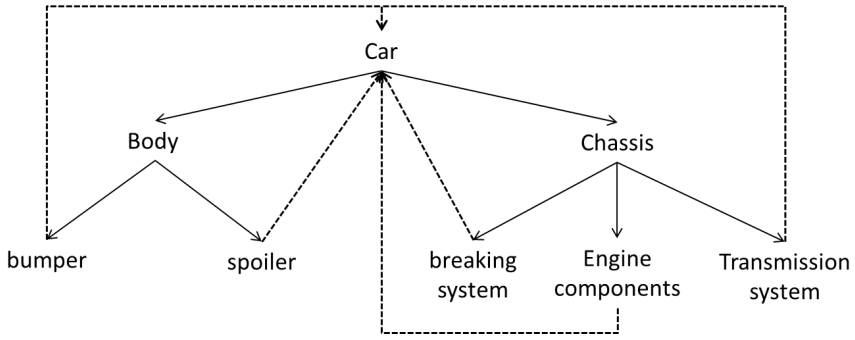


Figure 2.9: The transitive closure of a partonomy graph

compromise in this time-space trade off. Note that K usually contains little or no redundancy, leading to a near-optimal space usage while avoiding the overhead of searching for, and eliminating, redundancy.

2.5 Summary

In this Chapter we presented the assumptions and definitions adopted by this study. Firstly, we explained how we represent our knowledge and we addressed some considerations with its nature and scope (i.e., its epistemology). Secondly, we elaborated on the question on how provenance information can be modelled. We focused on the most popular model in this regard (i.e., OPM) and we compared to the one used as our application context. Thirdly, we reviewed some basic semantic concepts and assumptions involved in our reasoning, and finally we briefly introduced our working definitions for the rest of this paper.

Chapter 3

Provenance Inference Rules

In this Chapter we introduce a basic set of custom inference rules in order to accomplish our initial purpose for reducing the space storage requirements of provenance information. Thus, each rule is intended to infer dynamically certain relations among entities of CRMdig that otherwise had to be added by the ingestion process. The basic idea behind this reasoning is the propagation of features and attributes from parts to wholes. The rules chosen involve concepts that exist in almost all models for representing provenance information and thus reflect the generalization of this study. The inferred relations and the general reasoning respect the defined semantics of the CIDOC CRM ontology. One can define a much larger set of rules with regard to a particular underlying ontology, but that would result to a higher reasoning complexity and a more complicated update operations (see also Section 8.1 for a discussion concerning these matters).

Each Section in this Chapter presents the rules accompanied by examples of the cultural heritage domain. The rules are also encoded into first-order logic (FOL) as a general basis for further implementations to other languages such as SWRL or Datalog. All the relations and entities are instances of the small schema adopted in Fig. 2.7. Note that in the included figures we do not show the transitivity-induced properties *P46 is composed of* and *P9 forms part of*.

3.1 Participation of actors to activities

- **Rule 1. Participation of actors to activities**
  If an actor has carried out one activity, then he has carried out all of its subactivities.

$$\forall x, y, z(formsPartOf(y, x) \wedge carriedOutBy(x, z) \rightarrow carriedOutBy(y, z)) \tag{3.1}$$

Example

Scientists often use 3D laser scanning in order to construct digital 3D models. These processes involve taking many photographs of the desired model. One example from our data is the activity *Laser scanning acquisition of Canoe-shaped vase from the*

Archaeological Museum of Nicosia which was carried out by the *STARC-The Cyprus Institute*. The activity has its own subactivities. It can be analysed to *Sequence of shots - Canoe-shaped vase from Archaeological Museum of Nicosia* and that can have the following subactivities, ranging, for example, from one to ten: *Capture 1* to *Capture 10* which store information about each captured photo.

All the above subactivities have different recorded metadata. On the other hand, the information that the *Starc Institute* and *John* were responsible for carrying out the *Laser scanning* activity is desired to be preserved following the path of all the subactivities. Fig. 3.1 shows the edges (represented by dotted lines) which are inferred by the rule R1.

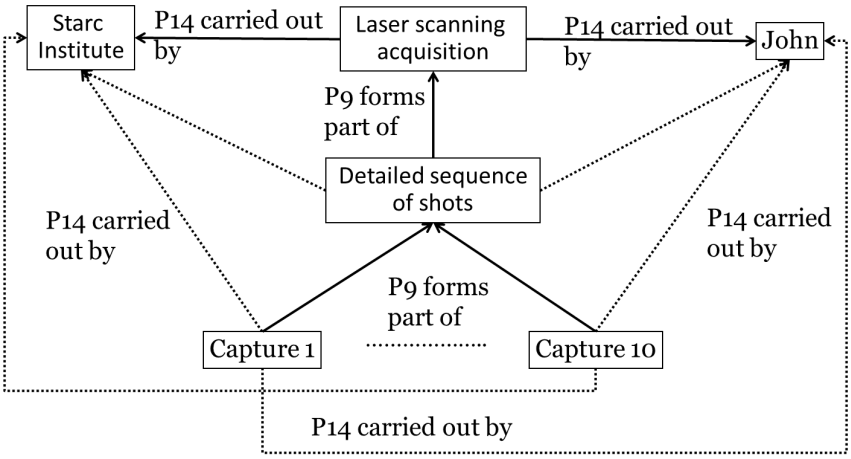


Figure 3.1: Example of rule R1

3.2 Use of objects and their parts

- **Rule 2. Use of Objects to activities**
  If an object (device) was used for an activity, then all parts of the object were used for that activity too.

$$\forall x, y, z(isComposedBy(x, y) \wedge wasUsedFor(x, z) \rightarrow wasUsedFor(y, z)) \tag{3.2}$$

Example

In 3D modelling, devices with many cameras, called multiviewdome devices (see Fig. 3.2), are usually employed. These devices are consisted of other cameras or lighting devices. In provenance metadata, there could be a fact stating that a multiviewdome device was used for a particular activity.

With R2 we can infer that the constituent devices were also used for that activity. Indeed, a multiviewdome device cannot be used without using its parts. Fig. 3.3

Figure 3.2: A multiview dome device

illustrates an indicative modelling of such a setting. The parts for example *Nikon D90*, *AF-5 Nikkor 18-105* and *Nikon D300* of the device have been associated to the *Detailed sequence of shots*.

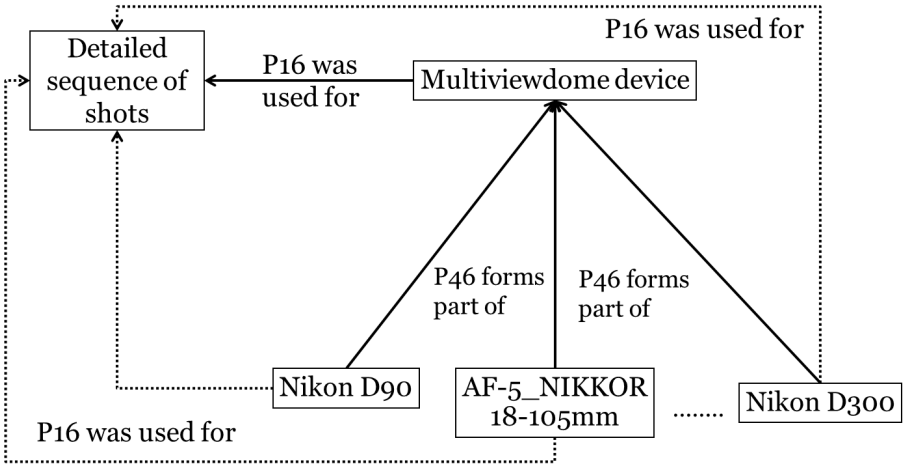


Figure 3.3: Example of rule R2

3.3 Presence of information objects

- **Rule 3. Presence of Information Objects**
  If a physical thing that carries an information object was present at an event, then that information object was present at that event too.

$$\forall x, y, z(carries(x, y) \wedge wasPresentAt(x, z) \rightarrow wasPresentAt(y, z)) \tag{3.3}$$

Example

3D reconstruction from images is a common process used in archaeology in order to document, digitize and model archaeological exhibits such as statues. Consider for example the exhibit shown in the next figure which is part of a column of Ramesses II located in the Egyptian museum garden in Cairo. The 3D reconstruction process of said exhibit could be modelled as an event and the exhibit itself as a physical man-made thing. Moreover that physical thing contains information which is represented by the carved hieroglyphics.

That information could be modelled as an information object. According to rule R3 if that part was present at an event, then that information was also present at that event. Rule R3 infers the presence of information in hieroglyphics at the event of a 3D reconstruction because the part of Ramesses II was also present at that event. The inference is reasonable because the information was carved in hieroglyphics when the column was built, thus the information in hieroglyphics coexists with the part of a column which carries it and this coexistence implies their presence at events. As a result, if there is a *carries* relationship between an information object and a physical thing, rule R3 infers the presence of the former in all the events that the latter was present at.

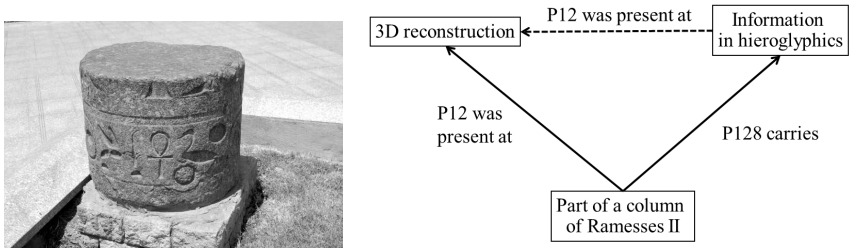


Figure 3.4: Part of a column of Ramesses II (left) example of rule R3 (right)

3.4 Summary

In this Chapter we presented a set of inference rules in order to satisfy our primary motivation of this study regarding the storage requirements of provenance information. We focused on these three rules as they frequently occur in practice. Of course, one could extend this set according to the details and conventions of the application at hand (see also Section 6.1). Table 3.1 shows an overview of the introduced inference rules accompanied by brief examples.

Table 3.1: Summary of the inference rules

Rule Num	Rule Name	Rule Description	Brief Example
R1	ActorCarried Activity	If an actor has carried out one activity, then he has carried out all of its sub activities.	"STARC-The Cyprus Institute" is the actor of a Laser scanning acquisition activity but also the actor of the detailed sequence of shots which are actually sub activities of the scanning acquisition activity.
R2	UsedPartof Object	If an object (device) was used for an activity, then all parts of that object were also used for that activity.	If a multidome camera was used for an event, then a lens of it was also used for that event.
R3	Information ObjectPresence	If a physical object (that carries the information object) was present at that event, then that information object was also present at that event. (note that an information object must have a physical object that carries that, otherwise it cannot exist).	If we know that a person X1 read a poem Y1 during an event E1, then certainly there was a carrier Z for that poem in that event. If a device was present at an event and uses a piece of software then this software was also present at that event.

Chapter 4

Provenance Inference Rules and Knowledge Evolution

A certain set of beliefs or knowledge, as the world itself is generally not static; it evolves over time. Possible causes of the alteration of one's beliefs could be that new, previously unknown, classified, or otherwise unavailable information may have become known; a new fact may have been revealed through a new observation or experiment. The research area which deals with the adaptation of a KB to *new information* is the area of the *belief change* (Gärdenfors, 1992). In this Chapter we tackle the challenge of how our knowledge can be changed by taking into account the inference rules introduced in Chapter 3.

4.1 Preliminaries

One of the purposes of this study is to examine the application of inference mechanisms in order to automate the derivation of implicit facts among the existing ones stored in a KB. Such mechanisms (see the introduced inference rules in Chapter 3) can be used in a metadata environment accompanied by an inference platform. In this regard, we assume an a priori knowledge of the latter's inference capabilities before the ingestion of new metadata. This concept is much related to the homogeneous approach which integrates both ontology and rules into a common logical language (Antoniou *et al.*, 2005). By this approach, the user/process that is responsible for the ingestion is expected to know this model and, as a consequence, avoid any redundant facts which can be inferred dynamically by the rules.

Be that as it may, the possibility of human errors should not be ignored. These errors are propagated by the rules among data affecting directly the quality of provenance. The user must be able to update/change the stored knowledge in order to correct such errors and thus the support for updates is unquestionable.

On the other hand, satisfying update requests while still supporting the aforementioned inference rules is a challenging issue, because several problems arise when we update knowledge taking into account rules and implicit facts. For example, should we remove an implicit fact from our KB or we should keep it to avoid any possible loss of information? Moreover, when changes are performed upon a belief

base, we temporarily have to ignore the logical consequences of the base; one can only apply direct changes to knowledge that is stored explicitly. On the other hand, implicit knowledge cannot be changed directly (though it could be indirectly affected by the changes in the explicit knowledge). In this regard, the challenge is the determination of *what* can be changed and *how* each change should be implemented based on the defined semantics.

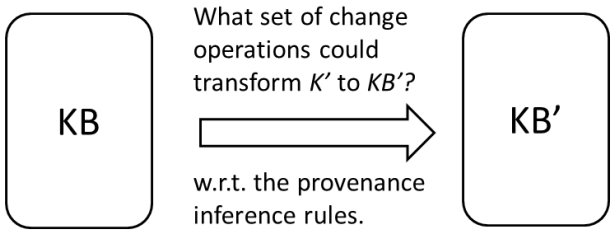


Figure 4.1: What set of change operations could transform KB to KB'

These issues will be described in more detail below using a running example. For each change operation, we describe the KB's states (through figures) and explain the challenges incurred in the update process due to the existence of the inference rules.

4.1.1 Changing our beliefs: Expansion and Contraction

In (Alchourrón *et al.*, 1985) the authors identify three different types of belief change. The simplest one, is expansion. Expansion refers to the naïve addition of information to our knowledge; reckless application without taking any special provisions to ensure the quality (i.e., consistency) of the KB after this addition. This operation is implemented trivially as a set-theoretical union of the new information (i.e., the change) and our beliefs (old KB). In the case that the new information contradicts the currently held beliefs the result will be an inconsistent KB. In this matter we apply the additional operation (discussed later in this Section) involving the removal of information in order for our KB to be consistent.

The second one is the revision operation is defined. Similar to expansion, revision has the very important difference that demands a certain quality on the results; the result should be a consistent set of beliefs. There are several ways to achieve this property on the resulting beliefs. When contradicting old knowledge one could choose to reject the change. Hence most of the times, one chooses to apply the changes upon the beliefs, facing however, possible inconsistency problems. These problems could be overcome with potential additional application of consequential changes, in order to retain the quality of the result. In some cases for example, one may need to abandon a part of the currently held beliefs while adding the new information. The difficult and interesting part of revision is how to select the beliefs that should be abandoned.

A fundamental operation and, consequently, the most important operation for theoretical purposes is contraction (Gärdenfors, 1992). Contraction corresponds

to the removal of information from a KB, but in a consistent way. For example, when a piece of information becomes unreliable and we would like to stop believing it, contraction may be necessary. A property of a contraction operator is that it should retract the unreliable information from both explicit and implicit knowledge; the latter could re-emerge as a consequence of the remaining beliefs, so simply removing the information from explicit knowledge may not be enough. Hence, a contraction operator may need to also remove beliefs which at first seem unrelated to the retracted piece of knowledge. Our suggested operations will be based on these three different types of belief change.

4.1.2 The Principles of Knowledge Revision

In this paper, we adhere to the most significant principles of Knowledge Revision theory as motivated by Dalal (1988). On of these principles is referred to as the Principle of *Primacy of New Information* and is related to the acceptance of the new information. This involves the complete entrustment to the incoming data and the unconditionally acceptance of the new information. Thus, the resulting KB should contain the new information (in case of a revision, for example) or should not contain it(e.g, in case of a contraction).

Another principle of Dalal (1988) is *Principle of Consistency Maintenance*. According to that principle the result of a change should be a consistent KB. This is generally accepted as we have already stated that inconsistent (under classical logic) KBs do not carry any interesting information and, therefore, should be avoided. Whereas for classical logic this principle is valid, there are frameworks in which the underlying logic itself has an internal mechanism to deal with inconsistencies, such as nonmonotonic and paraconsistent KBs (Antoniou, 1997). One thing that remains to be settled is the exact meaning of the "consistency" term; in terms of FOL we consider inconsistent any set that contains or implies both a proposition and its negation. More generally in the belief change literature, the meaning of a consistent KB is one which does not imply any tautologically false propositions.

Undebatably, the most important, and by so far the most influential principle related to the implementation of a change, is the *Principle of Persistence of Prior Knowledge* (Dalal, 1988). This principle argues that the resulting KB should retain most of the information from the old KB.

4.1.3 Foundational versus Coherence Theories

Knowledge Representation in literature has also been considered by several related philosophical studies. Mainly, two viewpoints have been discussed: foundational theories and coherence theories. We argue that this is an important distinction and thus we take them into account when dealing with deletion of knowledge.

Under the foundational viewpoint, each piece of our knowledge serves as a justification for other beliefs; our knowledge is like a pyramid, in which "*every belief rests on stable and secure foundations whose identity and security does not derive from the upper stories or sections*" (Sosa, 1980). This viewpoint implies that the ingested facts (the "base of the pyramid") are more important than other knowledge and that

implicit knowledge has no value of its own, but is depending on the existence and support of the explicit knowledge that caused its inference.

On the other hand, according to the coherence theory, our beliefs do not require any justification. A belief is justified by how well it fits with the rest of the beliefs, by how well it fits with the rest of the knowledge, in forming a coherent set of facts that contains no contradictions. In this sense, knowledge is like a raft, "*every plank of which helps directly or indirectly to keep all the others in place, and no plank of which would retain its status with no help from the others*" (Sosa, 1980). This means that all knowledge (implicit or explicit) have the same "value" and that every piece of knowledge (including implicit ones) is self-justified and needs no support from explicit knowledge.

4.1.4 Elementary and composite changes

According to (Maedche *et al.*, 2002; Stojanovic & Motik, 2002), change operations can be classified into elementary (involving a change in a single ontology construct) and composite ones (involving changes in multiple constructs), also called atomic and complex in (Stuckenschmidt & Klein, 2003). Elementary changes represent simple, fine-grained changes; composite changes represent more coarse-grained changes and can be replaced by a series of elementary changes. Even though possible, it is not generally appropriate to use a series of elementary changes to replace a composite one, as this might cause undesirable side-effects (Maedche *et al.*, 2002); the proper level of granularity should be identified in each case. Examples of elementary changes are the addition and deletion of elements (concepts, properties etc) from the ontology. There is no general consensus in the literature on the type and number of composite changes that are necessary. In (Maedche *et al.*, 2002), 12 different composite changes are identified; in (Stuckenschmidt & Klein, 2003) however, the authors mention that they have identified 120 different interesting composite operations and that the list is still growing!

In this work, we focus on three elementary change operations allowing to transform one KB to another, namely triple Addition(t), Disassociation(t), Contraction(t) and a composite one Replacement(t) where $t \subset T$. All the three operations are related to the acquirement of new knowledge and how this affects the existing knowledge which is stored in the current KB. Therefore, sometimes the new knowledge does not contradict the existing one (i.e., Addition) and other times does the opposite and thus some facts have to be removed in order for the KB to be consistent. In the sequel, we will formally introduce the semantics of these operations and give examples for each one. Even though we specify only one composite operation other similar ones can be defined by defined as a series of elementary operations similar to the Replacement.

4.2 Provenance Inference Rules and Knowledge Evolution

A KB changes over time, i.e., we may have requests for adding or deleting facts due to external factors (Flouris *et al.*, 2012) such as new observations. Satisfying

update requests while still supporting the aforementioned inference rules is a challenging issue, because several problems arise when updating knowledge taking into account rules and implicit facts. These issues will be described in more detail below using a running example. For each change operation, we describe the KB's states (through figures) and explain the challenges incurred in the update process due to the existence of the inference rules.

Consider a KB that contains the rule R1's example with the activities of *Laser scanning acquisition* that were carried out by the *Starc Institute*. The initial state of the KB is demonstrated in Fig. 4.2 (left), where inferred associations are illustrated by dotted lines. Over this example below we shall see examples of three change operations: *addition, disassociation, contraction*.

4.2.1 Addition of Information

The *addition* operation performs the insertion of a new triple into the KB and since there are no validity rules or negation, the addition of a triple cannot lead to a contradiction (as, e.g., in the case of (Flouris *et al.*, 2012) or in the generic belief revision literature (Gärdenfors, 1988)). Therefore, addition is a simple operation consisting only of the straightforward addition of the required information (triple) in our KB. Note that if we had taken into account a repository policy that stores the redundant-free KB, or its closure, we would have to apply additional operations after the addition in order to compute and store *Red(K)* (or *C(K)*).

Suppose a request for adding a new actor to the subactivity of the *Sequence of shots*, e.g., that *Michael*, who is a photographer, is also the actor of *Sequence of Shots*. The update is demonstrated in the following figures. We observe that *Michael* has been associated with the activity *Sequence of shots*, but also, due to rule R1, he has been associated with the subactivities *Capture 1_7* and *Capture 1_8*. Fig. 4.2 depicts the change of this example.

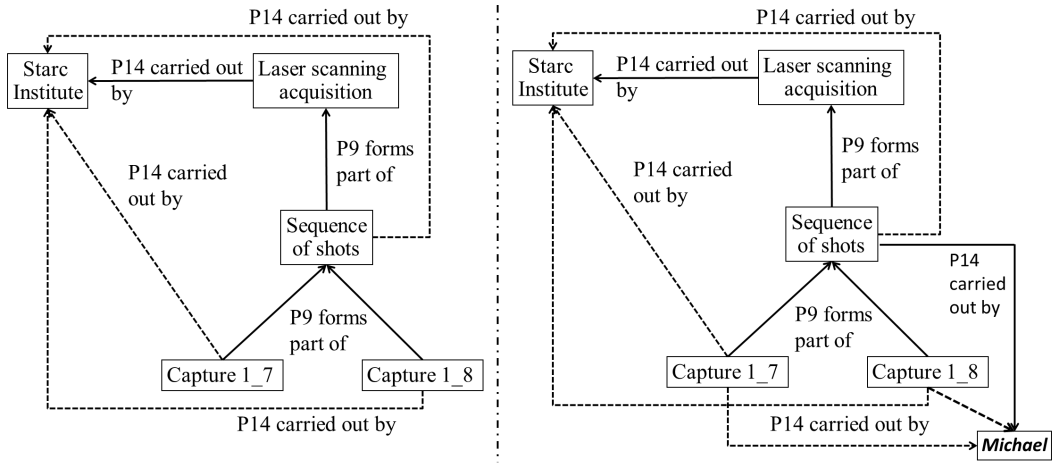


Figure 4.2: Initial state of the KB (left) and state of the KB after the addition (right)

4.2.2 Deletion of Information

The operation of deletion is more complicated than addition in our setting. The reason is that, due to the inference rules, we cannot succeed by simply deleting the required information, as the deleted information may re-emerge as a consequence of the application of the inference rules. Thus, it is often the case that additional information should be deleted, along with the one explicitly requested by the change. This raises the additional challenge of avoiding losing (deleting) any more knowledge than necessary. We will visualize how we address this problem using our running example below.

Consider an update request saying that the *Starc Institute* is not responsible for the activity *Capture 1_7*. The question raised is whether the *Starc Institute* is not responsible only for *Capture 1_7*, or also for other activities, i.e., how refuting the fact that the *Starc Institute* is responsible for *Capture 1_7* affects the information that it is responsible for the activities *Laser scanning acquisition, Sequence of shots* or *Capture 1_8*.

Initially, we note that the *Starc Institute* should also be disassociated from the responsibility of *Sequence of shots* and *Laser scanning acquisition*; failing to do so would cause the subsequent re-emergence of the refuted knowledge (i.e., that the *Starc Institute* is not responsible for *Capture 1_7*) due to inference.

A more complicated issue is whether the *Starc Institute* should remain responsible for *Capture 1_8*: note that this information was originally included because of the fact that the *Starc Institute* was considered responsible for Laser scanning acquisition, ergo (due to the inference rules), also responsible for *Capture 1_8*. Once the former information is dropped, as discussed above, it is questionable whether the latter (inferred) information should still remain in the KB, since its "reason for existence" is no longer there. On the other hand, the fact that the *Starc Institute* is not responsible for *Capture 1_7* does not in any way exclude the possibility that it is still responsible for *Capture 1_8*, therefore deleting this information seems like an unnecessary loss of knowledge.

To address this issue, one should go deeper and study the related philosophical issues regarding the epistemological status of the inferred knowledge, and whether such knowledge has the same or different value compared to primary, explicitly provided knowledge (i.e., ingested knowledge), (see Section 4.1.3).

This distinction is vital for effective management of data deletions. When a piece of knowledge is deleted, all implicit data that is no longer supported must be deleted as well under the foundational viewpoint. In our example, this should cause the deletion of the fact that the *Starc Institute* is responsible for *Capture 1_8*. On the other hand, the coherence viewpoint will only delete implicit data if it contradicts with existing knowledge, because the notion of support is not relevant for the coherence model. Therefore, in our case, the fact that the *Starc Institute* is responsible for *Capture 1_8* should persist, because it does not in any way contradict the rest of our knowledge, nor does it cause the re-emergence of the newly deleted information (i.e., that the *Starc Institute* is responsible for *Capture 1_7*.

Instead of positioning ourselves in favour of one or the other approach, we decided to support both. This is done by defining two different "deletion" operations, namely,

disassociation and *contraction*, that allow us to support both viewpoints. Their definition follows.

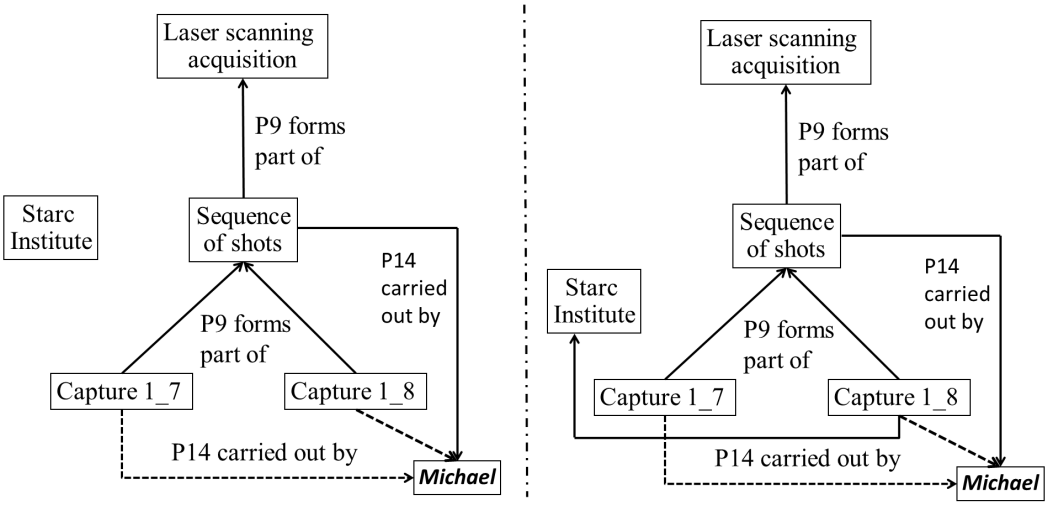


Figure 4.3: Actor disassociation (left) and actor contraction (right)

We will be referring to the above cases as *actor disassociation* and *actor contraction* respectively. Notice that in contrast to disassociation, the contraction operation preserves the association (Capture 1_8, carried out by, Starc Institute).

Disassociation

Disassociation handles deletion using the foundational viewpoint. In particular, the non-responsibility of the *Starc Institute* about *Capture 1_7* implies some uncertainty about its responsibility for other related activities (i.e., was he responsible for *Capture 1_8*?), since this knowledge is no longer supported by any explicit data. Based on the foundational viewpoint, all such associations must also be deleted, i.e., we should delete the following triples:

- (Capture 1_7, carried out by, Starc Institute) // as requested
- (Capture 1_8, carried out by, Starc Institute) //due to the loss of explicit support
- (Sequence of shots, carried out by, Starc Institute) //due to the loss of explicit support
- (Laser scanning acquisition, carried out by, Starc Institute) //to avoid re-emergence of the deleted knowledge

Note that, in practice, implicit facts are not stored so they do not need to be deleted; thus, in our case, we only need to delete (Laser scanning acquisition, carried out by, Starc Institute).

Contraction

Contraction handles deletion using the coherence viewpoint. In particular, this operation assumes that there is a high degree of certainty that the non-responsibility of the *Starc Institute* is only for *Capture 1_7*. Other activities which are still associated with *Starc Institute*, such as *Capture 1_8*, should persist despite the lack of explicit knowledge to support them. In this case we have to delete only the following triples:

- (Capture 1_7, carried out by, Starc Institute) // as requested
- (Sequence of shots, carried out by, Starc Institute) //due to the loss of explicit support
- (Laser scanning acquisition, carried out by, Starc Institute) //to avoid re-emergence of the deleted knowledge

Again, implicit facts do not need to be deleted, so the only actual deletion required is the deletion of (Laser scanning acquisition, carried out by, Starc Institute). For contraction, one should also be careful with the implicit knowledge that is supposed to persist. For example, the fact that the *Starc Institute* is responsible for *Capture 1_8*, is not explicitly stored and will be lost by the deletion of (Laser scanning acquisition, carried out by, Starc Institute) unless we explicitly add it back.

Since we focus on three inference rules, we should define similar operations for satisfying change requests related to the other custom inference rules, i.e., R2 and R3. The complete set of operations is given in the Appendix of this paper.

4.3 Algorithmic Perspective

The analysis like that of the previous example can be applied for deriving the exact change plan for each change operation (the semantics of the operations are given in the appendix this paper). Indicatively, we provide update plans (algorithms) for three operations below:

- AssociateActorToActivity (p:Actor, a:Activity)
- DisassociateActorFromActivity (p:Actor, a:Activity)
- ContractActorFromActivity (p:Actor, a:Activity)

Algorithm 1 AssociateActorToActivity (p:Actor, a:Activity)

```
1: if an explicit P14 link does not exist between a and p then
2:     Add an explicit P14 link between a and p
3: end if
```

Algorithm 1 takes as input an actor (p) and an activity (a), checks if the information that p is responsible for a already exists in the KB (line 1) and, if not,

Algorithm 2 DisassociateActorFromActivity (p:Actor, a:Activity)

```
1: if an explicit P14 link exists between a and p then
2:     Remove the requested P14 link between a and p
3: end if
4: for each superactivity:superAct of a related to p via the P14 link do
5:     Remove possible explicit P14 link between superAct and p
6: end for
```

Algorithm 3 ContractActorFromActivity (p:Actor, a:Activity)

```
1: if an explicit "carried out by" link exists between a or a superactivity of a and
   p then
2:     for each direct subactivity:subAct of a do
3:         Execute AssociateActorToActivity (p, subAct)
4:     end for
5: end if
6: if an explicit "carried out by" link exists between a and p then
7:     Remove the requested "carried out by" link between a and p
8: end if
9: for each maximal superactivity:supAct of a related to p via the "carried out
   by" link do
10:    for each subactivity:subAct of supAct do
11:        if subAct is not superactivity or subactivity of a then
12:            Add subAct to collection: Col
13:        end if
14:    end for
15: end for
16: Execute DisassociateActorFromActivity (p, a)
17: for each maximal activity:act in Col do
18:    Execute AssociateActorToActivity (p, act)
19: end for
```

it adds that fact as an explicit one in the KB (line 2). This is an easy operation, requiring just the addition of said triple in K.

Algorithm 2 takes the same input (actor p and activity a), but its purpose is to disassociate p from the responsibility for a. Firstly, the requested explicit association has to be removed from the KB (lines 1-3). Secondly, according to the semantics given above, this also requires the deletion of all associations of p with all superactivities of a (lines 4-6). Note that only explicit links need to be removed, because implicit ones do not actually exist in K.

Finally, Algorithm 3 contracts p from the responsibility for a. This requires, apart from the deletion of all associations of p with all superactivities of a (as in disassociation), the preservation of certain implicit associations that would otherwise be lost. At first, any inferred associations between p and any subactivities of a must be explicitly added to the KB (lines 1-5). Moreover, additional associations that need to be preserved are stored in Col (lines 9-15); to avoid adding redundant

associations, we only consider the maximal elements of Col to add the new explicit associations (lines 17-19).

We should clarify that each operation is independent in the sense that after its execution there is no need for other operations to be performed in order to complete a particular change. However, their design is modular and thus different operations may use the same algorithms or parts of them.

Furthermore, our operations guarantee that the resulting KB will not contain the deleted triple, either as an explicit or as an implicit fact, given the existing knowledge and the custom inference rules that we consider. In addition, our operations preserve as much as possible of the knowledge in the updated KB under the considered semantics (foundational/coherence for disassociation/contraction respectively). The two observations have been coined as general principles in the belief revision literature (Dalal, 1988).

4.3.1 Algorithmic Complexity

The complexity of the above algorithms is $O(logN)$ for Algorithm 1, $O(NlogN)$ for Algorithm 2 and $O(N^2)$ for Algorithm 3, where N is the number of triples in K. The above complexities assume that the triples in K are originally sorted (in a preprocessing phase); such a sorting costs $O(NlogN)$. Under this assumption, Algorithm 1 practically needs to make one addition to a sorted table, thus the $O(logN)$ cost.

Algorithm 2 requires the computation of all the superactivities of an activity. To do that, we need to find all the direct superactivities of a (i.e., those connected to a via the *P9 forms part of* property), a process which costs $O(logN)$; for each such activity, one needs to add it in a sorted collection that contains all the superactivities of a found so far, costing $O(logM)$, where M is the size of the collection. M can never exceed N, so the total computation time for finding one superactivity of a and adding it in our list is $O(logN)$. Continuing this process recursively, we will eventually add all superactivities of a, which are at most N, so the process can be repeated at most N times, costing a total of $O(NlogN)$. For each superactivity, we need to determine whether a *P14 carried out by* link to p exists, and, if so, delete it; this costs $O(logN)$ for each superactivity, i.e., $O(NlogN)$ in total (as we can have at most N superactivities). Thus, the combined computational cost for Algorithm 2 is $O(NlogN)$.

Algorithm 3 is more complicated. As in disassociation, we first need to find all superactivities of a, which costs $O(NlogN)$. Then, we need to find the maximal superactivities; this process requires a filtering over the set of all superactivities. In the worst-case scenario, this requires checking each superactivity against all others, costing $O(M^2)$ if the total number of superactivities are M. Thus, in the worst-case scenario (where M=N), the cost of finding the maximal superactivities is $O(N^2)$. After that, we need to compute Col (line 7), which is a process identical to the computation of superactivities and requires $O(NlogN)$ time. Finding the maximal elements in Col likewise requires $O(N^2)$, whereas the execution of the disassociation operation is $O(NlogN)$ as explained above. Summing up the above complexities, we conclude that the worst-case computational complexity for Algorithm 3 is $O(N^2)$.

4.3.2 Supplementary Operations

The algorithms for the operations related to the other inference rules, e.g., for DisassociateActivityFromMMObject and ContractActivityFromMMObject (related to R2), can be designed analogously. Since hierarchies of devices and parts, can be of the same graph morphology, the respective algorithms follow the same design as the ones of R1 discussed previously. However, some adjustments are essential such as the replacement of the *P14 carried out by* with the *P16 was used for* link and the activities with devices (i.e. man-made objects) which are composed of parts instead of subactivities. Moreover, the actor now is the activity that used the specific device. The operations for rule R3, since they are not applied on hierarchies, are not so complicated and thus their algorithms can be easily implemented. For instance, taking into account the foundational viewpoint, deletion of a *P12 was present at* link between an information object and event requires the deletion of the respective *P12 was present at* links between the carriers of the former and the latter.

The complete set of operations is given in the appendix of this paper. Using the same mindset, we could also develop algorithms for adding/deleting the transitive relationships used in our model, such as *P9 forms part of* ,
textitP46 is composed of etc, as well as for adding/deleting new objects, such as actors, activities etc.

In addition, one could compose the above operations to define more complex, composite ones. For instance, *Addition* and *Contraction* can be composed to define a *Replace* operation. Such an operation would be useful if, e.g., we acquire the information that *John* (another photographer) is responsible for *Capture 1_7* instead of the *Starc Institute*. This means that the *Starc Institute* should be replaced by *John*. Fig.4.4 illustrates a possible definition of replacement as a composition of an addition and a contraction. Another version of *replacement* could be formed by composing Add and Disassociate. One could similarly define more composite operations, but this exercise is beyond the scope of this work.

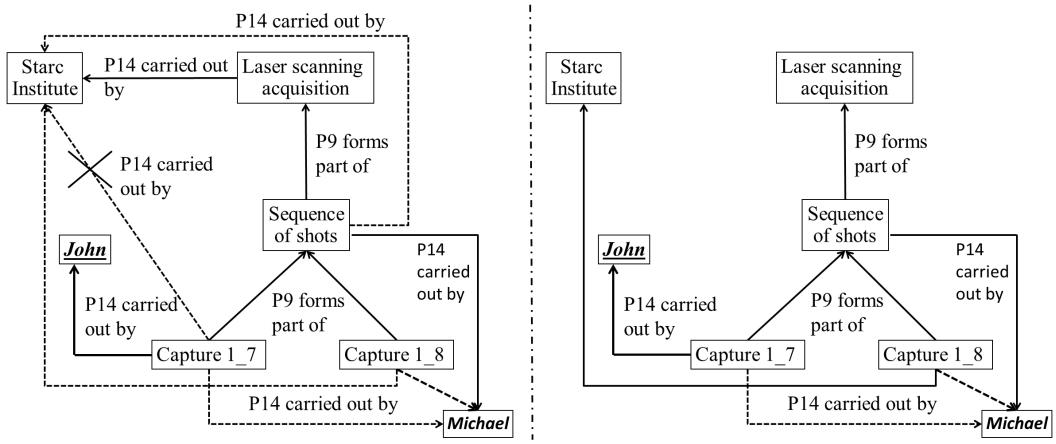


Figure 4.4: Actor replacement

Chapter 5

Implementation and Experimental Evaluation

This Chapter analyses some of the implementation issues of this study. It specifically details how our approach can be implemented (Section 5.1), what are the requirements (Section 5.1.2) and discusses the available repository policies with regard to the inference rules. Lastly, we present results of conducted experiments (Section 5.2) comparing two strategies of whether choose to store, or not the inferences with respect to the query performance, the storage space requirements and the performance of update operations. Even though the experiments were based on the *partOf* hierarchies of the *P9 forms partOf* relation of the CIDOC CRM ontology, we argue that they can also be indicative for other hierarchies, such as the RDF/S *subclassOf* one and for other rules of the same conjunctive form.

5.1 Implementation Considerations

5.1.1 On Closures and Reduction

As mentioned in Section 2.4, we use the term KB to refer to a Knowledge Base in the logical sense, either stored in a system (e.g. an RDF triple store) or composed by the contents of several metadata files.

We shall use the term KB to refer to a Knowledge Base in the logical sense, either stored in a system or composed by the contents of several metadata files. Let T be the set of all possible triples that can be constructed from an infinite set of URIs (for resources, classes and properties) as well as literals (Gutierrez *et al.*, 2004). We shall use K to refer to the set of RDF/S triples of the form (subject, predicate, object) of a KB. Furthermore, hereafter we assume that K denotes the set of triples as produced by the adopted metadata schemas and ontologies and the *ingested* facts (yielded by manual or automated processes). Then, a KB can be seen as a finite subset K of T , i.e. $K \subset T$.

We shall use *C(K)* to refer to the *closure* of K. Note that the closure can be defined with respect to the standard inference rules for RDF/S and/or other custom rules (e.g., like those that we introduce in Section 3 of this paper) and is unique.

More formally:

Definition 5. *The closure of a K, is the set of all triples that either are explicitly asserted or can be inferred from K.*

In this regard, we can distinguish C_RDFS(K), C_customRules(K) , C_customRulesAndRDFS(K). We shall use *C_RDFS(K)* and *C_customRules(K)* to refer to the closure of K defined with respect to the standard inference rules for RDF/S and the custom ones respectively. The set of triples in *C_RDFS(K)* is a proper subset of *C_RDFSandCustomRules*:

- $C\_RDFS(K) \subset C\_RDFSandCustomRules(K)$

Lastly, *C_RDFSandCustomRules(K)* is the set of the explicit and inferred triples derived both from RDFS and the custom rules. In other words, this set is the union of the two previously defined closures:

- $C\_RDFSandCustomRules(K) = C\_RDFS(K) \cup C\_customRules(K)$

We can use *Red(K)* to denote the reduction of K. The reduction is the minimal set of facts that have the same closure as K, and for the case of RDFS rules it is unique if the relations are acyclic (Zeginis *et al.*, 2011). More formally:

Definition 6. *The reduction is the minimal set of facts that have the same closure as K, i.e., $C(R(K)) = C(K)$.*

Analogously we can define R_customRules(K) and R_customRulesAndRDFS(K). The following diagrams (Fig. 5.1 and Fig. 5.2) show the Venn diagrams of these sets.

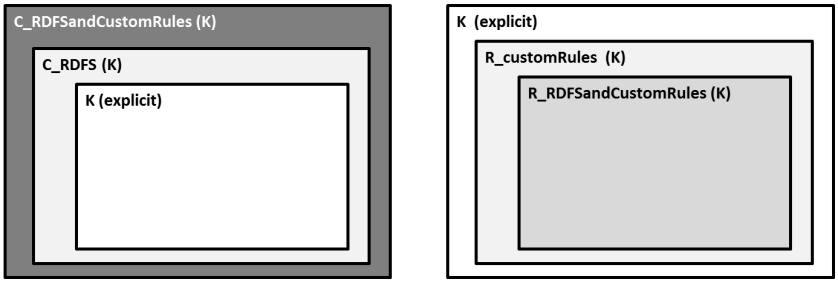


Figure 5.1: Closures and Reductions (part 1)

As a result, according to the definition of reduction and closure the following must hold:

- $C\_customRules(R\_customRules(K)) = C\_customRules(K)$
- $C\_RDFS(R\_RDFS(K)) = C\_RDFS(K)$
- $C\_RDFSandCustomRules(R\_RDFSandCustomRules(K) = C\_RDFSandCustomRules(K)$

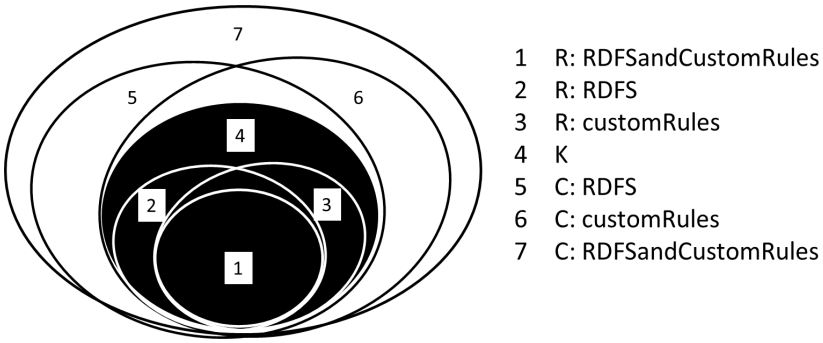


Figure 5.2: Closures and Reductions (part 2)

A repository policy could be to keep either K, or *Red(K)*, or *C(K)*stored. Storing *C(K)* could could offer better query execution time with regard to information retrieval. However, that would increase the storage space requirements. Thus, we exclude *C(K)* from our discussion due to the last fact. If, on the other hand, we decide to store *Red(K)*, that approach results to high maintenance cost, even though it can minimize further the storage space. This is because in order for the minimized storage space to be maintained, the reduction must be applied every time after there is an update request (i.e., addition or deletion of a triple). Therefore, deciding to keep stored K is a reasonable choice in terms of storage space and maintenance requirements.

One of the main characteristics of a repository is also to provide efficient mechanisms for information retrieval. One of the issues is whether the inferred triples should be materialized and stored or not. In the first case, the derived information can be found and retrieved efficiently, while this is not true for the second one. However, if the inferences are not stored, the storage space requirements are lower than the first case.

5.1.2 Requirements and Approaches

This section examines implementation requirements and approaches. Assuming that K is stored in a triple store/repository, the key aspect of a general implementation approach is that any inferable information (i.e., *C(K)*) according to our custom rules, should not be pre-computed and stored in the repository (static materialization) but be derived at query evaluation time instead (dynamic materialization). This requirement is crucial in order to satisfy our initial intention to reduce the space storage requirements. We shall also discuss other requirements and the case of Virtuoso, an RDF data store featuring a backward chaining reasoning.

Triple stores are frameworks providing mechanisms for storing and querying RDF graphs. Over the last few years there has been an explosion in the development of such frameworks leading to high competition. The technologies developed vary from commercial to open source technologies, however they all promise to handle the management of RDF data efficiently. Some of the most popular triple stores available are the following:

Table 5.1: Triple Stores and their capabilities

	OWLIM family	**Allegrograph**	**Sesame**	**Virtuoso**	**Jena**
SPARQL Support	√	√	√	√	√
SPARQL/ Update Support	√	√	√	√	√
Custom rule-based inference	√	√	√ with external reasoner	√	√
RDFS inference	√	rdfs:type rdfs:subClassOf rdfs:subPropertyOf rdfs:domain rdfs:range	√	rdfs:sub PropertyOf rdfs:sub ClassOf	Almost all RDFS axiomatic and entailment rules
Material-ization	static	dynamic	static	dynamic	static dynamic

- Allegrograph[1]
- OWLIM-Lite, OWLIM-SE[2]
- Sesame[3]
- Virtuoso[4]
- Jena[5]

Jena and Sesame are open source. Allegrograph and Virtuoso have both free and commercial versions. The free version of OWLIM family is the OWLIM-Lite, while the commercial is the OWLIM-SE. For a larger set of triple stores, their main features and references to their experimental evaluation one could consult the W3C's survey [6]. Thus, the issue here is choosing of a triple store that will meet the requirements of our approach's implementation.

One of such requirements is the ability of reasoning based on user defined custom rules. Moreover, since CIDOC CRM and CRMdig are based on a hierarchy of properties and classes, it would be desirable the support of inference associated to the RDFS entailment rules of subproperty and subclass. Another requirement, as previously discussed, is the computation of inferences (i.e., the transitive closure of K, $C(K)$) to be made at query time. Although, most of the triple stores meet the first requirement, they fail to meet the second one.

[1]http://www.franz.com/agraph/allegrograph/
[2]http://www.ontotext.com/owlim
[3]http://www.openrdf.org/
[4]http://virtuoso.openlinksw.com/
[5] http://incubator.apache.org/jena/
[6]http://www.w3.org/wiki/LargeTripleStores

More specifically, some of them such as the stores of OWLIM family materialize the full closure of K. The $C(K)$ is usually computed and stored in the repository along with K in a pre-processing phase before new queries can be evaluated. Examples that do not follow this approach are Allegrograph and Virtuoso. Their reasoning capabilities can be alternatively enabled for a specific query allowing the derivation of new implicit facts.

In addition to the above requirements, the satisfaction of update requests of the stored KB is not trivial. The evolution of the KB relies on the insertion of new triples and the deletion of existing ones. Thus, the query language implemented should support update operations on K. The new version of SPARQL query language, 1.1 (Harris & Seaborne, 2010) which is adopted by the majority of the different triple stores, provide semantics for such operations.

Comparing Virtuoso and Jena, from a performance point of view, it has been showed that generally the former outperforms the latter (Bizer & Schultz, 2008), (Thakker *et al.*, 2010). From a practical point of view, Virtuoso has more benefits that Jena. For instance, RDF views are supported by Virtuoso in contrary to Jena and that might be useful in the case of conjunctive inference rules. Other benefits of Virtuoso might be the implementation of transitivity closure's computation. That implementation offers additional information to query results, such as the depth of the inferred triple patterns in the transitive closure's graph.

According to our knowledge, Allegrograph has not been extensively benchmarked and compared to other triple stores taking also into account user defined custom rules. The free version of Allegrograph is limited by a 5 million triples maximum. Moreover, it does not support SQL in contrast to Virtuoso that can be used as a general RDBMS and not limited to an RDF triple store. One other advantage of Virtuoso is the offered flexibility with respect to RDFS reasoning. Even though, not all RDF/S inference rules are supported, the user might define different RDF/S schemas for a specific repository and for the same queries retrieve different information using backward reasoning. However, in Allegrograph, according to the provided documentation[7], a new repository has to be created. Lastly, the advantages of the implementation of the transitive closure's computation and the definition of RDF views, as discussed previously, also apply in this case.

The case of Virtuoso

As an implementation approach example that meets all the aforementioned requirements we consider the case of Virtuoso. Virtuoso is an RDF data triple store including the feature of backward chaining reasoning; it does not materialize all inferred facts but rather looks for the explicit facts as a foundation for the former ones (Erling & Mikhailov, 2009). Its internal storage method is relational and so the triples are stored in tables in the form of quads (g, s, p, o). Each column of a quad represents the graph, subject, predicate and object.

Virtuoso support the SPARQL query language and some extensions such as SPARUL statements. The latter is used for the update operations and includes statements such as insert, modify, delete, load etc. Regarding Virtuoso's reasoning

[7] http://www.franz.com/agraph/support/learning/SPARQL-with-Reasoning.lhtml

capabilities, its reasoner covers the related entailment rules of rdfs:subClassOf and rdfs:subPropertyOf and user defined custom rules can be expressed in construct queries.

Transitivity is also supported by two different ways. Given a RDF schema and a rule defined and associated with that, the predicates rdfs:subClassOf and rdfs:subPropertyOf are recognized and the inferred triples are derived when needed. In the case of another predicate the option for transitivity has to be declared in the query (Erling & Mikhailov, 2009). This way has the advantage of retrieving information regarding the evaluated transitivity steps. For example, it could return the length of time associated with each transitive step or the total number of steps for a particular triple. This might be useful in order for example to find the root activity in the hierarchy of the activities.

5.2 Experimental Evaluation

5.2.1 An Analytical Cost Model

In this section, we present an analytical cost model with respect to the number of inferences derived by the application of our custom reasoning rules. We focus on the hierarchies of activities and the actors who are responsible for the former ones.

We assume that our KB is represented by instances in the form of RDF triples (i.e., *s p o*) based on the CIDOC CRM's schema. We will denote the number of RDF data triples (i.e., instances) in our initial KB by T. We also denote the number of redundant instances by R.

Definition 7. *Let act be an activity. The set subAct(act) includes all the (direct and indirect) subactivities of act. More formally:*
$subAct(act) = \{subactivity | \exists t : (subactivity formsPartOf act) \in K \cup C(K)\}$.
We will denote the cardinality of subAct(act) as N i.e., $N = |subAct(act)|$.

Definition 8. *Let ac be an actor. The set Nac(ac) includes all the activities associated with ac. More formally:*
$Nac(ac) = \{activity | \exists (activity carriedOutBy ac) \in K \cup C(K)\}$.
We will denote the cardinality of Nac(ac) as N i.e., $Car = |Nac(ac)|$.

Definition 9. *Let act be a activity associated with an actor ac. The set subAct(act, ac) includes all the (direct and indirect) subactivities of act also associated to ac. More formally:*
$Nactacr(act, ac) = \{subactivity | subactivity \in subAct(act)$
$and subactivity \in Nac(ac)\}$.
We will denote the cardinality of Nactacr(act, ac) as M i.e., $M = |Nactacr(act, ac)|$.

The inference rule R1 propagates an actor ac of an activity act to all the (direct and indirect) subactivities of act. As a result,we have to estimate M for various morphologies of the hierarchy of activities. Below we start the estimation of N (i.e., the number of inferred transitive *partOf* associations) starting from the simple morphologies that can occur (i.e., tree-shaped hierarchies) and conclude to the most complex ones (i.e., DAG-shaped). Therefore the total number of triples is $N+M$.

Lemma 1. *Assuming a complete tree-shaped activity hierarchy of depth d and degree b, then*

$$N = T + \sum_{i=2}^{d} b^i(i-1) - R.$$

Proof. Inferred transitive activities can be derived after the level 2 of the tree. For each level i of the tree, we have b^i activities and for each activity we infer *i-1* triples of the form *(act formsPartOf superact)* for the upper levels of the tree. □

Lemma 2. *Assuming a tree-shaped activity hierarchy of depth d and a degree b_i for each level i of the trees, then*

$$N = T + \sum_{i=2}^{d} b_i(i-1) - R.$$

Proof. Inferred transitive activities can be derived after the level 2 of the tree. For each level i of the tree, we have b activities and for each activity we infer *i-1* triples of the form *(act formsPartOf superact)* for the upper levels of the tree. □

Lemma 3. *Assuming a DAG-shaped activity hierarchy of maximum depth d, and each partOf edge has a depth value following the path from the roots of the DAG, then*

$N = T + \sum_{i=1}^{n}(depth_{edge_i}(depth_{edge_i} - 1)) - R$, *where n is the total number of such edges.*

Proof. The depth of each *partOf* edge is denoted by the variable $depth_{edge}$. For each edge's depth (i.e., $depth_{edge_i}$) there can be inferred $depth_{edge_i}$ *partOf* edges towards the roots of the DAG. □

5.2.2 Experimental Design

The objective of this evaluations is to understand the trade off between storage space and the performance of queries and updates. To do so, we evaluate our algorithms experimentally and compare our strategy of storing K to the one of storing *C(K)*. We do not take into consideration the third strategy of storing the redundant-free KB, because of the additional costly reduction techniques that have to be applied after every update operation, either for addition or deletion of information.

In the first part we examine the results on real data and in the second part on synthetic data. In the case of real data, the experimental process involves only measurements on storage space taking into consideration the RDF/S rules and our custom rule R1. We did not examine the rules R2 and R3 due to the absence of data regarding those rules.

In the second case, the synthetic data was used mainly to test the scalability of our algorithms in large datasets; our synthetic data involved only rule R1 (rule R2 can be applied on similar graph morphologies and rule R3 is much more simpler as it is not not relying on hierarchies).

The evaluation consists of measurements on storage space and query execution time. Secondly, the time of maintenance and update operations is measured including the time for computing and storing transitive inferences and the time for

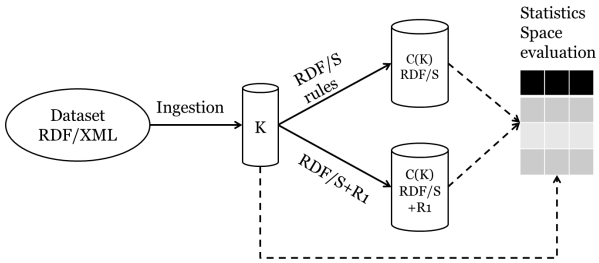


Figure 5.3: Experimental design on real data

disassociating/contracting a fact from the KB. The three experiments in brief are the following:

1. storage for real data
2. storage vs query for synthetic data
3. maintenance/update operations for synthetic data

The statistics of the data are shown in the Appendix of this paper.

Experimental Settings

We performed our experiments on a Pentium 4 Cpu at 2.55GHz and 2GB RAM, running Linux Ubuntu 11.10. The chosen triple store was the Virtuoso's open source version 6.1.5 taking into account its advantages discussed in the previous section. Based on the available memory we chose to change some parameters of Virtuoso in order to minimize swapping as much as possible. Refer to the Appendix for the configuration file of Virtuoso.

5.2.3 Results and Discussion

Real Data

To test our approach on real data, we extracted data of the most completed metadata repository available as part of the 3D COFORM project[8] in FORTH-ICS. We inserted the metadata in the virtuoso server; we compared the repository of the initial data ingestion and the one containing all the materialized inferences with respect to the RDF/S rules (as supported by Virtuoso) and rule R1 including the transitive relation *P9 forms part of* (see Fig. 5.3).

For this experiment, we ingested a total dataset of 150,434 triples with respect to the CIDOC CRM model in Virtuoso. The relations of *P9 forms part of* and *P14 carried out by* were 8,937 and 0 respectively. The reason for the zero result is that although the initial data did not consist of triples having as predicate the *P14 carried out by* property, there were triples having as predicates subproperties of *P14 carried out by* . As a result, after the RDF/S inference there were 129 triples with

[8] www.3d-coform.eu/

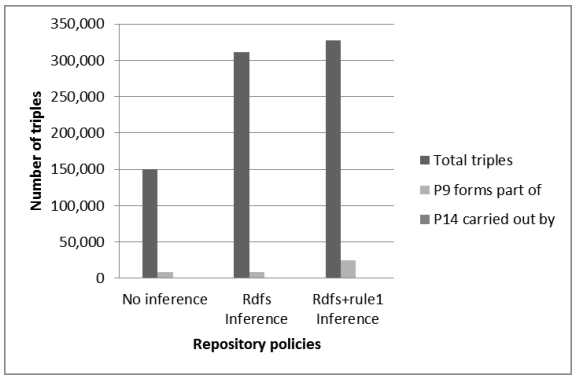


Figure 5.4: Space evaluation on real data

the *P14 carried out by* predicate. The total triples in the repository in this case were 310,919.

The same behaviour is observed with the triples regarding *P9 forms part of* predicate; after the RDF/S inference there were also 8,937 since *P9 forms part of* is a super property defined in CIDOC CRM. Subsequently, the total triples after applying rule R1 to the dataset were 327,452 and the triples regarding *P9 forms part of* and *P14 carried out by* predicates were 24,444 and 717 respectively (see also Fig. 5.4). The number of *P9 forms part of* triples in this case has been increased since we take into account the transitivity of this relation. As a result, we observe that there was an increase of 106% and 118% in the number of total triples from the repository containing K to the ones of the RDF/S inference and both RDF/S and rule R1 respectively. The increase percentages in the strategies of storing the inferences with respect to the rule R1 and/or the RDF/S rules indicate that our approach is more beneficial considering the space storage requirements.

Synthetic Data

In addition, to test the scalability of our algorithms in large datasets, we synthesized data involving only the *P9 forms part of* and *P14 carried out by* predicates. The goal was the evaluation of space and query performance with respect to our rule R1. Since it is based on the depth of the activity hierarchies, we had to realize how critical was this parameter to our goal. We expect running time to be depending on the depth of the activity hierarchies, so our experiments evaluated the effect of this parameter on the running time. Another parameter that we evaluated for its effect on performance was the number of the ingested triples.

Thus, having in mind these two parameters we synthesized hierarchies of the most complex form i.e., DAGs. For this purpose we used *PowerGen* (Theoharis *et al.*, 2012) that generates random DAG-RDF/S schemata with subsumption hierarchies. These are real-world-like schemata whose schema exhibits distribution commonly found in reality and can be parametrized by several parameters such as maximum depth and the number of classes. Thus, we generated multiple schemata of 1000 classes and depths of 3,4 and 6. Powergen produces schemata having generic prop-

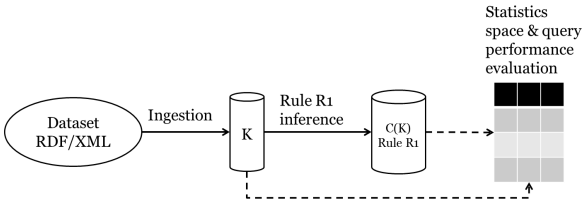


Figure 5.5: Experimental design on synthetic data

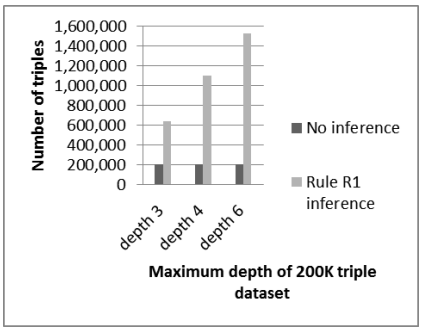
erty names. To test our approach, we require hierarchies of *P9 forms part of* property, so we chose to replace the *RDFS: SubClassOf* property with *P9 forms part of* . This way, the parameters of Powergen can also be used to tune the morphology of *P9 forms part of* hierarchies.

The resulted repositories contained datasets consisting of multiple disjoint unions of graphs of 200K, 400K, 600K, 800K and 1M *P9 forms part of* triples (K denotes thousand and M million). For each dataset we distributed a number of actors randomly to the activities equal to 1% of the *P9 forms part of* triples. Lastly, it should be noted that the number of activities and actors has been kept the same for each level of the graphs in each dataset. In the sequel we present the results of experiments on two policies that can be followed, regarding the choice of storing or not the inferred triples of the transitive *P9 forms part of* relation and the rule R1. Therefore, in the following figures "No inference" denotes the repository containing only the initial triples and the "Rule R1 inference" denotes the repository containing all inferences derived from rule R1.

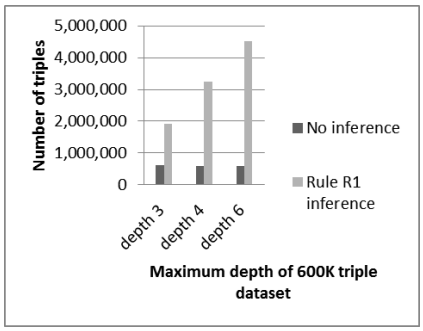
Storing inferred triples. In the previous experiment we showed the space requirements for different repository policies on real data. However, in order to further investigate how the space requirements are affected by the parameter of maximum depth of DAGs and the number of triples, we conducted similar experiments for synthetic data up to 1 million triples and depth 6 (as discussed previously). For this experiment, we ingested dataset of 200K, 400K, 600K, 800K and 1M triples with *P9 forms part of* and *P14 carried out by* predicates. After the ingestion, we materialized the transitive and the inferred triples derived by applying rule R1.

In the figures below we show the results of the 200K (Fig. 5.6a), 600K (Fig. 5.6b) and 1M (Fig. 5.6c) triples in the respective maximum depths 3,4 and 6. The total triples after the inference of rule R1 and for the 1M datasets were 3,229,902, 5,478,634 and 7,560,778 for 3, 4 and 6 depths respectively. Thus, we observe that for the 1M triples we have an increase of 218%, 435% and 648% for depths 3,4 and 6 respectively. The critical parameters in this case are the number of triples, and the maximum depth of the DAGs. As a consequence, both the results of the experiments on real data and synthetic data suggest that policies following the concept of storing the inferred triples contribute to a significant amount of overhead with respect to the storage space needed.

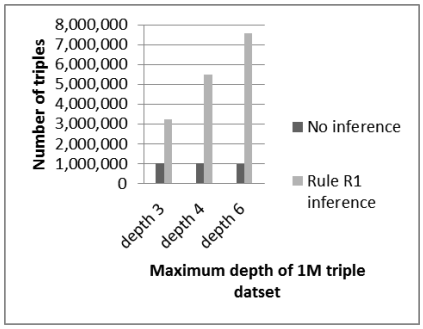
Querying RDFS activity hierarchies. We also experimented with the performance evaluation of a query that is expected to be one of the most common that would involve the computation of transitive *P9 forms part of* relations at query



(a) 200K triple dataset



(b) 600K triple dataset



(c) 1M triple dataset

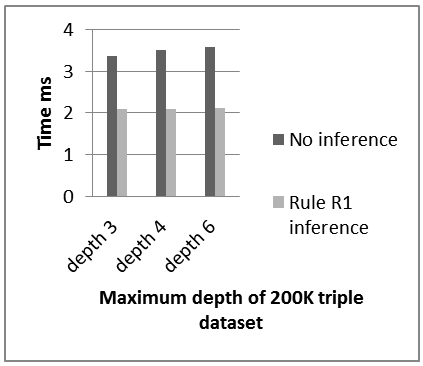
Figure 5.6: Storage space evaluation

time. That query was to "retrieve all associated actors given a particular activity" including those of its superactivities (i.e., application of rule R1). We evaluated this query for all the activities in the hierarchies and their respective depth, taking the averaged measurements. Every time we changed the dataset we cleared the cached buffers of the operating system in order to have more reliable results.

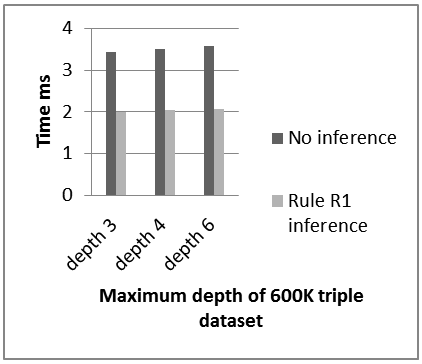
Figures 5.7a, 5.7b and 5.7c show the time of the query in milliseconds (ms) for the datasets of 200K, 600K and 1M triples and the depths 3, 4 and 6 respectively. We observe that in the case that our repository contained all the inferred triples of rule R1 the time is almost the same. Since all the triples are stored there is not any additional computation such as the transitive closure to be evaluated at query time.

In comparison, following our strategy the query time performance has an increase reaching almost at 3.5ms. This is acceptable since the transitive closure of the *P9 forms part of* relation has to be computed at query time. Moreover, Fig. 5.8 indicates a slight increase in ms as a result of the increasing depth for almost all datasets. Therefore, as the hierarchy becomes deeper, the number of the inferred *P9 forms part of* and *P14 carried out by* triples increases as well. Another observation that can be made is that, in average, the time needed to evaluate this query for all the activities is independent from the number of triples stored into the repository. This can be explained by Virtuoso's scalability feature (Erling & Mikhailov, 2009).

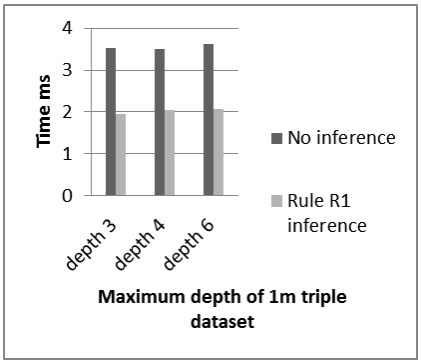
Computing and storing the transitive closure. One issue that concerns the materialization policy is the time needed for the computation and storage of



(a) 200K triple dataset



(b) 600K triple dataset



(c) 1M triple dataset

Figure 5.7: Query performance

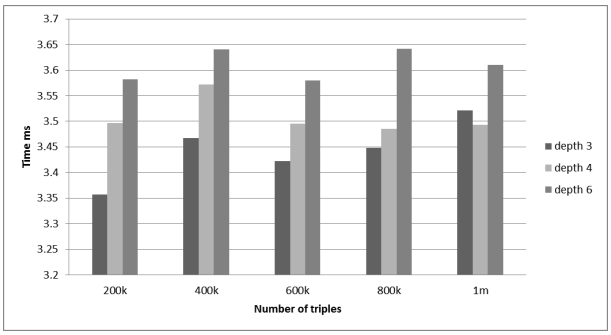


Figure 5.8: Query performance for all datasets

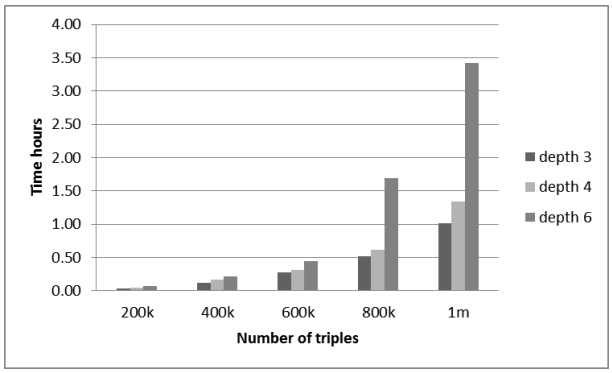


Figure 5.9: Time for computation and storing the transitive closure of all datasets

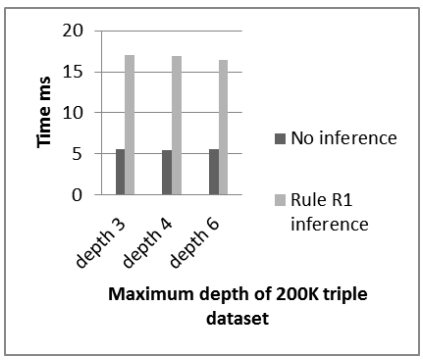
the transitive closure of the *P9 forms part of* relations. That has a general impact on the maintenance operations of a repository. In a database of initially 59,903,298 triples we performed the discussed task for each dataset. However, since the memory usage of this task was increased and there was some usage of the swap file that was cleared, before starting the task for a new dataset. In this Section we present the results of measurements with regard to this issue.

Figure 5.9 shows the results in hours for all the triple datasets and for the various depths. We observe that the both parameters of the depth and the number of triples affect the time needed for the transitive closure to be computed and stored in the repository. There seems to be an exponential correlation between the increase of time and the one of maximum depth for all datasets. Moreover, even though the inferences are stored in a different repository than the initial one, the numbers are still high, taking into account that these regard only one relation. As a result, our approach has one more advantage compared to the policy of storing all inferences.

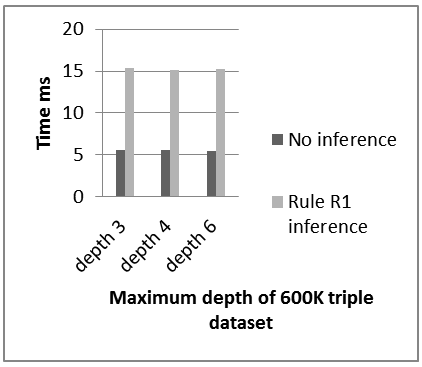
Time evaluation of disassociation. In this experiment we measure the time needed in order for the actor disassociation update operator to be completed. We run this operation for all the actors in our datasets and depths and took the average measurements.

Figures 5.10a, 5.10b and 5.10c show the results of the respective datasets and depths. We observe that the operation in the repository following our strategy of not storing the inferences requires less time compared to the one of storing all the inferences. In the first case the average required time is approximately 5ms, while in the second is 15ms. This amounts to an increase of 200% in the second case. The main tasks of disassociation is apart from the computation of the transitive closure, the deletion of several triples. In the first case, even though the transitivity is evaluated at query time there are no inferred triples to be deleted. However, in the second one, even though the transitivity is not evaluated at query time, the number of triples that are deleted by the operation exceeds the respective one in the first case.

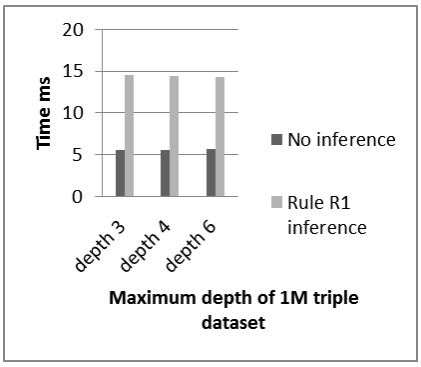
Time evaluation of contraction. Lastly, in this experiment we measure the time needed in order for the actor contraction update operator to be completed. Similar to the previous experiments, we run this operation for all the actors in our



(a) 200K triple dataset



(b) 600K triple dataset

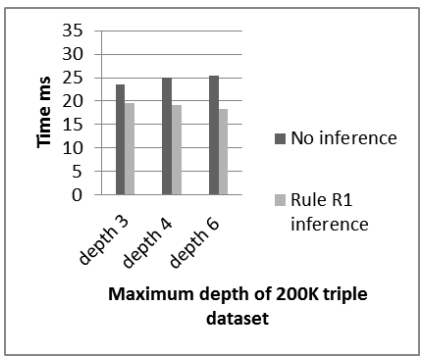


(c) 1M triple dataset

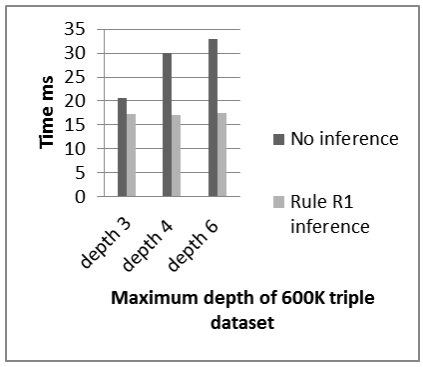
Figure 5.10: Time evaluation of disassociation

datasets and depths and took the average measurements.

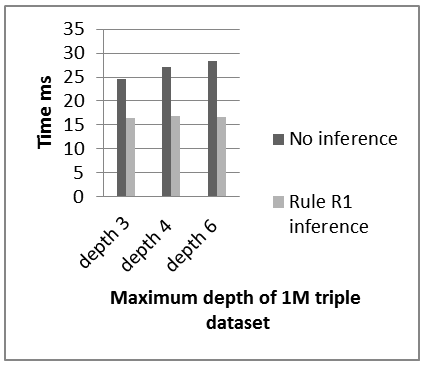
The operation of contraction is more complicated than disassociation. Except from disassociation which is performed in the end, the operation requires the computation of maximal activities and the addition of triples in order to preserve implicit knowledge. Figures 5.10a, 5.10b and 5.10c show the results of the respective datasets and depths. The time required in the repository in which all inferences are stored is considerably less than the one following our approach. More specifically, the numbers in average are close to the case of the disassociation operation, i.e., approximately 16-17ms. This can be explained by the fact that since all inferences are stored, Virtuoso only checks for their existence without adding new triples into the repository. Thus, the time required is mostly for the operation of disassociation in order for contraction to be completed. Regarding the repository following our policy, the time is increasing in relation to maximum depth suggesting that the tasks of the transitivity computation and the search for the maximal activities are more time consuming in this case.



(a) 200k triple dataset



(b) 600k triple dataset



(c) 1m triple dataset

Figure 5.11: Time evaluation of contraction

5.3 Summary

This section detailed on the implementation issues of our approach. We discussed how this could be implemented on several triple stores proposing some of which are the most adequate for this purpose. Choosing the most scalable one, we conducted experiments on real and synthetic data comparing two policies that could be adopted by a metadata environment.

Our findings showed that the policy of not storing the implicit information is beneficial with regards to the reduction of storage space requirements of provenance information, with no significant overhead in the query time performance. Both for the real and synthetic data the reduction percentages were over 100% and in the second case over 200%, and compared to the second policy of storing all the inferences, the overhead in query execution was approximately 1.5ms.

The disadvantage of the latter strategy is also realized by the fact that computing and storing the transitive inferences, which is a KB maintenance operation, seemed to have an exponential behaviour in relation to execution time. Lastly, regarding the update operations disassociation and contraction, the former was executed faster under the second strategy, while the latter one achieved better execution times following the first strategy. It should be noted that for all experimental measurements

involving the synthetic data the maximum depth of graph hierarchies had a quite influential role.

Chapter 6

Reasoning Extensions and Temporal Aspects

In the previous Chapters we introduced a set of custom inference rules in order to minimize the storage space requirements of provenance information. However, that set of rules might be extended. In this chapter, firstly we elaborate on several extensions to these reasoning forms (Section 6.1) describing a metamodel (Section 6.1.3) towards a more general approach for the specific patterns that could be exploited by the inference rules. Secondly, we explain the temporal reasoning related to CIDOC CRM (Section 6.2), and lastly we define an essential and optional parthood relation taking into account some of its temporal aspects as well (Section 6.3).

6.1 Extending our Reasoning Rules

In Chapter 3 we introduced a specific set of inference rules involving the propagation of features from wholes to parts. From a semantic point of view, the rules are reasonable with respect to the semantics of the classes and properties defined in the ontology. For example, we assume, by convention, that a person carrying out a process carries out all of its sub-processes; this is reflected by the formal phrasing of the corresponding rule.

However, alternative rules could also be reasonable and applicable. For example, an alternative rule could be defined by stating that a person carrying out a process carries out at least one of its sub-processes or that a person carrying out a process, also carried out all of its super-processes. In this Section we present such alternative rules respecting the semantics of CIDOC CRM's ontology. Following the format of Chapter 3, the two subsections introduce two more rules accompanied by respective examples. The last Section generalizes our approach by examining a metamodel which can be used for the specification of similar reasoning rules.

6.1.1 Presence of actors to super activities

- **Rule 4. Presence of actors to activities**
  If an actor carried out a subactivity, then he was present at its superactivity.

$$\forall x, y, z(carriedOutBy(x, y) \wedge formsPartOf(y, z) \rightarrow wasUsedFor(x, z)) \tag{6.1}$$

Example

Consider again the example of Section 3.1 taken from the cultural heritage domain consisting the activities of a *Laser Scanning acquisition* activity and a responsible actor the *Starc-Institure*. Now, we could have a state of our KB in which the *Starc-Institute* is only associated via the *P14 carried out by* relation only to some of the subactivities of *Laser scanning acquisition* e.g., to the *Capture 1_7*. In this case, by the application of rule 4 the presence of the *Starc Institute* will be inferred to all the superactivities of *Capture 1_7*. Thus, *P14 was present at* links will be added associating the *Starc Institute* with the activities *Laser scanning acquisition* and *Detailed sequence of shots* ,see Fig. 6.1.

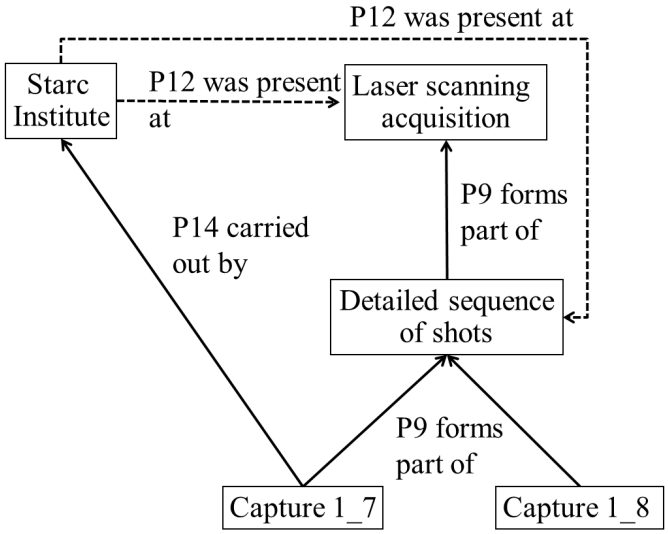


Figure 6.1: Example of rule R4

6.1.2 Use of objects to subactivities

- **Rule 5. Use of objects to subactivities**
  If an object (device) was used for an activity, then it was also used to all of its subactivities.

$$\forall x, y, z(isComposedBy(x, y) \wedge wasUsedFor(x, z) \rightarrow wasUsedFor(y, z)) \tag{6.2}$$

In the example of Section 3.2 we introduced the rule for propagating the use of devices to their parts. However, we did not consider the fact that activities can be composed into further subactivities. In this case, the use of a specific object can be propagated to the corresponding subactivities. In our example, the *multiviewdome* device will be associated *implicitly* via the *P16 was used for* to the subactivities of the particular activity. This is accomplished by the application of rule 5. As a result, the *multiviewdome* device will be associated to *Capture 1_7* and *Capture 1_8* which are subactivities of the *Detailed sequence of shots*, see Fig. 6.2.

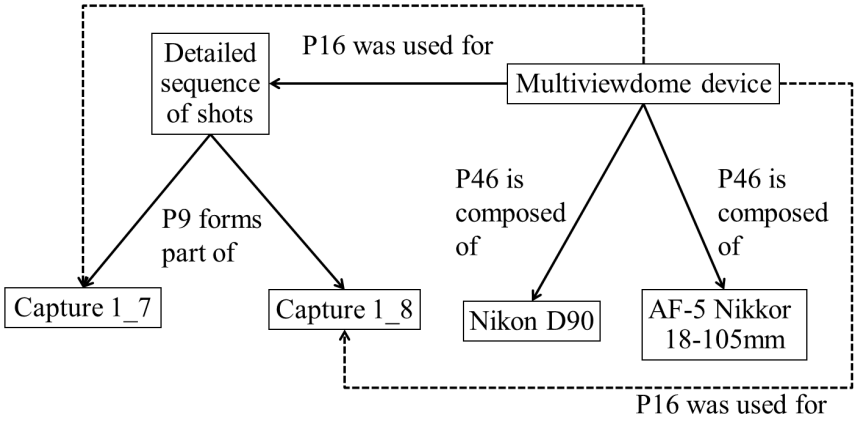


Figure 6.2: Example of rule R5

6.1.3 Towards a Metamodel for Reasoning Rules

In this Section, we propose a metamodel in order for a larger set of rules to be defined and a better understanding for the pattern be exploited as a basis for this process. This pattern is considered to be independent from our application context ontology CIDOC CRM. However, one should consider the semantics of the entities and the relations among them in order for the rules to be logically valid.

The basic characteristic of this model is the *part of* relation by which entities can be decomposed into subentities. Two entities are often related by a property representing some attribute, feature or role which an entity had regarding the other one. Fig. 6.3 depicts such a model consisting of two entities having been decomposed into further subentities and related with a property instance.

The set of reasoning rules which might be defined taking into consideration this pattern are to propagate the features from the entities to the respective sub entities, i.e. from wholes to parts (like the rules R1 and R2). In the same spirit one could define rules for propagating features from the sub entities to the entities i.e., from parts to wholes (like the rules R4 and R5).

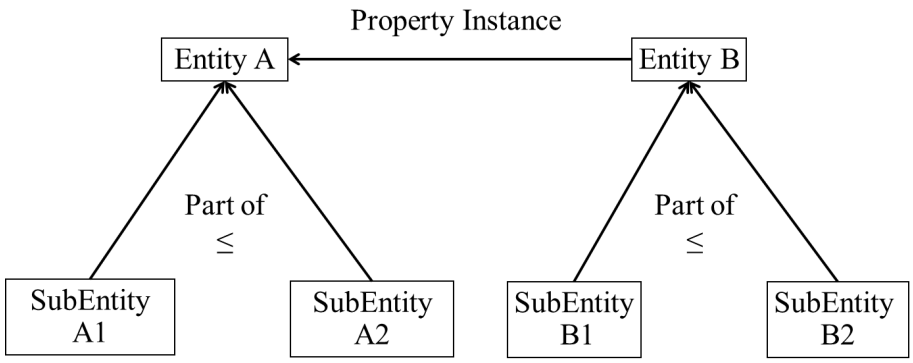


Figure 6.3: A metamodel for reasoning rules

6.2 Temporal Aspects

In this section, we present a temporal reasoning as an introduction to the subsequent section regarding temporal essential and optional parts. We explain how time is associated to ontological entities taking into account cases in which the precise time-spans of certain events are uncertain. We then integrate this kind of reasoning modelling the existence of material objects the so-called persistent items.

6.2.1 Time modelling in CIDOC CRM

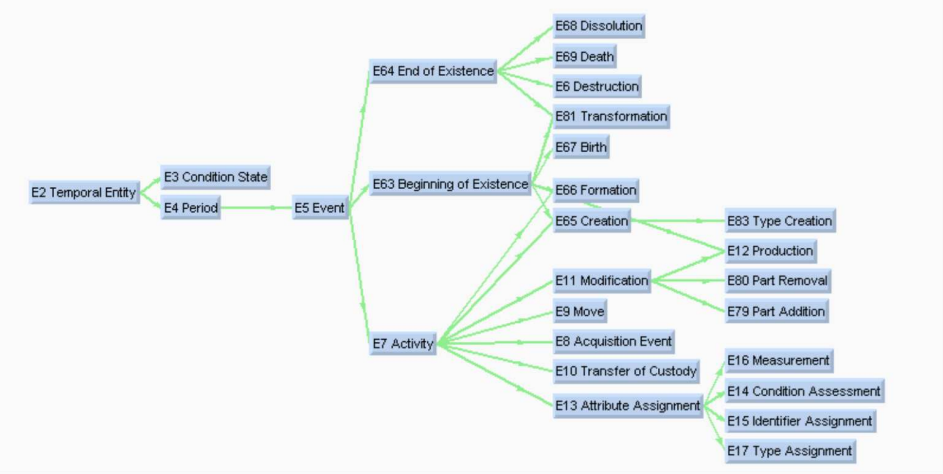


Figure 6.4: Temporal Entity Hierarchy of CIDOC CRM

The two disjoint main classes of the model is the E2 Temporal Entity and the E77 Persistent Item. All other classes are in most cases direct or indirect specializations of the aforementioned classes. Temporal entities comprise instances of the classes E4 Periods and E5 Events modelling phenomena that have a specific span in time (see

Fig. 6.4). Persistent items can either be physical entities, such as people, animals or things, or conceptual entities such as ideas, concepts, products of the imagination or common names.

6.2.2 Temporal Reasoning on Events

CIDOC CRM is a model of possible states of affairs in the real world in which historical phenomena are identified by periods. Periods are sets of coherent phenomena or cultural manifestations bounded in time and space. Examples of periods are the Neolithic Period, the Ming Dynasty or the Jurassic Period. Events are specializations of periods. Changes of states in cultural, social or physical systems can be regarded as events and be represented by the *E5 Event* class in CIDOC CRM. Moreover, events can be decomposed into subevents or composed by larger events which are compatible with the definition of periods. Examples of events are the birth of Cleopatra or the World War II (Crofts *et al.*, 2011).

Temporal reasoning on events is associated with time spans defining their extent in time. Events cannot have multiple time spans because that would express different evidence about their true temporal boundaries.

The class defined in CIDOC CRM for modelling time spans of events is the *E52 Time-Span*. Time spans are based on the principle that in most cases the events' temporal extents (i.e., their exact beginning and ending) are not precisely known (Doerr *et al.*, 2004). Most often there is only information about an outer interval, that is, by some terminus post quem (TPQ) and terminus ante quem (TAQ) for the event, defining the fuzzy boundaries of the event's time span (Holmen & Ore, 2009).

On the other hand, if there is enough evidence, one could define the respective inner interval of the event representing the certain span of its occurrence. In other words, the inner interval corresponds to the minimum period of time covered by an event and the outer interval corresponds to the maximum period of time within which an event occurred. The time which exists between the two intervals symbolizes the uncertainty of the exact timing of an event. In the case of absolute accuracy these two intervals are identical.

The class which is defined by the ontology of CIDOC CRM for modelling time spans of events is the E52 Time-Span. Since Time-Spans may not have precisely known temporal extents, there are two properties in order for these temporal boundaries to be defined. The two properties are the P81 and P82. P81 property is dedicated for modelling the Time-Span's maximum known temporal extent i.e. ongoing_throughout. P82 property is for modelling the minimum outer bounds of events i.e. at_some_time_within. The time which exists between P81 and P82 represents the uncertainty of the exact timing of an event. We can conclude that the level of uncertainty can be reduced according to how large is the P81 time interval in relation to P82. The above temporal aspects are illustrated in Fig. 6.5.

Alternatively, we can define the above boundaries of P81 and P82 properties and its temporal constraints. For the P82 the 2 boundaries can be defined as "begin of the begin" and "end of the end". For the P81 accordingly can be defined as "end of the begin" and "begin of the end".

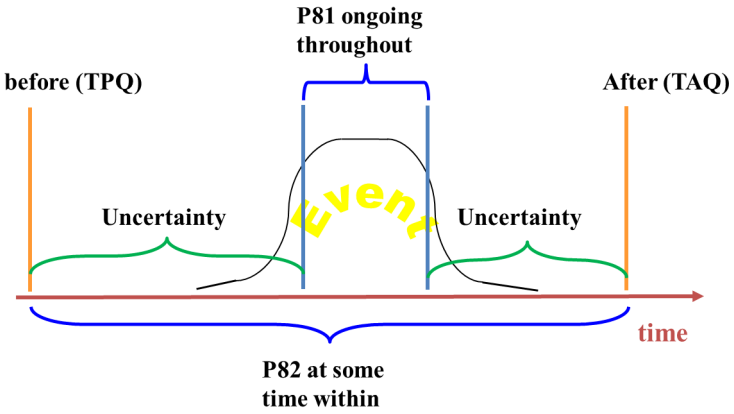


Figure 6.5: Timespans in CIDOC CRM

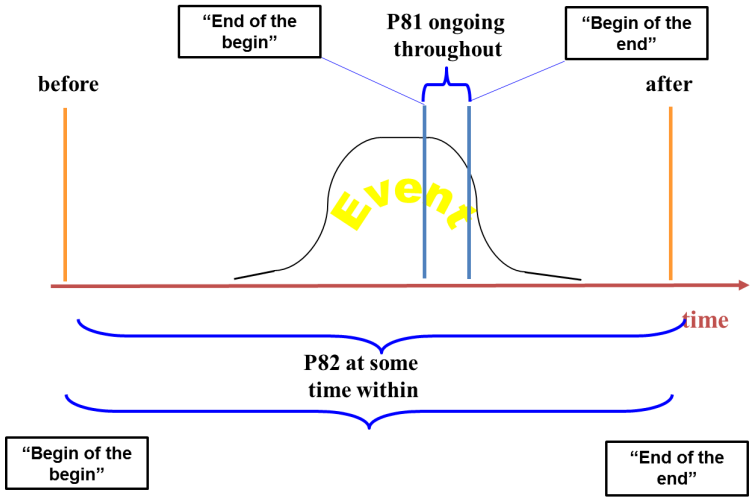


Figure 6.6: Alternative Representation of Time

Constraints

Following the above definitions for the boundaries of the 2 properties, we can constraint these intervals in order to allow some relative chronological reasoning. The "begin of the begin" is the start of a larger period so it has to be the precedent of the other boundaries. After that we can define the "end of the begin". Following the same logic, in the timeline the constraints can be the following:

- "begin of the begin" $\leq$ "end of the end"
- "begin of the begin" $\leq$ "end of the begin"
- "begin of the begin" $\leq$ "begin of the end"
- "end of the begin" $\leq$ "end of the end"

- "begin of the end" $\leq$ "end of the end"

It can be observed there is not a defined constraint between the boundaries of P81. As a consequence the time interval of P81 can be negative. So that the "end of the begin" can be precedent of the "begin of the end". This is allowed conventionally in cases in which the interval is unknown or there is not any time interval.

For simplicity purposes we can replace the above definitions:

- For the P82:
  - "begin of the begin" : "P82a"
  - "end of the end" : "P82b"
- For the P81:
  - "end of the begin": "P81a"
  - "begin of the end" : "P81b"

So, finally the constraints with the new naming of boundaries can be:

- "P82a" $\leq$ "P82b"
- "P82a" $\leq$ "P81a"
- "P82a" $\leq$ "P81b"
- "P81a" $\leq$ "P82b"
- "P81b" $\leq$ "P82b"

6.2.3 Temporal Reasoning on Persistent Items

Persistent Items can either be physical entities, such as people, animals or things, or conceptual entities such as ideas, concepts, or products of the imagination or common names. Persistent items are sometimes known in philosophy as "endurands". They can be repeatedly recognized within the duration of their existence by identity criteria rather than by continuity or observation. The class defined in CIDOC CRM which comprises these items is the *E77 Persistent Item*.

Persistent items' existence is bounded by events that cause its beginning and ending and thus by the classes (a) *E63 Beginning of Existence* and (b) *E64 End of Existence*. Their related properties also defined in CIDOC CRM are (a) *P92 brought into existence (was brought into existence by)* and (b) *P93 took out of existence (was taken out of existence by)*. The classes E63 and E64 are subclasses of *E5 Event* class. As a result, both of them are associated to E52 Time Spans (see section 6.2.2 via the *P4 has time-span (is time-span of)* property. The following figure illustrates the above relationships:

Pushing the above reasoning further, we assume that a persistent item's existence has a time-span (as it is described in Section 6.2.2) which is related to the time-spans of the aforementioned two events of existence. Its possible existence starts with the

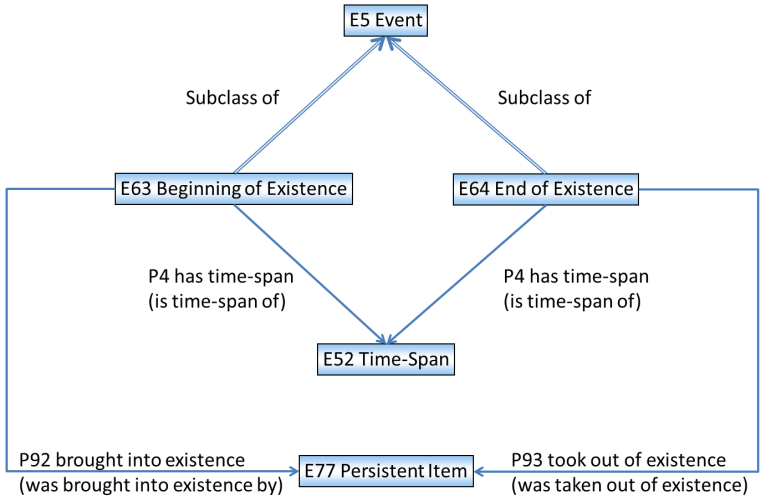


Figure 6.7: Temporal reasoning on persistent items

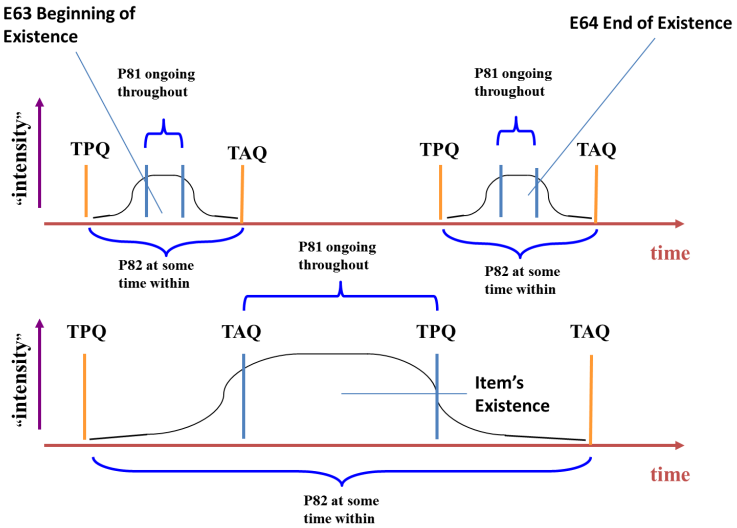


Figure 6.8: Existence of material objects

TPQ of beginning of existence and ends with the TAQ of end of existence. Its inner interval is defined in a similar way and starts with the TAQ of beginning of existence and ends with the TPQ of end of existence as shown in Fig. 6.8. Thus, the start of an E63 event denotes for the creation of a persistent item and the end of an E64 one for the destruction of the latter. However, it is certain that a persistent item exists after the end of an E63 event and before the start of an E64 one.

6.3 Modelling Essential and Optional Parts

In Section 2.3.2 we reviewed some theories on the parthood relation. We presented different mereonymic according to the relation of the parts and the complex objects and the adopted assumptions used in our reasoning. In this section, we refine some of the definitions regarding parts and wholes, introducing the notions of essential and optional parts.

As in previous chapters we will again be limited to heterogeneous physical objects. So wholes conceptualized as masses or collections are out of context. Thus, hereafter, we will consider that parts of a whole have solely a functional role type and as a result the whole is assumed to be a functional complex. Additionally, we will consider parts based on the compositional structure of the whole, the so-called structure-dependent parts following the theory of Gerstl & Pribbenow (1996). Structure-dependent parts are the parts that an entity can be naturally decomposed based on the a priori knowledge of its internal constructive structure and the resulting part-whole relations belong to the knowledge of the decomposed entity, i.e., the whole (Gerstl & Pribbenow, 1996).

6.3.1 Functional Essential and Optional parts

Parts can be further distinguished according to the functional role with respect to the wholes. In this section we will discuss distinction defining a functional essential and optional parthood relationship.

Ontological dependencies between parts and wholes is another aspect of parthood relation. Two of such dependencies are the essential and optional one. In general, we could say that a whole must have essential parts if it exists, but it could lack optional parts during its existence. The distinction described can be traced from the beginnings of philosophy (i.e., by Aristotle) and it remains a subject under discussion until now. In philosophy the term optional is referred as accidental. Philosophers often present arguments regarding the essential features and how the distinction between essential and not essential was derived (Gorman, 2005; Fine, 1994). However, there is some agreement that in modal terms, the word *must* reflects the notion of necessity, whereas *could* reflects possibility (Teresa, 2008). Thus, replacing the two words, a whole necessary has essential, whereas it is possible to have optional parts.

We shall specify the above notion of essential and optional dependency adding the concept of the functionality. Thus, we are interested in the behaviour of a device and subsequently its parts. Humans use different functional interpretations in order to explain what a device *does*. Especially in engineering different meanings for the term *behaviour* have been stated. Chandrasekaran & Josephson (2000) have confined them into five different interpretations. For example, an audio amplifier's behaviour might be to amplify the low power audio signals or behaves well as an effect to the output power. We are not concerned about these alternatives rather than the general notion of *function*.

In order to formalize the notion we introduce the predicate *isOperational(x,functionOf(x))* where the functionOf returns some value correspond-

ing to the behaviour of x and means that the component x has a function which is given by the previously referred function. Of course, x cannot have any function if it does not exist. So we have the following axiom:

$$\forall x(isOperational(x, functionOf(x) \rightarrow \mathcal{E}(x)). \tag{6.3}$$

Therefore, in order for a device to function and be operational it is essentially depended on some parts contributing to its behaviour. These are called *functional essential parts*. For other behaviours additional parts might be used but they are not necessary for the device to be operative. The latter are called *functional optional parts*. Hereafter, we will omit the term functional referring only to essential and optional parts. In order further understand the distinction we shall present the following example.

Suppose that we would like to buy a certain device from our local shop. This device is composed of several smaller devices that are considered to be its essential parts. These parts are defined by the manufacturer and accompany the device in every other event, such as in the case of a service. However, there might be other parts that are not sold with the device but have to be bought separately and are later embedded on the device for a specific function. These are the optional parts. But before defining the two relations we should define the two functional dependences between two components as follows:

Definition 10. *(**essential functional dependency**): A component x is functionally dependent on another component y iff, as a matter of necessity, y is operational and exists whenever x is operational and exists:*

$$\begin{aligned} efd(x, y) \equiv \Box(\mathcal{E}(y) \wedge isOperational(y, functionOf(y) \rightarrow \\ \mathcal{E}(x) \wedge isOperational(x, functionOf(x)) \end{aligned} \tag{6.4}$$

Definition 11. *(**optional functional dependency**): A component x is functionally dependent on another component y iff, as a matter of possibility, y is operational and exists whenever x is operational and exists:*

$$\begin{aligned} ofd(x, y) \equiv \Diamond(\mathcal{E}(y) \wedge isOperational(y, functionOf(y) \rightarrow \\ \mathcal{E}(x) \wedge isOperational(x, functionOf(x)) \end{aligned} \tag{6.5}$$

We are now ready to define more formally the essential and optional parthood.

Definition 12. *(**essential part**): A component x is an essential part of another component y iff, as a matter of necessity, y is operational and exists whenever x has a function, exists, and is part of y, or formally:*

$$EP(x, y) \equiv efd(x, y) \wedge \Box partOf(x, y)) \tag{6.6}$$

Definition 13. *(**optional part**):A component x is an optional part of another component y iff, as a matter of possibility, y is operational and exists whenever x exists, or formally:*

$$OP(x,y) \equiv ofd(x,y) \wedge \Diamond partOf(x,y)) \tag{6.7}$$

The essential parts of the device are necessary in order for the device to be operational. These parts also determine the identity of the device and coexist with it. However, the optional parts might exist for a specific time-span of the device's existence and they must be related with some events of part addition or part removal. In general, the time-span of the optional parts is defined with respect to the time-spans of the aforementioned events (see section for more details). The components are described purely in terms of their functions. This makes it possible in principle to replace components by structurally different but functionally identical components. Furthermore, the components themselves can be represented as devices in their own terms (Chandrasekaran, 1994). Otherwise, without these parts the device cannot operate.

One of the most fundamental defined properties in CIDOC CRM for representing a parthood relationship among physical things is the *P46 is composed of (forms part of)*. The property is related to instances of the class *E18 Physical Thing*. The E18 class comprises all persistent physical items with a relatively stable form, man-made or natural.

CIDOC CRM's property P46 is a property which is (a) reflexive, (b) transitive and (c) antisymmetric following the axioms of *Ground Mereology* (see Section 2.3.2). Thus, objects are composed of other subcomponents which are assumed to also be modelled as objects themselves holding a parthood relationship with the former ones, thereby creating a hierarchy of part decompositions.

In order to support the above notion of essential and optional of parthood relation in CIDOC CRM, we extend the *P46 is composed of (forms part of)* property by introducing two more properties which are sub properties of the P46 one:

- is essentially composed of (essentially forms part of)
- optionally composed of (optionally forms part of)

Figure 6.9 depicts the above modelling.

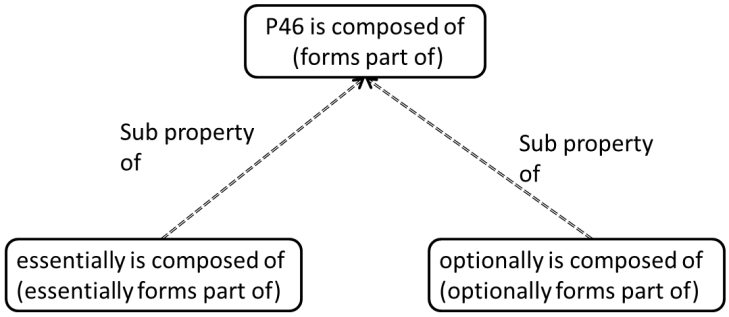


Figure 6.9: Subproperties of P46 forms part of

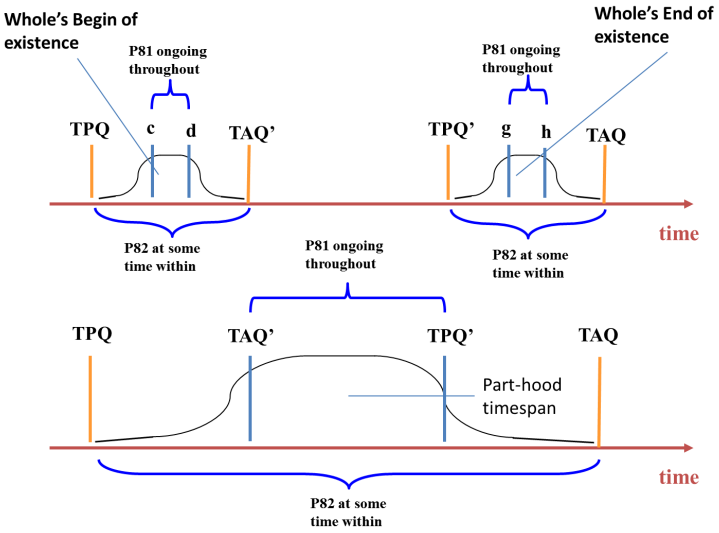


Figure 6.10: Time spans of essential parts

6.3.2 Temporal Essential and Optional Parts

In the section 6.2.2 we reviewed how chronological information about events can be expressed in CIDOC CRM. In this section, we discuss how time-spans of essential and optional objects can be defined.

Temporal Essential Parts

Essential parts, as it has been already discussed, exist with their whole that holds a parthood relationship by default. Thus, their time spans are related to the events of begin of existence and the end of existence when its whole was generated and destroyed. Every such event, according to the modelling of CIDOC CRM, has two boundaries (P81 ongoing throughout) and two (P82 at some time within) ones which symbolize the certainty and the uncertainty of the exact timing of the event's occurrence respectively.

The TPQ of an essential parthood time-span's outer interval corresponds to that of its whole's probable begin of existence and the TAQ to that of its whole's probable end of existence. The boundaries of its inner interval, i.e., TAQ' and TPQ', are the TAQ of its whole's certain begin of existence and the TPQ of its whole's certain end of existence, see Fig. 6.10.

We assume that the part has started to exist before the probable end of begin of existence event (i.e., before the TAQ' boundary) and end to exist after the end of end of existence event (i.e., after the TAQ boundary).

Thus, the uncertainty of the parthood's time-span includes the uncertainty of the occurrence of events from which the whole's existence is depended on (i.e. begin of existence and end of existence). Though, the certainty that a part holds an essential parthood is limited after the TAQ' certain begin of existence event and before the TPQ' of end of existence event.

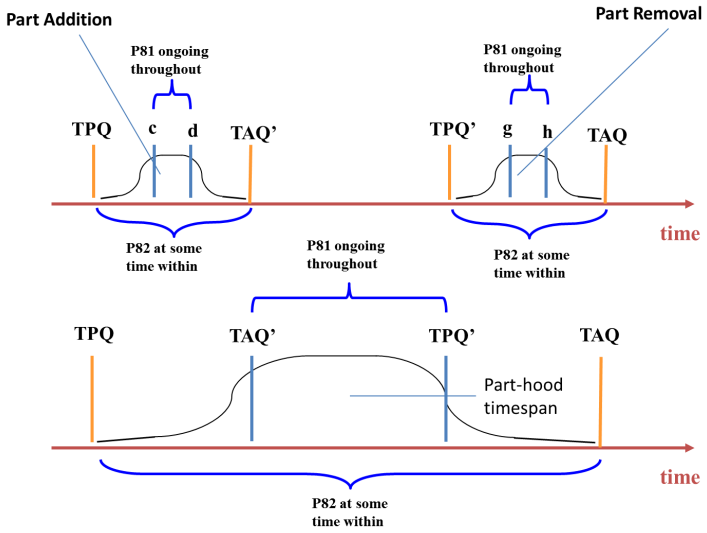


Figure 6.11: Time spans of optional parts (1st case)

Temporal Optional Parts

In order for a part to be characterized as optional, it must be related to events of part addition or part removal. Otherwise, it is an essential by default. We consider two cases:

- the part was added but then removed from the whole
- the part was added but remained in the whole

In the first case, the part was added with a part addition event and then removed with a part removal event. A part can be optional or non-optional, or essential, according to the existence of the above events. If there is at least a part-addition event related to that part then we assume that this part is an optional one. The non existence of such an event implies that the particular part is an essential one. As a consequence, we define the time span of optional parts in relation to the ones of the above events. The TPQ of an optional parthood time-span's outer interval is the corresponding one of the part addition event and the TAQ of the part removal one. Its inner interval starts with the certain beginning of the part addition's event (i.e., the TAQ' boundary) and ends with the certain ending of the part removal's event (i.e., the TPQ' boundary). Fig. 6.11 might be helpful in order for this reasoning to be understood.

In the second case, the part was only involved in a part addition event and so it was not removed until the end of existence of the whole. As a result, we have a similar mapping to the temporal boundaries as in the previous case but now since the relation holds until the end of the existence of its whole, we replace the part removal event with the whole's end of existence event. The replacement is depicted in Fig. 6.12.

Considering the possibility that our knowledge base may contain multiple events of part addition and part removal, then a part can have multiple parthood time-

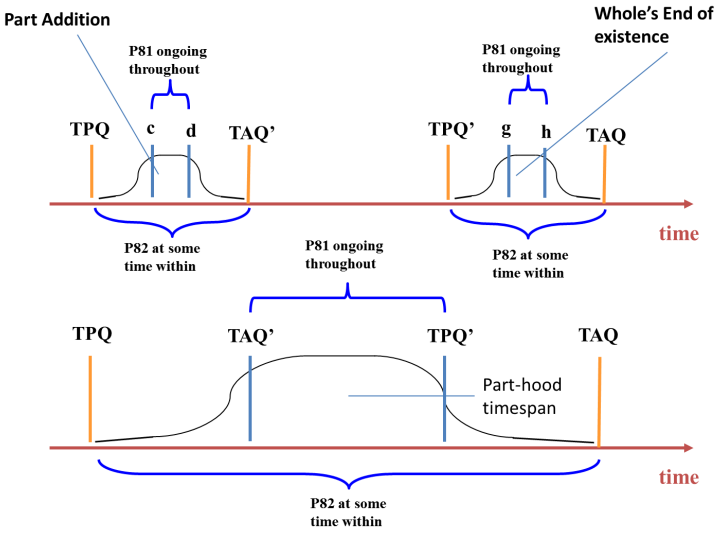


Figure 6.12: Time spans of optional parts (2nd case)

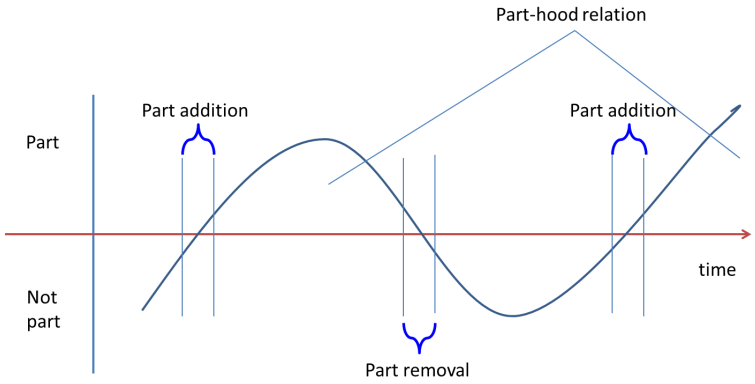


Figure 6.13: Multiple part additions and removals

spans related again to these events. The number of these events is actually the number of the part additions. This idea is illustrated in Fig. 6.13.

In Fig. 6.13 we can observe the two part addition events that are related to an optional part and the parthood relations that exist in between. After a part addition event the relation holds until a possible part removal one. However, during a part removal and a part addition event the optional parthood relation does not hold.

6.4 Summary

The main considerations of this Chapter are the challenge of how a larger set of rules can be defined and the refinement of the parthood relation of objects by defining whether they are essential or optional. Regarding the first challenge, we introduced

two more reasoning rules and showed a simple pattern that could be identified in models and ontologies in order for other similar rules to be defined. Regarding the second challenge, we defined how parts can be distinguished into essential and optional when are part of wholes. We formally defined several dependencies of the essential and optional parts, and we extended this reasoning by taking into account the temporal aspects of such relationships.

Chapter 7

Related Work

This Chapter studies the related current work focusing on works regarding the techniques for reducing the provenance storage (Section 7.2), reasoning mechanisms applied on provenance information (Section 7.1) and the problem of knowledge evolution (Section 7.3). We compare these works and ideas related with our approach. The existing limitations of the former suggest the advantages of our approach.

7.1 Inference Rules

Our approach is based on the implicit knowledge that can be derived by inference rules. Nonetheless, we should clarify that the implicit knowledge considered in this work is different than tacit knowledge (Polanyi, 1966; Smith, 2003); the former has some grounds on explicit one, whereas the latter may be the result of common sense or common knowledge. Tacit knowledge is considered implicit and is distinguished from the explicit by the fact that tacit knowledge can be acquired through experience ("know-how"). Our considered implicit knowledge, however, is based solely on facts ("know-what").

In addition, the reasoning forms that we consider in this work, aim at the dynamic completion (deduction) of facts from original input by resolving transitive closures and propagating the property instances at query time, rather than at ingestion time. This is complementary to reasoning on "data provenance", which traces causal dependencies of individual elements of data sets between input and output. In the latter category we should mention the works of (Moreau & Missier, 2011) and (Moreau *et al.*, 2011), two formal models representing provenance information. The inference rules defined in (Moreau & Missier, 2011; Moreau *et al.*, 2011) are focused on the derivation of further causal dependencies between processes and artifacts and not the propagation of features or properties among entities.

Lastly, inference rules with annotations are exploited in (Bonatti *et al.*, 2011) for scalable reasoning on web data. Even though these annotations are indicators of data provenance, they do not directly model the latter.

7.2 Provenance Storage

The problem of efficient storage of provenance information has been extensively recognized in literature. Different methods have been presented for reducing space storage requirements of provenance information. For example, provenance minimization via polynomials has been studied in (Amsterdamer *et al.*, 2011). The authors study algorithms in order to find the "core" minimal provenance of tuples in the results of equivalent queries. Even though, this work might be useful in database operations, it is not focused on provenance workflow information (as our work) which is a different problem.

Another example is (Heinis & Alonso, 2008) in which workflow DAGs are transformed into interval encoded tree structures. Furthermore, similar to our notion of property propagation, (Chapman *et al.*, 2008) proposes provenance to inheritance methods assuming a tree-form model. Moreover, it should be noted that in (Chapman *et al.*, 2008) DAGs are not taken into account. However, DAGs is a more general and more complicated kind of a graph morphology than trees in which a workflow graph might result. Moreover, the works of Heinis & Alonso (2008) and Chapman *et al.* (2008) are applied in information already stored. This is less advantageous in comparison to our approach which can be applied before the information ingestion and its storage.

Additional techniques have been proposed in Anand *et al.* (2009). The authors present a more general model than CIDOC CRM under the assumption that output data may depend on some not all input data and that many scientific workflows are based on this assumption. That model mainly captures only the dependencies among data and taking into consideration the fact of being more general than CIDOC CRM, it may suffer from over generalization and possible loss of information. On the other hand, CIDOC CRM is a rich conceptual model capable of describing provenance under a user-defined level of granularity, since its design is based on class and property hierarchy. Nevertheless, Anand *et al.* (2009) also uses several inference rules for collapsing provenance traces.

7.3 Knowledge Evolution in RDF/S

The research field of ontology evolution (Gabel *et al.*, 2004) deals with updating knowledge in ontologies; a detailed survey of the field appears in (Flouris *et al.*, 2008). However, none of these works considers custom inference rules.

Some works (e.g., (Stojanovic & Motik, 2002)) address the problem using ontology editors. However, it has been argued (Stojanovic & Motik, 2002) that the naive set-theoretical application of changes that takes place in most editors is insufficient, because, first, it introduces a huge manual burden to the ontology engineer, and, second, it does not consider the standard inference semantics of RDF/S and other ontological languages.

In response to this need, works like (Bechhofer *et al.*, 2001; Gabel *et al.*, 2004; Noy *et al.*, 2000; Sure *et al.*, 2003; Klein & Noy, 2003) have proposed and implemented change semantics which consider the standard inference rules of RDF/S and

determine, for each type of change request, the side-effects necessary to properly execute said change taking into account the inference semantics. However, these works do not consider custom inference rules and do not discriminate between coherence and foundational semantics for their change operations (they only consider foundational semantics). In certain cases, some flexibility is provided to independently customize the semantics of some of the operations (Gabel *et al.*, 2004). Similarly, in (Lösch *et al.*, 2009), one can explicitly define the semantics of change operators.

Some works deal with all change operations in a generic manner. For example, (Magiridou *et al.*, 2005) proposes a declarative approach which can handle all possible changes on the data part of an RDF/S KB. Under this approach, operations (and their effects) are generically modelled, avoiding the need to define a specific operation for each type of change request. A similar, and more generic, approach which can handle both schema and data changes in a generic manner, appears in (Konstantinidis *et al.*, 2008). These works consider only the standard inference rules of RDF/S (but (Magiridou *et al.*, 2005) can be extended to support also custom inference rules) and only the coherence semantics (i.e., there is no support for the foundational semantics).

Our future plans include the generalization of the approach presented in (Konstantinidis *et al.*, 2008) in order to support the foundational semantics as well; this will allow us to apply the updates according to the semantics described earlier, but in a generic manner, without having to resort to the specific update plan of each operation (which would emerge as special cases of the generic approach).

Finally, we should mention (Gutierrez *et al.*, 2011) which elaborated on the deletion of triples (including inferred ones) assuming the standard inference rules of RDF/S for both schema and instance update focusing on the "erase" operation. This work also considers generic operations, but only for the case of removing information. The considered semantics is coherence semantics (only), and the authors describe how one can compute all the "optimal" plans for executing such an erase operation (contraction in our terminology), without resorting to specific-per-operation update plans.

7.4 Summary

This study focuses on the examination of the introduction of inference mechanisms in order to reduce the amount of provenance information produced by workflow systems. The inference rules considered are different from just the derivation of further dependencies among data, as currently proposed by several other provenance models. They are based on the propagation of attributes in hierarchical structures of data. Moreover, these are assumed before the ingestion of data, contrary to some works that minimize already ingested and stored provenance information which is less advantageous to our approach.

In the second part, we define the semantics of update operations for the problem of knowledge evolution, taking into account two approaches epistemological value of the inferred knowledge and the considered inference rules. This presents us with a novelty since current works do not consider both approaches or take into consider-

ation any epistemological assumptions. Moreover, they take into account only the RDF/S inference rules but not any user defined custom inference rules which may include more complex inferences.

Chapter 8

Concluding Remarks

This Chapter actually concludes this thesis. In Section 8.1 we further discuss some of the assumptions adopted in this study and explain the consequences of ignoring one or more of these assumptions. Section 8.2 summarizes the results of this study and suggests directions for further research.

8.1 On Applicability

The purpose of this section is to discuss some of the hypotheses used in this study. For instance, the general assumption endorsed by this study is provenance information which has been recorded by empirical evidence (Mudge *et al.*, 2008) and we may distinguish three epistemological situations:

1. The facts can reliably and completely be registered by a monitoring system, such as a workflow shell.

2. There are facts which users need to input manually to the monitoring system and may not be willing to do so.

3. Facts come from different monitoring systems or uncontrolled human input.

From a semantic point of view, the rules are reasonable with respect to the semantics of the classes and properties defined in the provenance model. For example, we assume, by convention, that a person carrying out a process carries out all of its sub-processes; this is reflected by the formal phrasing of the corresponding rule.

Alternative rules could also be reasonable, e.g., a rule stating that a person carrying out a process carries out at least one of its sub-processes or that a person carrying out a process, also carried out all of its super-processes. That of course would give a different semantics to the "carries out" property and would affect, consequently the modelling of evolution. More specifically, in the second case, the semantics of the property "carry out" implies more a presence than a responsibility of the actors to the respective activities and it might be replaced by a more appropriate property (e.g., "was present at").

Even though we assume a specific set of inference rules, a different set of rules could also be reasonable and applicable. From a practical point of view, each rule

causes the derivation of additional knowledge but a very large set of rules might introduce other difficulties such as further dependencies among the rules affecting the update operations (e.g., when an addition or deletion causes cascading violations of other rules, which result in more side-effects for each operation (Flouris *et al.*, 2012)) and result to a more complex reasoning system.

Another assumption is that the user/process responsible for the ingestion has an a priori knowledge of the model and the considered set of inference rules. If we ignore this assumption, there might be redundant facts in the KB that can be derived by the application of the rules. This redundancy would only attribute to repetitive facts and to larger amount of space. In the case that the foundational viewpoint is regarded, these facts are valuable explicit information, while in other cases a reduction algorithm would be needed in order to eliminate such facts. Although the first case does not exploit the full benefits of our approach, with respect to the storage space requirements, it should be noted that these cases do not affect the correctness of our proposed algorithms for the update operations.

Regarding the problem of knowledge evolution, we defined the semantics of our update operation having in mind the exceptional cases that might occur; for example the activity delegation to different actors. Since we assume the default application of our rules, the support of handling such cases is significant. Yet another assumption might be that our rules are not always valid, but their application depends on some other condition or fact that needs to exist in the KB. However, this exercise is beyond the scope of this study.

8.2 Synopsis and Future Work

In this paper, we argued for the need for provenance-based inference aiming at the dynamic completion (deduction) of facts from the original input in order to reduce the storage space requirements. The inference rules considered are different from simply the derivation of further dependencies among data, as currently proposed by other provenance models. They are based on the propagation of attributes in hierarchical structures of data. This dynamic propagation may also propagate possible errors in the KB, resulted from the initial ingested data input. However, they can be easily corrected since they are only attributed to the original (explicit) data input and thus the search space for their identification has been reduced. Instead of searching the whole KB for errors, an examination of the ingested facts is sufficient.

The application of inference rules introduces difficulties with respect to the evolution of knowledge; we elaborated on these difficulties and described how we can address this problem. We identified two ways to deal with deletions in this context, based on the philosophical stance against explicit (ingested) knowledge and implicit (inferred) one (foundational and coherence semantics). In this regard, we elaborate on the specific algorithms for these operations respecting the application of our custom rules. This presents us with two novelties, since (a) current works do not consider both approaches and (b) they take into account only the RDF/S inference rules but not any user defined custom inference rules.

Based on these ideas, we specified a number of update operations that allow

knowledge updating under said inference rules. Although we confined ourselves to three specific inference rules, the general ideas behind our work (including the discrimination between foundational and coherence semantics of deletion) can be applied to other models and/or sets of inference rules.

We conducted experiments on real and synthetic data comparing two policies, (a) storing all the inferences from the rules and (b) storing only the initial ingested facts (i.e., our considered approach), that could be adopted by a metadata environment. The comparison was made in relation to the time performance of queries and update operations, and the storage space requirements for each policy. The results showed that although, the query time performance was slightly increased in the case of the first policy compared to the second one. Nevertheless, the latter was very beneficial not only for reducing the storage space requirements, but also for achieving better time performance of update and maintenance operations.

A next step would be the examination of a larger set of inference rules and the respective update operations. In this regard our plan is to study the problem in a more generic manner, in order to deal with change operations without having to resort to specific, per-operation update plans, in the spirit of (Konstantinidis *et al.*, 2008).

Bibliography

Alchourrón, C. E., Gärdenfors, P., & Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *J. symb. log.*, 510–530.

Amsterdamer, Y., Deutch, D., Milo, T., & Tannen, V. 2011. On provenance minimization. *Pages 141–152 of:* Lenzerini, Maurizio, & Schwentick, Thomas (eds), *Pods*. ACM.

Anand, M. K., Bowers, S., McPhillips, T., & Ludäscher, B. 2009. Efficient provenance storage over nested data collections. *Pages 958–969 of: Proceedings of the 12th international conference on extending database technology: Advances in database technology*. EDBT '09. ACM.

Antoniou, G. 1997. *Nonmonotonic reasoning*.

Antoniou, G., Damásio, C. V., Grosof, B., Horrocks, I., Kifer, M., Ma, J., & Peter, F. 2005. *Combining rules and ontologies. a survey.*

Artale, A., Franconi, E., & Guarino, N. 1996. Open problems with part-whole relations. *Pages 70–73 of: Description logics'96*.

Bechhofer, S., Horrocks, I., Goble, C., & Stevens, R. 2001. Oiled: A reason-able ontology editor for the semantic web. *Pages 396–408 of: Proceedings of the joint german/austrian conference on ai: Advances in artificial intelligence*. KI '01.

Bizer, C., & Schultz, A. 2008. Benchmarking the performance of storage systems that expose sparql endpoints. *In: In proceedings of the iswc workshop on scalable semantic web knowledgebase.*

Bonatti, P. A., Hogan, A., Polleres, A., & Sauro, L. 2011. Robust and scalable linked data reasoning incorporating provenance and trust annotations. *Web semantics: Science, services and agents on the world wide web*, **9**(2).

Chandrasekaran, B. 1994. Functional representation: A brief historical perspective. *Applied artificial intelligence*, 173–197.

Chandrasekaran, B., & Josephson, J. 2000. Function in device representation. *Engineering with computers*, **16**(3-4), 162–177.

Chang, C.C., & Keisler, H.J. 1990. *Model theory*. Studies in logic and the foundations of mathematics. North-Holland.

Chapman, A., Jagadish, H.V., & Ramanan, P. 2008. Efficient provenance storage. *In: Proc. of the acm sigmod/pods conference.*

Crofts, N., Doerr, M., Gill, T., Stead, S., & Stiff, M. 2011 (November). *Definition of the cidoc conceptual reference model.*

Dalal, M. 1988. Investigations into a theory of knowledge base revision. AAAI Press / The MIT Press.

Doerr, M. 2003. The cidoc conceptual reference module: an ontological approach to semantic interoperability of metadata. *Ai mag.*, **24**(September), 75–92.

Doerr, M., Plexousakis, D., Kopaka, K., & Bekiari, C. 2004. Supporting chronological reasoning in archaeology. *Pages 13–17 of: Computer applications and quantitative methods in archaeology conference, caa2004.*

Erling, O., & Mikhailov, I. 2009. *Sparql and scalable inference on demand.* "http://virtuoso.openlinksw.com/whitepapers/SPARQL

Fine, K. 1994. Essence and modality. *Philosophical perspectives*, **8**, 1–16.

Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., & Antoniou, G. 2008. Ontology change: Classification and survey. *The knowledge engineering review*, **23**(02), 117–152.

Flouris, Giorgos, Konstantinidis, George, Antoniou, Grigoris, & Christophides, Vassilis. 2012. Formal foundations for rdf/s kb evolution. *International journal on knowledge and information systems (kais)*, 1–39.

Gabel, T., Sure, Y., & Völker, J. 2004. *Kaon - ontology management infrastructure.* SEKT informal deliverable D3.1.1.a.

Gantz, J., & Reinsel, D. 2011 (June). *Extracting value from chaos.*

Gärdenfors, P. 1988. *Knowledge in flux. modelling the dymanics of epistemic states.* Mit Press.

Gärdenfors, P. 1992. Belief revision: An introduction.

Gerstl, P., & Pribbenow, S. 1995. Midwinters, end games, and body parts: a classification of part-whole relations. *Int. j. hum.-comput. stud.*, **43**(5-6), 865–889.

Gerstl, P., & Pribbenow, S. 1996. A conceptual theory of part-whole relations and its applications. *Data knowl. eng.*, **20**(3), 305–322.

Gorman, M. 2005. The essential and the accidental. *Ratio*, **18**(3), 276–289.

Gruber, T. R. 1995. Toward principles for the design of ontologies used for knowledge sharing. *Int. j. hum.-comput. stud.*, **43**(5-6), 907–928.

Guarino, N. 1998. *Formal ontology in information systems: Proceedings of the 1st international conference june 6-8, 1998, trento, italy*. 1st edn. Amsterdam, The Netherlands, The Netherlands: IOS Press.

Guizzardi, G. 2009. The problem of transitivity of part-whole relations in conceptual modeling revisited. *Pages 94–109 of: Proceedings of the 21st international conference on advanced information systems engineering*. CAiSE '09. Berlin, Heidelberg: Springer-Verlag.

Gutierrez, C., Hurtado, C., & Mendelzon, A. O. 2004. Foundations of semantic web databases. *Pages 95–106 of: Proceedings of the twenty-third acm sigmod-sigact-sigart symposium on principles of database systems*. PODS '04. New York, NY, USA: ACM.

Gutierrez, C., Hurtado, C. A., & Vaisman, A. A. 2011. Rdfs update: From theory to practice. *Pages 93–107 of: Eswc (2)*.

Harris, S., & Seaborne, A. 2010. Sparql 1.1 query language. *W3c working draft*.

Heinis, T., & Alonso, G. 2008. Efficient lineage tracking for scientific workflows. *Pages 1007–1018 of:* Wang, Jason Tsong-Li (ed), *Sigmod conference*. ACM.

Holmen, J., & Ore, C.E. 2009. *Deducing event chronology in a cultural heritage documentation system*.

Johansson, I. 2004. On the transitivity of the parthood relations. *Pages 161–181 of:* Hochberg, H., & (eds.), K. Mulligan (eds), *Relations and predicates*. Ontos Verlag.

Klein, M., & Noy, N. 2003. A component-based framework for ontology evolution. *In: Proc. workshop on ontologies and distributed systems, ijcai 2003 (acapulco, mexico)*.

Konstantinidis, G., Flouris, G., Antoniou, G., & Christophides, V. 2008. A formal approach for rdf/s ontology evolution. *Pages 70–74 of: Ecai*.

Lösch, U., Rudolph, S., Vrandečić, D., & Studer, R. 2009. Tempus fugit - towards an ontology update language. *Pages 278–292 of: 6th european semantic web conference (eswc 09)*.

Maedche, A., Motik, B., Stojanovic, L., & Stojanovic, N. 2002. User-driven ontology evolution management. Springer-Verlag.

Magiridou, M., Sahtouris, S., Christophides, V., & Koubarakis, M. 2005. Rul: A declarative update language for rdf. *Pages 506–521 of: International semantic web conference*.

M.Doerr. 2011. *Mets and cidoc crm - a comparison*. http://culturalheritageimaging.org/What_We_Do/Publications/mets-crm-doerr/mets_crm.pdf.

Moreau, L. 2010. The foundations for provenance on the web. *Foundations and trends in web science*, **2**(2–3), 99–241.

Moreau, L., & Missier, P. 2011 (October). *The prov data model and abstract syntax notation*. http://www.w3.org/TR/2011/WD-prov-dm-20111018/.

Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P.T., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E. G., & den Bussche, J. V. 2011. The open provenance model core specification (v1.1). *Future generation comp. syst.*, **27**(6), 743–756.

Mudge, M., Malzbender, T., Chalmers, A., Scopigno, R., Davis, J., Wang, O., Gunawardane, P., Ashley, M., Doerr, M., Proenca, A., & Barbosa, J. 2008. Image-Based Empirical Information Acquisition, Scientific Reliability, and Long-Term Digital Preservation for the Natural Sciences and Cultural Heritage .

Noy, N., Fergerson, R., & Musen, M. 2000. The knowledge model of protégé-2000: Combining interoperability and flexibility. *Pages 17–32 of: Proceedings of the 12th european workshop on knowledge acquisition, modeling and management*. EKAW '00. Springer-Verlag.

Polanyi, M. 1966. *The tacit dimension*. Garden City, NY: Doubleday.

RidingTheWave. 2010 (October). *Riding the wave - how europe can gain from the rising tide of scientific data*.

Romano, P. 2008. Automation of in-silico data analysis processes through workflow management systems. *Briefings in bioinformatics*, **9**(1), 57–68.

Russell, S.J., & Norvig, P. 2010. *Artificial intelligence: A modern approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall.

Salza, S., Guercio, M, Grossi, M., Pröll, S., C.Stroumboulis, Tzitzikas, Y., Doerr, M., & Flouris, G. 2012. *D24.1 report on authenticity and plan for interoperable authenticity evaluation system*.

Simmhan, Y. L., Plale, B., & Gannon, D. 2005. A survey of data provenance in e-science. *Sigmod rec.*, **34**(3), 31–36.

Smith, M. K. 2003. Michael polanyi and tacit knowledge. *The encyclopedia of informal education*.

Sosa, E. 1980. The raft and the pyramid: Coherence versus foundations in the theory of knowledge. *Midwest studies in philosophy*, **5**(1), 3–26.

Stojanovic, L., & Motik, B. 2002. Ontology evolution within ontology editors. *Pages 53–62 of: Ekaw'02/eon workshop*.

Stuckenschmidt, Heiner, & Klein, Michel. 2003. *Integrity and change in modular ontologies*.

Sure, Y., Angele, J., & Staab, S. 2003. Ontoedit: Multifaceted inferencing for ontology engineering. *J. data semantics*, **1**, 128–152.

Teresa, R. 2008. Essential vs. accidental properties. *In:* Zalta, Edward N. (ed), *The stanford encyclopedia of philosophy*, fall 2008 edn.

Thakker, D., Osman, T., Gohil, S., & Lakin, P. 2010. A pragmatic approach to semantic repositories benchmarking. *Pages 379–393 of: Proceedings of the 7th international conference on the semantic web: research and applications - volume part i*. ESWC'10. Berlin, Heidelberg: Springer-Verlag.

Theodoridou, M., Tzitzikas, Y., Doerr, M., Marketakis, Y., & Melessanakis, V. 2010. Modeling and querying provenance by extending cidoc crm. *Distributed and parallel databases*, **27**, 169–210.

Theoharis, Yannis, Georgakopoulos, George, & Christophides, Vassilis. 2012. Powergen: A power-law based generator of rdfs schemas. *Inf. syst.*, **37**(4).

Tversky, B. 1986. *Components and categorization*. John Benjamins Publishing Company. Pages 63–75.

Varzi, A. 1996. Parts, wholes, and part-whole relations: The prospects of mereotopology. *Data and knowledge engineering*, **20**, 259–286.

Zeginis, D., Tzitzikas, Y., & Christophides, V. 2011. On computing deltas of rdf/s knowledge bases. *Acm trans. web*, **5**, 14:1–14:36.

Zhao, J., Goble, C., Stevens, R., & Bechhofer, S. 2004. Semantically linking and browsing provenance logs for e-science. *Pages 158–176 of:* Bouzeghoub, Mokrane, Goble, Carole, Kashyap, Vipul, & Spaccapietra, Stefano (eds), *Semantics of a networked world*. Lecture Notes in Computer Science, vol. 3226. Springer Berlin / Heidelberg.

Appendix A

Formal Definition of Classes and Properties

A.1 Classes

E5 Event
Subclass of: E4 Period
Superclass of: E7 Activity
E63 Beginning of Existence
E64 End of Existence

Scope note:
This class comprises changes of states in cultural, social or physical systems, regardless of scale, brought about by a series or group of coherent physical, cultural, technological or legal phenomena. Such changes of state will affect instances of E77 Persistent Item or its subclasses.

The distinction between an E5 Event and an E4 Period is partly a question of the scale of observation. Viewed at a coarse level of detail, an E5 Event is an 'instantaneous' change of state. At a fine level, the E5 Event can be analysed into its component phenomena within a space and time frame,
and as such can be seen as ans E4 Period.
The reverse is not necessarily the case: not all instances of E4 Period give rise to a noteworthy change of state.

Examples:
the birth of Cleopatra (E67)
the destruction of Herculaneum by volcanic eruption in 79 AD (E6)
World War II (E7)
the Battle of Stalingrad (E7)
the Yalta Conference (E7)
my birthday celebration 28-6-1995 (E7)

the falling of a tile from my roof last Sunday
the CIDOC Conference 2003 (E7)

Properties:
P11 had participant (participated in): E39 Actor
P12 occurred in the presence of (was present at): E77 Persistent Item

E7 Activity
Subclass of: E5 Event
Superclass of: E8 Acquisition
E9 Move
E10 Transfer of Custody
E11 Modification
E13 Attribute Assignment
E65 Creation
E66 Formation
E85 Joining
E86 Leaving
E87 Curation Activity

Scope note:
This class comprises actions intentionally carried out by instances of E39 Actor that result in changes of state in the cultural, social, or physical systems documented. This notion includes complex, composite and long-lasting actions such as the building of a settlement or a war, as well as simple, short-lived actions such as the opening of a door.

Examples:
the Battle of Stalingrad
the Yalta Conference
my birthday celebration 28-6-1995
the writing of "Faust" by Goethe (E65)
the formation of the Bauhaus 1919 (E66)
calling the place identified by TGN '7017998' 'Quyunjig' by the people of Iraq

E22 Man-Made Object
Subclass of: E19 Physical Object
E24 Physical Man-Made Thing
Superclass of: E84 Information Carrier

Scope note:
This class comprises physical objects purposely created by human activity. No assumptions are made as to the extent of modification required to justify regarding an object as man-made. For example, an inscribed piece of rock or

a preserved butterfly are both regarded as instances of E22 Man-Made Object.

Examples:
Mallard (the World's fastest steam engine)
the Portland Vase
the Coliseum

E24 Physical Man-Made Thing
Subclass of: E18 Physical Thing
E71 Man-Made Thing
Superclass of: E22 Man-Made Object
E25 Man-Made Feature
E78 Collection

Scope Note:
This class comprises all persistent physical items that are purposely created by human activity.
This class comprises man-made objects, such as a swords, and man-made features, such as rock art. No assumptions are made as to the extent of modification required to justify regarding an object as man-made.
For example, a "cup and ring" carving on bedrock is regarded as instance of E24 Physical Man-Made Thing.

Examples:
the Forth Railway Bridge (E22)
the Channel Tunnel (E25)
the Historical Collection of the Museum Benaki in Athens (E78)

E39 Actor
Subclass of: E77 Persistent Item
Superclass of: E21 Person
E74 Group
Scope note:
This class comprises people, either individually or in groups, who have the potential to perform intentional actions for which they can be held responsible.
The CRM does not attempt to model the inadvertent actions of such actors. Individual people should
be documented as instances of E21 Person, whereas groups should be documented as instances of either E74 Group or its subclass E40 Legal Body.

Examples:
London and Continental Railways (E40)
the Governor of the Bank of England in 1975 (E21)

Sir Ian McKellan (E21)

E73 Information Object
Subclass of: E89 Propositional Object
E90 Symbolic Object
Superclass of: E29 Design or Procedure
E31 Document
E33 Linguistic Object
E36 Visual Item

Scope note:
This class comprises identifiable immaterial items, such as a poems, jokes, data sets, images, texts, multimedia objects, procedural prescriptions, computer program code, algorithm or mathematical formulae, that have an objectively recognizable structure and are documented as single units.

An E73 Information Object does not depend on a specific physical carrier, which can include human memory, and it can exist on one or more carriers simultaneously. Instances of E73 Information Object of a linguistic nature should be declared as instances of the E33 Linguistic Object subclass. Instances of E73 Information Object of a documentary nature should be declared as instances of the E31 Document subclass. Conceptual items such as types and classes are not instances of E73 Information Object, nor are ideas without a reproducible expression.

Examples:
image BM000038850.JPG from the Clayton Herbarium in London
E. A. Poe's "The Raven"
the movie "The Seven Samurai" by Akira Kurosawa
the Maxwell Equations

A.2 Properties

P9 consists of (forms part of)
Domain: E4 Period
Range: E4 Period
Quantification: one to many, (0,n:0,1)

Scope note:
This property describes the decomposition of an instance of E4 Period into dis-

crete, subsidiary periods.
The sub-periods into which the period is decomposed form a logical whole - although the entire picture may not be completely known - and the sub-periods are constitutive of the general period.

Examples:
Cretan Bronze Age (E4) consists of Middle Minoan (E4)

P12 occurred in the presence of (was present at)
Domain: E5 Event
Range: E77 Persistent Item
Superproperty of:
E5 Event. P11 had participant (participated in): E39 Actor
E7 Activity. P16 used specific object (was used for): E70 Thing
E9 Move. P25 moved (moved by): E19 Physical Object
E11 Modification. P31 has modified (was modified by): E24 Physical Man-Made Thing
E63 Beginning of Existence. P92 brought into existence (was brought into existence by):
E77 Persistent Item
E64 End of Existence. P93 took out of existence (was taken out of existence by):
E77 Persistent Item
E79 Part Addition.P111 added (was added by): E18 Physical Thing
E80 Part Removal.P113 removed (was removed by): E18 Physical Thing
Quantification: many to many, necessary (1,n:0,n)

Scope note:
This property describes the active or passive presence of an
E77 Persistent Item in an E5 Event without implying any specific role.
It connects the history of a thing with the E53 Place and E50 Date of an event.
For example, an object may be the desk, now in a museum on which a
treaty was signed. The presence of an immaterial thing implies the presence of at least one of its carriers.

Examples:
Deckchair 42 (E19) was present at The sinking of the Titanic (E5)

P14 carried out by (performed)
Domain: E7 Activity
Range: E39 Actor
Subproperty of: E5 Event. P11 had participant (participated in): E39 Actor
Superproperty of:
E8 Acquisition. P22 transferred title to
(acquired title through): E39 Actor
E8 Acquisition. P23 transferred title from
(surrendered title through): E39 Actor
E10 Transfer of Custody. P28 custody surrendered by (surrendered custody

through): E39 Actor
E10 Transfer of Custody. P29 custody received by (
received custody through):
E39 Actor
Quantification: many to many, necessary (1,n:0,n)

Scope note:
This property describes the active participation of an E39 Actor in an E7 Activity.
It implies causal or legal responsibility.
The P14.1 in the role of property of the property allows the nature of an Actor's participation to be specified.

Examples:
the painting of the Sistine Chapel (E7) carried out by Michaelangelo Buonaroti (E21) in the role of master craftsman (E55)
Properties: P14.1 in the role of: E55 Type

P128 carries (is carried by)
Domain: E24 Physical Man-Made Thing
Range: E90 Symbolic Object
Subproperty of: E70 Thing.P130 shows features of (features are also found on): E70 Thing
Superproperty of:
E24 Physical Man-Made Thing. P65 shows visual item (is shown by): E36 Visual Item
Quantification: many to many (0,n:0,n)

Scope note:
This property identifies an E73 Information Object carried by an instance of E24 Physical Man-Made Thing.
In general this would be an E84 Information Carrier P65 shows visual item (is shown by) is a specialisation of P128 carries (is carried by) which should be used for carrying visual items.

Examples:
Matthew's paperback copy of Reach for the Sky (E84) carries the text of Reach for the Sky (E73)

P46 is composed of (forms part of)
Domain: E18 Physical Thing
Range: E18 Physical Thing
Superproperty of:
E19 Physical Object. P56 bears feature (is found on): E26 Physical Feature
Quantification: many to many (0,n:0,n)

Scope note:
This property allows instances of E18 Physical Thing to be analysed into component elements. Definition of the CIDOC Conceptual Reference Model 50
Component elements, since they are themselves instances
of E18 Physical Thing, may be further analysed into sub-components,
thereby creating a hierarchy of part decomposition. An instance of E18 Physical Thing may be shared between multiple wholes,
for example two buildings may share a common wall.
This property is intended to describe specific components that are individually documented, rather than general aspects. Overall descriptions of the structure of an instance of E18 Physical Thing are captured by the P3 has note
property. The instances of E57 Materials of which an item of E18 Physical Thing is composed should be documented using P45 consists of (is incorporated in).

Examples:
the Royal carriage (E22) forms part of the Royal train (E22)
the "Hog's Back" (E24) forms aprt of the "Fosseway" (E24)

Appendix B

Update Operations

Algorithm 4 AssociateActorToActivity (p:Actor, a:Activity)

1: **if** an explicit "P14 carried out by" link does not exist between a and p **then**
2: Add an explicit "P14 carried out by" link between a and p
3: **end if**

Algorithm 5 DisassociateActorFromActivity (p:Actor, a:Activity)

1: **if** an explicit "P14 carried out by" link exists between a and p **then**
2: Remove the requested "P14 carried out by" link between a and p
3: **end if**
4: **for** each superactivity:superAct of a related to p via the "P14 carried out by" link **do**
5: Remove possible explicit "P14 carried out by" link between superAct and p
6: **end for**

Algorithm 22 CreateInformationObject (io:InstanceName)

1: **if** an E73 Information object class instance with name io does not exist **then**
2: Create an instance of the class E73 Information Object with name io
3: **end if**

Algorithm 23 DeleteInformationObject (io:InformationObject)

1: **for** each physical man-made thing(carrier): ph related to io via the "P 128carries" link **do**
2: Remove the "carries" link between ph and io
3: **end for**
4: **for** each event:e related to io via the "was present at" link **do**
5: Remove possible explicit "was present at" link between io and e
6: **end for**
7: Remove the requested E73 Information Object class instance io

Algorithm 6 ContractActorFromActivity (p:Actor, a:Activity)

```
1: if an explicit "P14 carried out by" link exists between a or a superactivity of a
   and p then
2:     for each direct subactivity:subAct of a do
3:         Execute AssociateActorToActivity (p, subAct)
4:     end for
5: end if
6: if an explicit "P14 carried out by" link exists between a and p then
7:     Remove the requested "P14 carried out by" link between a and p
8: end if
9: for each maximal superactivity:supAct of a related to p via the "P14 carried
   out by" link do
10:    for each subactivity:subAct of supAct do
11:        if subAct is not superactivity or subactivity of a then
12:            Add subAct to collection: Col
13:        end if
14:    end for
15: end for
16: Execute DisassociateActorFromActivity (p, a)
17: for each maximal activity:act in Col do
18:    Execute AssociateActorToActivity (p, act)
19: end for
```

Algorithm 7 AssociateSubActivityToActivity (suba:Activity, a:Activity)

```
1: if an explicit "P9 forms part of" link does not exist between suba and a then
2:     Add an explicit "forms part of" link between suba and a
3: end if
```

Algorithm 8 DisassociateSubActivityFromActivity (suba:Activity, a:Activity)

```
1: if an explicit "P9 forms part of" link exists between suba and a then
2:     Remove the requested "forms part of" link between suba and a
3: end if
```

Algorithm 9 CreateActor (p:InstanceName)

```
1: if an E39 Actor class instance with name p does not exist then
2:     Create an instance of the class E39 Actor with name p
3: end if
```

Algorithm 10 DeleteActor (p:Actor)

```
1: for each activity:a related to p via the "P14 carried out by" link do
2:     Remove possible explicit "P14 carried out by" link between a and p
3: end for
4: Remove the requested E39 Actor class instance p
```

Algorithm 11 CreateActivity (a:InstanceName)

```
1: if an E7 Activity class instance with name a does not exist then
2:     Create an instance of the class E7 Activity with name a
3: end if
```

Algorithm 12 DeleteActivity (a:Activity)

```
1: for each superactivity:superAct of a do
2:     Execute DisassociateSubActivityFromActivity (a, superAct)
3: end for
4: for each performer:p related to a via the "P14 carried out by" link do
5:     Remove possible explicit "P14 carried out by" link between a and p
6: end for
7: for each man-made object:o related to a via the "P9 was used for" link do
8:     Remove possible explicit "P9 was used for" link between o and a
9: end for
10: Remove the requested E7 Activity instance a
```

Algorithm 13 AssociateActivityToMMObject (a:Activity, o:ManMadeObject)

```
1: if an explicit "P9 was used for" link does not exist between o and a then
2:     Add an explicit "P9 was used for" link between o and a
3: end if
```

Algorithm 14 DisassociateActivityFromMMObject (a:Activity, o:ManMadeObject)

```
1: if an explicit "P9 was used for" link exists between o and a then
2:     Remove the requested "P9 was used for" link between o and a
3: end if
4: for each superPart:superP of o related to a via the "P9 was used for" link do
5:     Remove possible explicit "P9 was used for" link between superP and a
6: end for
```

Algorithm 15 ContractActivityFromMMObject (a:Activity, o:ManMadeObject)

```
1: if an explicit "P9 was used for" link exists between o or a superpart of o and a
   then
2:     for each direct subPart of o do
3:         Execute AssociateActivityToMMObject (a, subPart)
4:     end for
5: end if
6: if an explicit "P9 was used for" link exists between o and a then
7:     Remove the requested "P9 was used for" link between o and a
8: end if
9: for each maximal superPart:superP of o related to a via the "P9 was used for"
   link do
10:    for each subPart:subP of superP do
11:        if subP is not superPart or subPart of o then
12:            Add subP to collection: Col
13:        end if
14:    end for
15: end for
16: Execute DisassociateActivityFromMMObject (a, o)
17: for each maximal part in Col do
18:    Execute AssociateActivityToMMObject (a, part)
19: end for
```

Algorithm 16 AssociatePMMThingToEvent (ph:PhysicalManMadeThing, e:Event)

```
1: if an explicit "was present at" link does not exist between ph and e then
2:     Add an explicit "was present at" link between ph and e
3: end if
```

Algorithm 17 DisassociatePMMThingFromEvent (ph:PhysicalManMadeThing, e:Event)

```
1: if a "P12 was present at" link exists between ph and e then
2:     Remove the requested "was present at" link between ph and e
3: end if
```

Algorithm 18 AssociateIObjectToEvent (io:InformationObject, e:Event)

```
1: if an explicit "was present at" link does not exist between io and e then
2:     Add an explicit "was present at" link between io and e
3: end if
```

Algorithm 19 DisassociateIObjectFromEvent (io:InformationObject, e:Event)

1: **if** an explicit "P12 was present at" link exists between io and e **then**
2: Remove the requested "was present at" link between io and e
3: **end if**
4: **for** each physical man-made thing(carrier): ph related to io via the "carries" link **do**
5: Execute DisassociatePMMThingFromEvent (ph, e)
6: **end for**

Algorithm 20 CreateEvent (e:InstanceName)

1: **if** an E5 Event class instance with name e does not exist **then**
2: Create an instance of the class E5 Event with name e
3: **end if**

Algorithm 21 DeleteEvent (e:Event)

1: **for** each information object:io related to e via the "P12 was present at" link **do**
2: Remove possible explicit "P12 was present at" link between io and e
3: **end for**
4: **for** each physical man-made thing(carrier):ph related to e via the "P12 was present at" **do** link
5: DisassociatePMMThingFromEvent (ph, e)
6: **end for**
7: Remove the requested E5 Event class instance e

Appendix C

Statistics of Real Data

Table C.1: Statistics of Real Data

	No inference	**Rdfs Inference**	**Rdfs+ru1e1 Inference**
Total triples	150.434	310.919	327.452
Number of classes	41	70	70
Number of subjects	35.244	35.244	35.244
Number of predicates	77	99	99
Number of activities	12.898	12.898	12.898
Number of Actors	0	86	86
P9 forms part of	8.937	8.937	24.444
P9 consists of	10	10	10
P14 carried out by	0	129	708
P14 performed	0	0	0
P128 carries	0	0	0
P128 carried out by	0	0	0
P16 used specific object	0	1.865	1.865
P16 was used for	0	49	49
P12 was present at	0	195	195
P12 occurred in the presence of	0	19.889	20.329
P46 is composed of	0	0	0
P46 forms part of	0	0	0

Appendix D

Statistics of Synthetic Data

Table D.1: Statistics of Synthetic Graphs

Total activities (depth 3)	**200k**	**400k**	**600k**	**800k**	**1m**
Depth 0	2.653525398	2.677488955	2.671271545	2.665987434	2.660000027
Depth 1	10.05812484	10.04726189	10.04509642	10.04423587	10.04809882
Depth 2	22.32649632	22.32199733	22.32375711	22.32513623	22.32635807
Depth 3	64.96185344	64.95325182	64.95987493	64.96464046	64.96554308

Table D.2: Statistics of Synthetic Graphs

Total activities (depth 4)	**200k**	**400k**	**600k**	**800k**	**1m**
Depth 0	3.605391218	3.595468349	3.605298433	3.612158485	3.609139326
Depth 1	13.31450105	13.31587165	13.31175228	13.30370418	13.31398334
Depth 2	8.149728095	8.150567028	8.145239484	8.145506502	8.149411209
Depth 3	20.00434591	20.00640516	20.0054854	20.00475303	20.00356808
Depth 4	54.92832105	54.93397537	54.93454218	54.93619636	54.92618527

Table D.3: Statistics of Synthetic Graphs

Total activities (depth 6)	**200k**	**400k**	**600k**	**800k**	**1m**
Depth 0	3.361814306	3.369403531	3.370718243	3.369096492	3.361393734
Depth 1	14.4688454	14.51887545	14.50093815	14.4815262	14.49460902
Depth 2	7.605460893	7.613378262	7.6112755	7.608238176	7.611102949
Depth 3	7.884972679	7.866725174	7.87826673	7.882154069	7.884158889
Depth 4	7.050273757	7.047625356	7.050041522	7.05185083	7.049968574
Depth 5	16.14427193	16.14217233	16.14265802	16.14337462	16.14343832
Depth 6	43.83183256	43.7818236	43.7889554	43.80763791	43.80062498

Appendix E

Virtuoso's parameter file

```
;
;  virtuoso.ini
;
;  Configuration file for the OpenLink Virtuoso VDBMS Server
;
;  To learn more about this product, or any other product in our
;  portfolio, please check out our web site at:
;
;      http://virtuoso.openlinksw.com/
;
;  or contact us at:
;
;      general.information@openlinksw.com
;
;  If you have any technical questions, please contact
; our support staff at:
;
;      technical.support@openlinksw.com
;


;
;  Database setup
;
[Database]
DatabaseFile =
/usr/local/virtuoso-opensource/var/lib/virtuoso/db/virtuoso.db
ErrorLogFile =
/usr/local/virtuoso-opensource/var/lib/virtuoso/db/virtuoso.log
LockFile =
/usr/local/virtuoso-opensource/var/lib/virtuoso/db/virtuoso.lck
TransactionFile =
/usr/local/virtuoso-opensource/var/lib/virtuoso/db/virtuoso.trx
xa_persistent_file =
```

```
/usr/local/virtuoso-opensource/var/lib/virtuoso/db/virtuoso.pxa
ErrorLogLevel = 7
FileExtend = 150
MaxCheckpointRemap = 1000
Striping = 0
TempStorage = TempDatabase


[TempDatabase]
DatabaseFile =
/usr/local/virtuoso-opensource/var/lib/virtuoso/db/
virtuoso-temp.db
TransactionFile =
/usr/local/virtuoso-opensource/var/lib/virtuoso/db/
virtuoso-temp.trx
MaxCheckpointRemap = 1000
Striping = 0


;
;  Server parameters
;
[Parameters]
ServerPort = 1111
LiteMode = 1
DisableUnixSocket = 1
DisableTcpSocket = 0
;SSLServerPort = 2111
;SSLCertificate = cert.pem
;SSLPrivateKey = pk.pem
;X509ClientVerify = 0
;X509ClientVerifyDepth = 0
;X509ClientVerifyCAFile = ca.pem
ServerThreads = 10
CheckpointInterval = 0
O_DIRECT = 0
CaseMode = 2
MaxStaticCursorRows = 5000
CheckpointAuditTrail = 0
AllowOSCalls = 0
SchedulerInterval = 10
DirsAllowed = ., /usr/local/virtuoso-opensource/share/virtuoso/
vad/home/strubul
ThreadCleanupInterval = 0
ThreadThreshold = 8
ResourcesCleanupInterval = 0
```

```
FreeTextBatchSize = 100000
SingleCPU = 0
VADInstallDir = /usr/local/virtuoso-opensource/share/
virtuoso/vad/
PrefixResultNames               = 0
RdfFreeTextRulesSize = 100
IndexTreeMaps = 256
MaxMemPoolSize                  = 10000000
PrefixResultNames               = 0
MacSpotlight                    = 0
IndexTreeMaps                   = 64

TransactionAfterImageLimit = 500000000

;;
;; When running with large data sets, one should configure the
;; Virtuoso process to use between 2/3 to 3/5 of free system
;; memory and to stripe storage on all available disks.
;;
;; Uncomment next two lines if there is 2 GB system memory free
;       NumberOfBuffers          = 170000
;       MaxDirtyBuffers          = 130000
;; Uncomment next two lines if there is 4 GB system memory free
;       NumberOfBuffers          = 340000
;       MaxDirtyBuffers          = 250000
;; Uncomment next two lines if there is 8 GB system memory free
;       NumberOfBuffers          = 680000
;       MaxDirtyBuffers          = 500000
;; Uncomment next two lines if there is 16 GB system memory free
;       NumberOfBuffers          = 1360000
;       MaxDirtyBuffers          = 1000000
;; Uncomment next two lines if there is 32 GB system memory free
;       NumberOfBuffers          = 2720000
;       MaxDirtyBuffers          = 2000000
;; Uncomment next two lines if there is 48 GB system memory free
;       NumberOfBuffers          = 4000000
;       MaxDirtyBuffers          = 3000000
;; Uncomment next two lines if there is 64 GB system memory free
;       NumberOfBuffers          = 5450000
;       MaxDirtyBuffers          = 4000000
;;
;; Note the default settings will take very little memory
;; but will not result in very good performance
;;
NumberOfBuffers          = 30000
MaxDirtyBuffers          = 20100
```

```
[HTTPServer]
ServerPort = 8890
ServerRoot =
/usr/local/virtuoso-opensource/var/lib/virtuoso/vsp
ServerThreads = 10
DavRoot = DAV
EnabledDavVSP = 0
HTTPProxyEnabled = 0
TempASPXDir = 0
DefaultMailServer = localhost:25
ServerThreads = 5
MaxKeepAlives = 10
KeepAliveTimeout = 10
MaxCachedProxyConnections = 10
ProxyConnectionCacheTimeout = 15
HTTPThreadSize = 28000
HttpPrintWarningsInOutput = 0
Charset = UTF-8
;HTTPLogFile         = logs/http.log

[AutoRepair]
BadParentLinks = 0

[Client]
SQL_PREFETCH_ROWS = 5
SQL_PREFETCH_BYTES = 12000
SQL_QUERY_TIMEOUT = 0
SQL_TXN_TIMEOUT = 0
;SQL_NO_CHAR_C_ESCAPE = 1
;SQL_UTF8_EXECS = 0
;SQL_NO_SYSTEM_TABLES = 0
;SQL_BINARY_TIMESTAMP = 1
;SQL_ENCRYPTION_ON_PASSWORD = -1

[VDB]
ArrayOptimization = 0
NumArrayParameters = 10
VDBDisconnectTimeout = 1000
KeepConnectionOnFixedThread = 0

[Replication]
ServerName = db-SHADOWFAX
ServerEnable = 1
QueueMax = 50000
```

```
;
;  Striping setup
;
;  These parameters have only effect when Striping is set to 1
;  in the [Database] section, in which case the DatabaseFile
;  parameter is ignored.
;
;  With striping, the database is spawned across multiple
;  segments where each segment can have multiple stripes.
;
;  Format of the lines below:
;    Segment<number> = <size>, <stripe file name>
;    [, <stripe file name> .. ]
;
;  <number> must be ordered from 1 up.
;
;  The <size> is the total size of the segment which is equally
;  divided across all stripes forming  the segment. Its
;  specification can be in gigabytes (g), megabytes (m),
;  kilobytes (k) or in database blocks (b, the default)
;
;  Note that the segment size must be a multiple of the
;  database page size which is currently 8k.
;  Also, the segment size must be divisible by the number of
;  stripe files forming the segment.
;
;  The example below creates a 200 meg database striped on two
;  segments with two stripes of 50 meg and one of 100 meg.
;
;  You can always add more segments to the configuration, but
;  once added, do not change the setup.
;
[Striping]
Segment1 = 100M, db-seg1-1.db, db-seg1-2.db
Segment2 = 100M, db-seg2-1.db
;...

;[TempStriping]
;Segment1 = 100M, db-seg1-1.db, db-seg1-2.db
;Segment2 = 100M, db-seg2-1.db
;...

;[Ucms]
;UcmPath = <path>
```

```
;Ucm1 = <file>
;Ucm2 = <file>
;...


[Zero Config]
ServerName = virtuoso (SHADOWFAX)
;ServerDSN = ZDSN
;SSLServerName =
;SSLServerDSN =


[Mono]
;MONO_TRACE = Off
;MONO_PATH = <path_here>
;MONO_ROOT = <path_here>
;MONO_CFG_DIR = <path_here>
;virtclr.dll =


[URIQA]
DynamicLocal = 0
DefaultHost = localhost:8890


[SPARQL]
;ExternalQuerySource = 1
;ExternalXsltSource  = 1
;DefaultGraph       = http://localhost:8890/dataspace
;ImmutableGraphs     = http://localhost:8890/dataspace
ResultSetMaxRows            = 900000000
MaxQueryCostEstimationTime  = 40000000 ; in seconds
MaxQueryExecutionTime       = 60000000 ; in seconds
DefaultQuery                = select distinct ?Concept where
[] a ?Concept LIMIT 100
DeferInferenceRulesInit     = 0  ; controls inference
;                                    rules loading
;PingService        = http://rpc.pingthesemanticweb.com/


[Plugins]
LoadPath = /usr/local/virtuoso-opensource/lib/virtuoso/hosting
Load1 = plain, wikiv
Load2 = plain, mediawiki
Load3 = plain, creolewiki
;Load4 = plain, im
```

```
;Load5 = plain, wbxml2
;Load6 = plain, hslookup
;Load7 = attach, libphp5.so
;Load8 = Hosting, hosting_php.so
;Load9 = Hosting,hosting_perl.so
;Load10 = Hosting,hosting_python.so
;Load11 = Hosting,hosting_ruby.so
;Load12 = msdtc,msdtc_sample
```