UNIVERSITY OF CRETE

COMPUTER SCIENCE DEPARTMENT

Design and Implementation of a tool for formulating recall-oriented structured queries on semantic networks

TZOMPANAKI AIKATERINI

MASTER OF SCIENCE THESIS

Heraklion, September 2012

Design and Implementation of a tool for formulating recall-oriented structured queries on semantic networks

Tzompanaki Aikaterini

Thesis submitted in partial fulfillment 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:

Assistant Prof. Yannis Tzitzikas

Researcher Martin Doerr

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

The work is partially supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS).

Design and Implementation of a tool for formulating recall-oriented structured queries on semantic networks

Tzompanaki Aikaterini

Master Thesis

Abstract

In the recent years there is a trend towards the creation of massive metadata repositories, usually based on the RDF/S technology, as in the domain of cultural heritage. ISO21127 (CIDOC Conceptual Reference Model) is a rich conceptual model (or *ontology*) capable of describing the world stored in such repositories. Simpler models like those consisting only of "core metadata" as in Dublin Core, lack the expressiveness and the potentiality to integrate the knowledge and to apply reasoning on it. Nevertheless, the more complex structure complicates the information searching: the declarative SPARQL query formulation becomes harder for the user due to the large number of ontology classes and properties, while on the other side keyword search does not take advantage of the information structure.

To address this problem, we suggest a new approach: we introduce a simpler model consisting of few fundamental classes and relationships aimed to be used for querying purposes only. Information search with this model is easier and more intuitive for the users, since its size and structure resemble those of the core metadata. Additionally, this model provides high recall rates because in the fundamental relationships we capture the total of potential paths over the CIDOC-CRM and also include property propagation through these paths. With the latter though, we introduce a statistical factor that may deteriorate precision since a property is not necessarily propagated along a path. Precision improvement can be achieved by creating specializations of the fundamental relationships or by adding more constraints on the queries.

To define the paths over the CIDOC-CRM schema that correspond to each fundamental relationship, we have created a "paths' language" which is designed to be easy to write and to be comprehended by non-expert users. Thereafter, we constructed

a tool that utilizes this language, permits the editing and validation of the fundamental relationships and their translation to SPARQL and provides extra supportive functions.

The proposed approach was proven adequate for expressing real research queries originating from independent (to this work) scientists in the domain of cultural heritage. The results of queries performed on repositories consisting of real metadata were encouraging, showing even 100% recall, when the repository's information was well-structured. Moreover we have shown that the usage of combined FRs in the query can improve the precision rate.

Supervisor: Yannis Tzizikas
Assistant Professor
Co-supervisor: Martin Doerr
Researcher

Σχεδίαση και υλοποίηση ενός εργαλείου για την διατύπωση δομικών επερωτήσεων επί σημασιολογικών δικτύων για πληροφοριακές ανάγκες υψηλής ανάκλησης

Τζομπανάκη Αικατερίνη

Μεταπτυχιακή Εργασία

Περίληψη

Τα τελευταία χρόνια δημιουργούνται όλο και μεγαλύτερα αποθετήρια μεταδεδομένων, συνήθως εκφρασμένων σε RDF/S, όπως για παράδειγμα συμβαίνει στο χώρο της πολιτισμικής κληρονομιάς. Το ISO21127 (CIDOC Conceptual Reference Model) είναι ένα πλούσιο εννοιολογικό μοντέλο (ή αλλιώς *οντολογία*) ικανό να περιγράψει τον κόσμο που φυλάσσεται σε τέτοια αποθετήρια. Απλούστερα μοντέλα, όπως εκείνα που αποτελούνται μόνο από «μεταδεδομένα πυρήνα» (core metadata), για παράδειγμα το Dublin Core, υστερούν ως προς την εκφραστικότητα και τις δυνατότητες που προσφέρουν για ολοκλήρωση πληροφοριών (integration) και για συλλογισμό (reasoning) επί αυτών. Ωστόσο, η πιο πολύπλοκη δόμηση δυσχεραίνει την αναζήτηση πληροφοριών: η δηλωτική διατύπωση επερωτήσεων SPARQL είναι πιο δύσκολη στο χρήστη λόγω του μεγάλου πλήθους κλάσεων και ιδιοτήτων της οντολογίας, ενώ η αναζήτηση μέσω λέξεων κλειδιών δεν αξιοποιεί την δόμηση της πληροφορίας.

Για να προσφέρουμε έναν εύκολο και αποτελεσματικό τρόπο αναζήτησης σε τέτοια αποθετήρια, προτείνουμε μια νέα προσέγγιση: εισάγουμε ένα απλούστερο σχήμα το οποίο αποτελείται από λίγες και «θεμελιώδεις» (fundamental) κλάσεις και σχέσεις και το οποίο χρησιμοποιείται *μόνο* για τις ανάγκες της αναζήτησης. Η αναζήτηση πληροφορίας μέσω αυτού του μοντέλου είναι ευκολότερη και πιο διαισθητική για τους χρήστες αφού το μέγεθος και η δομή του προσομοιάζει με αυτά των μεταδεδομένων πυρήνα που είναι οικεία στους χρήστες. Συνάμα η χρήση του προσφέρει μεγάλο βαθμό ανάκλησης αφού για κάθε θεμελιακή σχέση του σχήματος συμπεριλαμβάνουμε το σύνολο των πιθανών μονοπατιών επί του CIDOC-CRM και επίσης εκμεταλλευόμαστε τη διάδοση των ιδιοτήτων επί αυτών (property

propagation). Ωστόσο, ο στατιστικός χαρακτήρας που εισάγεται με την χρήση του property propagation (επειδή μια ιδιότητα δε διαδίδεται κατ' ανάγκη) μπορεί να χειροτερεύσει το βαθμό ακρίβειας. Περαιτέρω ακρίβεια στην ανάκτηση μπορεί να επιτευχθεί εξειδικεύοντας τις θεμελιώδεις σχέσεις, ή εκφράζοντας επιπλέον περιορισμούς στις επερωτήσεις.

Για να δημιουργήσουμε τις απεικονίσεις των θεμελιωδών σχέσεων στο CIDOC-CRM δημιουργήσαμε μια «γλώσσα μονοπατιών» που να είναι εύκολη στο γράψιμο και κατανοητή από μη ειδήμονες χρήστες. Εν συνεχεία κατασκευάσαμε ένα εργαλείο που αξιοποιεί τη γλώσσα αυτή και επιτρέπει τη συγγραφή και τον έλεγχο εγκυρότητας των θεμελιωδών σχέσεων, την μετάφρασή τους σε SPARQL και επιπρόσθετα προσφέρει πλήθος άλλων υποστηρικτικών λειτουργιών.

Η προτεινόμενη προσέγγιση αποδείχτηκε ότι είναι ικανή να εκφράσει αληθινά ερωτήματα ανεξάρτητων ερευνητών του τομέα της πολιτισμικής πληροφορίας. Επίσης, εκτελέσθηκαν ερωτήματα σε πραγματικά αποθετήρια μεταδεδομένων και τα αποτελέσματα ήταν ενθαρρυντικά επιδεικνύοντας έως και 100% ανάκληση, όπου η πληροφορία του αποθετηρίου ήταν σωστά δομημένη. Επιδεικνύεται επίσης η ικανότητα βελτίωσης της ακρίβειας, με τη χρήση συνδυαστικών ερωτημάτων και εξειδικεύσεων.

Επόπτης Καθηγητής: Γιάννης Τζίτζικας
Επίκουρος καθηγητής
Συν-επόπτης: Μάρτιν Ντοέρ
Ερευνητής

Acknowledgments

Firstly, I would like to thank my master's advisors, Yannis Tzitzikas and Martin Doerr. Their comments and advice as well as the guidance that they provided me with, were essential for the completion of this master thesis. I would also like to thank Mr. Doerr for giving me the chance to work with him as engineer and later as master student, in the Information Systems laboratory (ISL) of the Institute of Computer Science (ICS) in F.O.R.T.H. Our cooperation was seamless and I truly gained a lot of knowledge from him. Moreover, I would like to thank all the people from the ISL that offered me their precious experience and help during my master.

Then I would like to thank my friends Dimitra, Ioanna, Chara, Anna, Maria, Christos and Giorgos for being so supportive and encouraging during these two years of my master studies. Special thanks go to Christos Asaridis who partially contributed in the implementation of the 'Fundamental Relationships Configuration Tool'. Also, I would like to thank Stamatis who has always been there for me, supporting me in all possible ways.

Finally, I would like to thank my parents Athina and Yannis and my brother Spyros. I am grateful for all their love and support. Without them, I would not be the person I am today.

Στους γονείς μου

Contents

List of Figures i
List of Tables iii
Introduction 1
1.1. The General Problem 1
1.2. The Proposed Framework 3
1.3. Thesis Structure 4
CHAPTER 2 5
Background and Related Work 5
2.1. Background 5
2.2. Problem Statement 7
2.3. Related Work 9
CHAPTER 3 13
The Model Of Fundamental Categories And Fundamental Relationships 13
3.1. Fundamental Categories 13
3.2. Fundamental Relationships 15
CHAPTER 4 23
Technical Approach 23
4.1. Path Expressions Language 23
4.1.1. Syntax 28
4.1.2. Expressive Power - Semantics 30
4.2. Fundamental Relationships configuration tool 33
4.2.1. Description 33
4.2.2. Technical Details-Prerequisites 34
4.2.3. Functionality 37
4.2.3.1. Path Validation 38
4.2.3.2. Path to SPARQL Translation 44
4.2.3.3. Path to IVB-Template Translation 46
4.2.3.4. Sub-relationship Detector 47
4.2.3.5. Schema-Coverage Checker 57
4.2.3.6. Rules Detector 58
4.2.3.7. Multiple Instantiation and Disjoint Cases Handler 60
4.3. Fundamental Relationships Materialization on CIDOC-CRM 64

CHAPTER 5 67
Application and Experiments 67
5.1. 3D-COFORM Project 67
5.2. Research Space Project 72
5.3. Evaluation 73
5.3.1. Research questions 73
5.3.2. Test queries 81
CHAPTER 6 87
Conclusions and Further Work 87
CHAPTER 7 91
Bibliography 91
Appendix A 97

List of Figures

Figure 1: 'Thing refers to Thing' path 22
Figure 2: FCs-FRs folder hierarchy 36
Figure 3: Fundamental Relationships configuration tool 37
Figure 4: Defining the repository for the FR configuration tool 38
Figure 5: Example of a correct validation result 40
Figure 6: Example of a consistency error 41
Figure 7: Example of a syntactic error 42
Figure 8: Define FR name for saving 43
Figure 9: FR saved under displayed path 43
Figure 10: Example of a path to SPARQL translation 46
Figure 11: Example of a path to IVB-Template translation 47
Figure 12: Example of a successful sub-relation finding 49
Figure 13: Sub-paths of P 54
Figure 14: Sub-paths of F 55
Figure 15: Example of finding schema uncovered properties 58
Figure 16: Example of finding that a rule has been defined in the repository and can be changed in some paths. 60
Figure 17: Example of defining a multi-instantiation case 61
Figure 18: Example of finding a disjointness case 63
Figure 19: Free text search on QFI 68
Figure 20: FRs for category Thing 69
Figure 21: Results to the query 70
Figure 22: Browse to the first query result (Pechell statue) 71
Figure 23: Browse to Figure-1 (Pechell statue's component) 72
Figure 24: Query 1 graph 74
Figure 25: Query 2 graph 76
Figure 26: Query 3 graph 76
Figure 27: Query 4 graph 77
Figure 28: Query 5 graph 78
Figure 29: Query 6 graph 79
Figure 30: Query 7 graph 80
Figure 31: Query 8 graph 81

List of Tables

Table 1:Fundamental Categories and Fundamental Relationships. 19
Table 2: FRs for Query 3 77
Table 3: FRs for Query 6 80
Table 4: Queries and relevant objects 83
Table 5: Precision and Recall Rates per Query 84
Table 6: Combined query results 85

CHAPTER 1

Introduction

1.1. The General Problem

The growth of the daily information supply, prerequisites the existence of sufficient data storage means. This leads to the existence of different kinds of databases basically split into two categories, depending on the nature of information they store: structured or unstructured. Different information structure requires also different information access means. So when we are searching for the appropriate querying method we must as well think of the kind of database we are querying.

Unstructured database systems can successfully be accessed by text search engines that use keywords without prior knowledge of the structure of the content. Similar to them, web search engines rely on this method in order to return as many relevant results (i.e. documents) as possible, however in the cost of precision [1]. In other cases though not very frequent, this querying method may even result in low or no recall at all.

Structured databases are based on a schema description, which is used by structured querying languages for querying formulation, like SQL[1] for relational databases. In these databases, the semantics of the schema (e.g. table and column names) must be known to the user in order to create a query. Most of such systems, for example business data warehouses, are based on the "Closed World assumption" in which all applicable data of a domain and restricted context are supposed to be known. Only if all fields are filled in with all applicable data about the objects or subject of the information system will the query paradigm yield 100 percent recall and precision.

On the other hand, there are the semantic repositories (like digital libraries or cultural heritage semantic repositories) structured following a specific schema but also following the "Open World Assumption", accepting that knowledge is by nature incomplete. This assumption is also supported by the fact that metadata may be organized by different people or by using the schema in different ways, making the

[1] http://www.sql.org/

correlation among different sources or even the same source a great challenge. If we expand the notion of semantic repository to semantic network that may consist of several schemata, the complexity grows even more. Currently, the most popular search method in Open World systems is the keyword search and usually yields high recall rate but a medium to low precision rate. But yet again this requires the existence of the search term in the metadata in all correct fields.

The Semantic Web promises to overcome the recall and precision problem for information not backed up by high redundancy by resorting to rich, formally structured metadata for documents and objects of interest and links between them, and combining them with general, formal background knowledge. Large-scale metadata repositories, semantic networks of Resource Description Framework (RDF) [2] triples integrating large amounts of cultural–historical data have been developed that are globally accessible via the Internet, such as the Europeana[2], the cultureSampo[3], and many other projects world-wide. In these systems, the CIDOC Conceptual Reference Model (CRM) [3] is becoming more and more popular as a rich RDF schema adequate to integrate complex museum data, for instance submitted in the form of Lightweight Information Describing Objects (LIDO) Extensible Markup Language (XML) [4] records. Among those projects[4] are the German Digital Library[5], the ResearchSpace Project[6], the WISSKI[7] Project, and the CLAROS[8] Project. "Linked Open Data"[9] are advocated for cultural institutions, in which RDF data reside on local servers, but are accessible under published RDF schemata from the Internet.

The Semantic Web is an Open World system. Therefore users cannot know precisely what's out there, nor in which terms exactly things have been documented. Querying individually hundreds of different kinds of properties creates a huge recall gap compared to keyword search. Particular queries become so "precise" that real data rarely fit to the question, and querying a combination of even a few properties tends to frustrate the users with empty answers [5]. A global restriction of the semantic

[2] http://www.europeana.eu/portal/
[3] http://www.kulttuurisampo.fi/index.shtml
[4] http://www.cidoc-crm.org/uses_applications.html
[5] http://www.deutsche-digitale-bibliothek.de/
[6] www.researchspace.org
[7] http://wiss-ki.eu
[8] http://explore.clarosnet.org
[9] http://linkeddata.org/

network to “core metadata” on the other side, as propagated by defenders of Dublin Core [6] or VRA [7], deprives the systems of the reasoning capability and information integration that the Semantic Web promises and researchers need, and does not solve the inherent problem of incomplete knowledge. In addition to that, the lack of hierarchy among the classes and the lacto g the notion of ‘event’ don’t support associative querying. Associative querying is however essential for scientific information retrieval needs as described in chapters 3 and 5.

1.2. The Proposed Framework

In order to overcome the problem of effective searching in semantic networks as described above, we propose a querying system for semantic networks based on a few fundamental categories (FCs) and (binary) relationships (FRs). These categories are “base classes” covering the domain, and the relationships are deductions from complex path expressions of all sorts of deep relationships and documentation alternatives in a much richer and more specialized semantic network, rather than a reduction of the primary documentation to core metadata. The deductions are formulated to ensure the high recall of each FR by comprising the respective formulation variants of the facts documented directly or indirectly in the network. The FRs simulate a much simpler network of a small number of intuitive relationships, which can be combined in an advanced search to cover a wide range of frequent and relevant queries. However precision is no longer 100 percent, as people may have been used to in their local collection system. Many of the deductions only entail the probability that the relationship holds; e.g., we may conclude that a painting from a particular workshop was most probably created at the location of the workshop.

The implementation of this proposed framework in the context of the CIDOC-CRM schema has also been part of this thesis. In order to map each Fundamental Relationship in the CIDOC-CRM schema we need to create complex paths in the CIDOC-CRM that contain all possible interpretations of the Fundamental Relationship in CIDOC-CRM terms. To facilitate this process we have designed a “path’s language” that is easy for use especially for non-experts. Based on this language we have also created a tool that facilitates the creation, validation and translation of the paths to the SPARQL [8] querying language, providing also extra assisting functionality.

The proposed framework is applied in the framework of the European Integrating Project 3D-COFORM[10] , funded by the European Community's Seventh Framework Program (FP7/2007-2013, no 231809). In this project along with legacy metadata, metadata describing the digital provenance for empirical three-dimensional (3D) modeling processes are recorded. Digital provenance data [9][10] form deep chains of events connected by output–input, with up to tens of thousands of intermediary products that inherit many properties along the processing chains up to data about the digitized objects themselves. Using reasoning rules, we result in high recall rates, as not only explicitly documented properties but also derived properties across independently created metadata records can be combined for calculating the desired results. In parallel the Research Space[11] project is also implementing this approach.

The idea of the Fundamental Categories and Fundamental Relationships has already been presented to the public, in the "Museums and the Web 2012" conference [11] and the CIDOC Conference [12].

1.3. Thesis Structure

This thesis is structured as follows: in chapter 2 the background of the problem and the proposed solution is described, including a theoretical introduction to the used terms and a review of related work aiming at solving the problem; in chapter 3 our proposal for the Model of the Fundamental Categories and Fundamental Relationships is displayed; chapter 4 describes the newly invented technologies, the paths expressive language and the Fundamental Relationships tool, the implementation of the proposed ideas in the concept of the CIDOC-CRM and its practical implementation in the concepts of the projects 3D-COFORM Project and the Research Space Project; chapter 5 provides the validation and results and chapter 6 finally gives conclusions and future work. In Appendix A we provide the Fundamental Relationships materialization on the CIDOC-CRM schema.

[10] http://www.3d-coform.eu/
[11] http://www.researchspace.org/

CHAPTER 2

Background and Related Work

2.1. Background

In the context of this thesis we use and refer to many concepts from the Semantic Web field. In order for the reader to follow with the terminology and concepts used, in this section we display a brief description of the most frequently ones.

The Semantic Web is an initiative of the World Wide Web Consortium (W3C) towards to the building of a "Web of Data" in addition to the classic "Web of documents". The ultimate goal of the Web of data is to enable computers to do more useful work and to develop systems that can support trusted interactions over the network. The term "Semantic Web" refers to W3C's vision of the Web of linked data. Semantic Web technologies aid people to create data stores on the Web, build vocabularies, and write rules for handling data. Linked data are empowered by technologies such as RDF, SPARQL and OWL[12], which are described in the following.

The Resource Description Framework (RDF) is a framework for representing information in the Web using a simple schema. The design of the RDF has been motivated by a set of possible uses, like in web metadata description, metadata integration or automated processing of Web information by software agents, providing a world-wide lingua franca for these and other processes. The underlying structure of any expression in RDF is a collection of triples, each consisting of a subject, a predicate and an object, in the pattern: subject-predicate-object [2].

A semantic web querying language (SWQL) is defined as a technique developed to facilitate the data retrieval from the Semantic Web. Several categories of query languages can be distinguished, according to the format of the data they can retrieve: i) Query languages for XML, ii) Query languages for Topic Maps, iii) Query languages for RDF and iv) Query languages for OWL [13].

[12] http://www.w3.org/standards/semanticweb/

SPARQL is the most accepted querying language for RDF and is a W3C recommendation. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions [14].

Web Ontology Language (OWL) [15] is intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web. Moreover, OWL is used for reasoning purposes promoting effective querying and data integration. OWL 2 [16] has now succeeded OWL providing further syntactic features and new expressivity, but also remaining compatible with OWL.

According to Tom Gruber [17] an ontology is a specification of a conceptualization. In general, an ontology describes formally a domain of discourse. Typically, an ontology consists of a finite list of terms and the relationships between these terms. The terms denote important concepts (classes of objects) of the domain [1]. An ontology schema provides vocabulary and structure for describing the concepts and relationships of the ontology [18] .

Categorization can be considered as the process of separating things or concepts into categories. Rosh [19] proposes two general and basic principles for the formation of categories: The first has to do with the function of category systems and asserts that the task of category systems is to provide maximum information with the least cognitive effort. The second principle has to do with the structure of the information provided and asserts that the perceived world comes as structured information rather than as arbitrary or unpredictable attributes. Thus maximum information with least cognitive effort is achieved if categories map the perceived world structure as closely as possible. This condition can be achieved either by the mapping of categories to given attribute structures or by the definition or redefinition of attributes to render a given set of categories appropriately structured.

Information retrieval (IR) is the area of study concerned with searching for documents, for information within documents, and for metadata about documents, as

well as that of searching structured storage (relational databases) and the World Wide Web[13]. Even though the overlapping of the terms "information" and "data" retrieval is frequent, information retrieval more serves the needs of retrieving *information* about a subject than of retrieving data which satisfies a given query [20].

Basic performance and correctness measures for Information Retrieval Systems (IRS) are the precision and recall rate. The precision rate measures how many of the retrieved objects are relevant to the query. The recall rate measures how many of the relevant objects are included in the retrieved object set. IRSs aim at maximizing both the recall and precision rates, in order to best satisfy the user query needs.

2.2. Problem Statement

As described in the Chapter 1, in order to improve recall and precision rates in semantic web queries, a resort must be sought towards the direction of constructing rich metadata and using associative queries. If we take as granted that rich metadata structure schemata are used, what remains is to search for the reasons why queries and in particular associative queries, demonstrate poor performance in Open World systems. In particular in information aggregation services such as cultural metadata repositories the following reasons can be highlighted.

The first and most common case is that the relationship which the user is looking for is not documented, or is represented in a different way. For instance, someone may look for things "made from: steel", but some objects were registered as "has part," which are "made from: steel," or as "has type: steel object." As the modeling of ISO21127 evinces, it is impossible to normalize a global model for information integration to one unique representation for each property. Rather, in aggregation systems and the Semantic Web, one has to accept that properties are represented by sets of reasonable alternatives that can be related to each other by deductions.

Another factor is that most frequent associative queries include "AND" conditions (conjunctions). A typical query may ask for a particular type of thing from a particular period of time, a particular geographic area, used/made by a particular group of people. The problem is that the lack of recall in each field, due to missing

[13] http://en.wikipedia.org/wiki/Information_retrieval

alternative associations or incompleteness of knowledge, roughly multiplies with the number of fields put in the conjunction. If each element of the conjunction has a recall of 90 percent, the conjunction of four has about 66 percent ($=0,9^4$) if there is no other correlation between the fields. Therefore, a successful advanced search facility must strive to increase recall of each individual parameter. Even better would be to indicate to the user which parameter may have been most "catastrophic" to the result, so that he can alter it.

The more analytical and precise a global model is, the less obvious it is for the user how a simple, intuitive question relates to the ontology. Transitive properties (such as parts of parts or derivatives of derivatives) cause "propagation" of properties along those property paths. Propagation may be very complex to formulate as query, but is also very powerful when it comes to recall improvement. For instance, one could assume that the actors, place and time that are reported for the building of a house (the "super-event") also apply for or include the building of its walls (a "sub-event"); or that materials a part is made of are considered to be among the materials the whole is made of; or that the subjects a thing represents also apply to its copy or derivative, etc. Such reasoning allows for inferring facts that are not stated within a single metadata record. Take for example the following information taken from two different sites. On one hand, we have the British Museum[14] website saying that the "Horsemen from the west frieze of the Parthenon" is part of the Parthenon, and on the other hand there is the Acropolis Museum[15] stating that Parthenon was created by Pheidias. Using the CIDOC-CRM schema (indicated by the namespace "crm") the metadata describing these pieces of information would be:

- "Horsemen from the west frieze of the Parthenon" crm: forms part of "Parthenon"
- "Parthenon" crm: was produced by "Construction of Parthenon" crm: carried out by 'Pheidias'

Using reasoning on the integrated metadata we could assume that Pheidias was involved in the making of the Horsemen as well. In other words, in a query about the maker of the Horsemen, Pheidias would be deducted as a plausible answer. Thus poor queries that do not take into account such inferences, are more likely to fail to show high recall rates.

[14] http://www.britishmuseum.org/
[15] http://www.theacropolismuseum.gr/

Last but not least, another difficulty in creating adequate semantic queries is the usage of semantic querying languages like SPARQL. Most favored by information technology (IT) experts, SPARQL has transferred the old relational paradigm onto the graph structure of the Semantic Web, creating an incredibly complex system, even for specialists. Especially for the kind of SPARQL queries that are needed to represent the proposed framework, it is difficult to verify that it will yield the results intended by a domain expert simply be reading it.

2.3. Related Work

The problem of efficiently and conveniently querying semantic networks is one of immense importance. Recognizing this, several works have been realized and have been published in this field. In this section we display the most important ones, separated into three categories. The first category includes those proposals that aim at helping the user formulate their queries providing facilities exposed in the interface. The second category includes works that propose that the simplification begins from the foundations of the network, thus focus on using simple, flat metadata schemata. The third category refers to works realized towards the field of simplifying the query model, in a similar way that this work also does.

Systems that aim at helping users to formulate semantic queries on the interface are listed in the first category. This can be done with menu-guided user interfaces used to specify subject-property-object triples, combined with a look-ahead enhanced search, such as the one realized in DBpedia [21]. In [22] a work is proposed on query formulation based on both class and property browsing over plain or even fuzzy RDF/S. Nevertheless, formulating queries by transitions on the RDF schema would be successful in the case that the end user knows the semantics of the classes and the links of the schema. Moreover since a straight translation of the schema terms to natural language is not always valid, it would probably lead to misusage of the schema. In addition to these, queries constituting long chains of triples that also count a lot on property propagation through part/whole and derivation chains would be hard for the end user to formulate by browsing on querying time. However, this approach could be useful for formulating the paths responding to the Fundamental Relationships, a thought that is considered as future work. Other systems, like NiteLight [24], employ a query formulation facility with graph representations of the

ontology, but still require SPARQL knowledge. Another approach is the use of natural language queries, which are automatically mapped to associations of triples of the implemented ontology by a built-in dictionary and some inference mechanism, such as the Power Aqua system [25]. This approach relieves the user from knowing by heart the ontology terms, but it inherits all the well-known polysemy of natural language, which deteriorates precision, and often provides even worse recall than the explicit use of ontology terms. Other natural language search systems, such as Swoogle [24] and SemSearch [26] do not interface to a triple store.

The second category includes proposals that are in favor of reducing the complexity of the Semantic Network itself in order to reduce the complexity of querying it. The Dublin Core Metadata Elements [6], VRA Core [7] and other metadata standards reduce the network to flat relationships. The most widely accepted model, the Dublin Core, proposes a set of 15 properties: contributor, coverage, creator, date, description, format, identifier, language, publisher, relation, rights, source, subject, title and type. Similarly, the Consortium of Interchange of Museum Information[16] has proposed the metadata elements: who, what, when, where ("4W") as a domain of independent relations to four kinds of entities (person, thing, time, place), a kind of "faceted search" [27]. In other terms, whatever relation a thing may have to a person is an answer to the question "who," etc. This works relatively well for metadata describing only the history of objects. Otherwise, the ambiguity, for instance between history and subject, becomes overwhelming. Systems like Artifacts Canada (http://www.pro.rcip-chin.gc.ca/artefact/index-eng.jsp) provide an advanced search facility based on this paradigm (plus a "how"). However, so few metadata properties are too poor to allow for reasoning with integrated metadata [50] or query refinement. The lack of precision in the primary documentation, in particular the missing concept of events, results in the inability to integrate related data from different sources, as has been shown in [28]. Moreover the lack of reasoning capability makes the creation of these "simple" metadata a "not-simple" procedure. If the metadata are to be created individually for all elements of the complex correlation graphs characteristic for history, interesting works of arts and e-science data, the same facts may have to be repeated manually hundreds to tens of thousands of times, which is ineffective and error prone. Consider, for instance, the over twenty implementations

[16] http://old.cni.org/pub/CIMI/framework.html

of the "Thinker" by Rodin[17] and all related artifacts. All these systems cannot be scaled up to higher precision, because the precise knowledge is lost in the documentation in favor of recall.

An interesting intermediate between a full-fledged semantic network and faceted search (combining distinctive properties of the two aforementioned categories) is the very successful Finnish CultureSampo [29]. It uses nine instead of four facets, including material, events, and object types. For each facet it uses rich term hierarchies of inclusion or subsumption, and provides multiple explicit, direct relationships among facets, such as fifty kinds of social agent-agent relationships. It avoids the error-prone search for suitable properties to query, provides deductions from term hierarchies, class and property subsumption, selection of valid query parameters, and faceted search. It even provides a natural language search. Still misses, however, other deductions, such as property propagation along part-whole relations and derivation chains.

The third category summarizes works that could be used as alternative techniques in the place of the selected SPARQL querying language, in order to implement the abstract model that this work proposes. The Research Space[18] Project belongs to this category, using OWLIM[19] rules to interpret the corresponding to the Fundamental Relationships' paths that this work proposes. In this way, each Fundamental Relationship is mapped to a new single relationship that is materialized in the repository. Then the query of one Fundamental Relationship is reduced to the query of one single property. Another alternative, would be to use semantic views in order to represent each one of the Fundamental Relationship, like the vSPARQL [30] based on the SPARQL querying language or the RVL[31] based on the RQL [32] querying language, the latter of which is though yet not implemented. These alternatives resemble a lot the way we treat Fundamental Relationships in a view-like manner, but still require technical knowledge by the user constructing the views or the rules that represent the query. Instead, we provide with the "paths' language" a simple method for building the path that represents the query. Moreover, for the rules proposal where the rules are materialized on the provided metadata, the mechanism

[17] http://en.wikipedia.org/wiki/The_Thinker
[18] http://www.researchspace.org/
[19] http://www.ontotext.com/owlim

should also care for updates/modifications of the rules. Such a process would require extra time and effort for updating the respective relationships in the database and could become very frustrating when other rules are affected as well. This approach also entails the problem of overloading the database with extra-relationships, especially when we are referring to already massive databases. This problem does not apply for non-materialized views.

In this work, we propose a method that tries to combine the best from the aforementioned approaches and to go beyond them.

CHAPTER 3

The Model Of Fundamental Categories And Fundamental Relationships

The proposed model of the Fundamental Categories (FCs) and Relationships (FRs) is designed for the needs of the cultural heritage field and is based on the CIDOC CRM and extensions for the domain of digitization. Nevertheless, our approach can be applied to other discourses in an analogous way. Much of its reasoning capability depends on explicit representation of events, which is also the case in the ABC Harmony model [33], DOLCE [34], BFO [35] , the Europeana EDM [36], various digital provenance models and other ontologies. Our primary target is the generic search for things, ideas, people, and facts from the past which is characteristic for digital libraries, cultural–historical research, science, business intelligence and political inquiries. For the design of the model we draw on rich experience in the cultural domain such as from the Polemon Project [37] and explicit queries collected from archaeologists and museum curators and analyzed by us in the context of the 3D-COFORM Project.

3.1. Fundamental Categories

In a typical web search engine, a search would homogeneously return just web pages or, in a digital library, only documents. In a semantic network, however, users can retrieve any instance of any class known to the system by any kinds of direct or indirect relations to other entities in the network. Therefore, we first need to identify the main categories in which the entities of our universe of discourse can be split into. In this direction, we divide them into a set of relevant FCs that appear to be founded deeply in our intuitive understanding of the world in this or a similar form. These FCs serve as domains and ranges of the FRs (the properties of the model) described below. Similar to core metadata, we try to cover the domain with as few FRs as possible, which a user can easily learn, but still to be able to make some powerful distinctions keyword search cannot do, such as discerning places from people with the same name.

The idea is to try to satisfy as many different kinds of questions as possible by asking a few more general ones, and not only the most frequently asked questions. For the selection of the FCs, we follow the tradition of Ranganathan [38], CIMI's 4Ws [39], and others. Still, our method is mostly intuitive and by insight; the future may give us the chance for wider explicit user studies.

In our design, we have selected (the crm namespace refers to the CIDOC-CRM schema[20]:

1. **Thing** = crm:E70.Thing, comprising material and immaterial things, a special case of "what" and Ranganathan's "Matter."
2. **Actor** = crm:E39.Actor, comprising persons, organizations, offices, and informal groups, equal to "who" and Ranganathan's "Personality."
3. a.**Event** = crm:E2.Temporal_Entity, comprising states, historical and other periods in the sense of the CRM (crm:E4.Period), and events (crm:E5.Event) and activities (crm:E7.Activity) in the narrower sense. It is equal to Ranganathan's "Energy." In some cases, periods can be regarded as a "when."
   b.**Time** = crm:E52.Time-Span, comprising date-time intervals, a special case of "when" and equal to Ranganathan's "Time."
4. **Place** = crm:E53.Place, comprising geometric extents in space, on earth and on objects, often related to or even identified by some stable and prominent configuration of matter, such as a settlement. It is equal to "where" and Ranganathan's "Space."
5. **Concept** = crm:E55.Type, comprising all kinds of universals, such as types of things, people, events, places, species, etc. This is a special case of "what." Ranganathan and many library subject catalogues do not distinguish between particular things and types of things; however, FRBR [49] introduces the notion of "Concept."

These categories should cover the domain of interest as a "base level" distinction similar to Lakoff [40], but they are neither completely disjointed nor absolute. Disjointedness is actually not helpful for recall. For instance, a settlement can be at least a "Thing" and a "Place." A person (Actor) undergoing a surgery or a person as a body in an excavated tomb, may be described besides others, in terms of properties of a "Thing." Such implications may appear odd in other contexts. A

[20] http://www.cidoc-crm.org/rdfs/cidoc_crm_v5.0.2_english_label.rdfs

modern biologist may regard species as “Things”—i.e., human inventions with creators and other historical attributes—whereas other domains may see species only as “Concepts.” Therefore, the FCs should be adjustable/adjusted to the audience by adding or subtracting “less prototypical” subclasses [40], or even by extending them.

In the cultural–historical context, which we initially anticipated, queries with numerical values as parameters are rather rare (except for dates and geo-coordinates). However, in the 3D processing domain, such queries do occur. Therefore we will add in the future “crm:E54_Dimension” to the FCs, but we have not (yet) explored a generic treatment of different metrics in intuitive user queries.

3.2. Fundamental Relationships

The Fundamental Relationships (FRs) design is the most important part of the model design, as the FRs are the main building elements for queries. To create a precise query, a user must first “select” (in the sense of the SQL “Select” statement) the FC from which the question should return instances. In a normal digital library, this may be fixed to “document.” Then the user must compose a sort of “Where Clause.” The most simple one consists of a flat list of properties (i.e. FRs) with the selected FC as domain and with range values combined by AND. The design challenge is to find a minimal set of FRs intuitive to the user and easy to learn, which widely cover the respective discourse with high recall and a precision great enough not to be flooded by unrelated answers.

Pustejovsky [41] observed how language disambiguates words by the relations to other words in a phrase. For instance, “He spoke to the museum” versus, “He walked around in the museum” seems contradictory in an ontology, but does not surprise people in whatever language we translate it to. This “complementary polysemy,” as Pustejovsky calls it, can be explained by classifying contextual expressions into relatively few, language-neutral categories (“quales”). When a user selects a relationship term and a value, we use a similar mechanism to disambiguate the relationship term as a further help to the user: The term is interpreted according to the selected FC and the FC the range value is instance of, rather than forbidding “illegal” values.

A good example is the term *from*, a very natural relationship term describing any sort of origin or provenance. For instance, in good museum practice and intuition

"Things from New Guinea" may mean things found, produced, or used in New Guinea or things with parts from there. It may also mean things produced by people coming from New Guinea. This interpretation is common for all Place values. Museum metadata frequently contain the term "provenance" in this sense. However, "Things from J.W. Goethe" (an Actor) has a different interpretation: It could mean things created, produced, modified, said, acquired, owned, kept, or used by him or his household; gifts he gave or received; or awards he received. "Things from the Parthenon" (a Thing) may mean parts or pieces of the Parthenon, but it may also comprise inscriptions found on it. Quite differently, we would interpret "Actors (people) from New Guinea" as a sort of nationality concept, whereas "Actors (people) from Siemens Company" (Actor) would pertain to membership. "Places from Time" make no sense. All interpretations correspond to composite path expressions in the CIDOC CRM. Constrained to a particular combination of FCs as domain and range, it is feasible to find all relevant expressions in the ontology for this interpretation.

Our empirical sources for the FR are "simple" metadata schemata, such as the Dublin Core and the VRA models, but also the Europeana Data Model (EDM), experiences from structuring museum information [37], generalizations of the CRM itself, and intuition. We divide the relationships into those describing (1) how and what something is (classification, part-whole structure), (2) what an item has undergone in its history, and (3) what it may "show," say, or refer to. We have not looked at relationships of intention, motivation, or cause, because they are rarely documented. This may be the subject of future extensions. In our current implementation, we have selected:

1. ***has type***: denotes relations of an item to a classification, category, type, essential role or other unary property, such as a format, material, color. It generalizes over dc:type, dc:classification, dc:format, dc:language. The relationship is applicable to all FCs and has always range Concept.
2. ***is type of***: the inverse of "has type". The relationship is applicable to all FCs and has always domain Concept.
3. ***has part***: Denotes structural relations of an item to a narrower unit it contains. In the case of Actors, one would rather speak of "has member", and persons are the minimal elements. Domain and range must be identical. Part/whole relationships are handled as transitive properties.

4. ***is part of***: the inverse of "has part". Denotes structural relations of an item to a wider unit it is contained in. In the case of Actors, one would rather speak of "is member of" and again domain and range must be identical.
5. ***from, has generator***: denotes the relations of an item to constituents of a context in its history which is either significant for the item, or the item is significant for the context, "provenance" in the widest sense, including time intervals and places. In the case of genealogy or group formation, natural language prefers the terms parent and founder respectively in order to refer to Actors. The relationship is a special case of has met.
6. ***is origin of, generator of***: the inverse of "from", "has geneator". In case of Actor as domain, one would rather speak of "is owner" or "is creator of ".
7. ***is similar or the same with***: denotes the symmetric relation between items that share features or are possibly identical and so it is treated as a transitive property. It is only usual for Things to document similarity manually. There exist enough comparison algorithms that deduce degrees of similarity automatically. We do not deal with these in this work.
8. ***has met***: denotes the symmetric relation between items that were present in the same event, including time intervals and places. Applicable to any combination of FCs except for Concepts. Can be considered as the super-relationship of many FRs, such as the "from" or the "has part" and their inverse ones.
9. ***refers to or is about***: denotes the relation of an item that is information, contains information or has produced information to the item this information refers to or is about. The relation can even be extended to a Place from where such information originates.
10. ***is referred to by/ is referred to at***: the inverse of refers to.
11. ***borders or overlaps with***: this symmetric Relationship denotes the relationship between instances of the category Place or Time that limit with one another or overlap.
12. ***by***: denotes the active participation of an actor upon a Thing or Event

Table 1 describes which of the above relationships are applicable to respective combinations of FCs as domain and range. Together with the Fundamental

Relationships specializations have been declared for certain FRs (specializations are described latter in this section). In the table the reader can view which specializations each FRs contains. They are displayed under the respective FR and one level deeper.

Each relationship has a different interpretation for each applicable combination of domain and range, which adapts the general meaning described above to the concrete case. Counting the different interpretations of the FRs that depend on the FCs set as domain and range, we reach the number of ninety four different interpretations. Nevertheless, the user is displayed a maximum of six FRs per each FC combination.

Domain (select)	**Range(query parameter)**					
	Thing	**Actor**	**Place**	**Event**	**Time**	**Concept**
Thing	8.has met 9.refers to or is about 10.is referred to by 3.has part 7.is similar or same with 5. from 4.is part of	8.has met 5.from 9.refers to or is about 10.is referred to by 12.by Used by Created by Modified by Found or acquired by	9.refers to 10.is referred to at 5.from used at created at found or acquired at is created by person from is located at	9.refers to 10.is referred to by 5.from destroyed in created in modified in used in	9.refers to 5.from destroyed on created on modified on used on	1.has type
Actor	8.has met 6.is owner or creator of 9. refers to 10.is referred to by	4.is member of 3.has member 8. has met 5.has generator 6.is generator of 9.refers to 10.is referred to by	8.has met 5.from 9.refers to 10.is referred to at	9.refers to 10.is referred to by 5.from 8.has met was brought into existence at was taken out of existence at performed action at influenced	9.refers to 5.from 8.has met was brought into existence at was taken out of existence at performed action at influenced	1.has type

Place	8.has met 6.Is origin of 9.refers to or is about 10.is referred to by	8.has met 6.Is origin of 9.refers to or is about 10.is referred to by 8.has met	4.is part of 3.has part 11.borders or overlaps with	9.refers to 10.is referred to by 8.has met	8.has met	1.has type
Event	6.is origin of 10.is referred to by 9.refers to or is about 8.has met created destroyed modified used	12.by 10.is referred to by 9.refers to or is about 8.has met brought into existence took out of existence	9.refers to or is about 10. is referred to at 5.from	9.refers to or is about 10.is referred to by 3.has part 5.from	9.refers to or is about 5.from starts ends has duration	1.has type
Time	10.is referred to by 8.has met	10.is referred to by 8.has met brought into existence took out of existence	8.has met	10. is referred to at 4.is part of 3.has part	4.is part of 3.has part	1.has type
Concept	2.is type of	2.is type of	2.is type of	2.is type of	2.is type of	1.has type 2.is type of

Table 1: Fundamental Categories and Fundamental Relationships.

The category Concept plays a special role. Concepts can be subdivided into subtypes of the FCs themselves, such as "Thing-Concepts," "Place-Concepts," etc. The FR 1.has type has domain all FCs, but the range is restricted to subtypes of the domain, such as "Thing has type Thing-Concept," "Place has type Place-Concept", etc. Further, all relationships in table 1 can be extended into "categorical" questions. For instance, the relation "Things from Place" can be extended into "Things from type of Place" via a join with "Place has type Place-Concept." This is implemented as generic mechanism. Moreover, the relationships in Table 1 are not all disjoint. There

are some subsumption relations between them; for instance, has met is in many cases a generalization of ‘from’.

Our framework foresees open-ended specializations of each of the FRs to dynamically meet demands for increased precision, down to the source ontology level. For instance, “Thing was created at Place” is an obvious specialization of “Thing from Place.” The user will be able to browse from the basic FRs to their specializations. This mechanism is absolutely impossible in an implementation based on core metadata. The user interface will further allow the user to combine the FRs even to simple path expressions, as if they were properties of the network; for instance, “all Things from Events of type ‘Excavation’ AND at time ‘1890-1910’ AND at place ‘Crete’.” In the 1990s, a similar system of customizable predefined “where” clauses that are presented as simple properties and can be combined to other queries for the Greek Archive of Monuments [37] was implemented. That system is, however, based on relational technology and a knowledge base managing the query mediation system but is still in use, providing the value of this approach. All together, this approach allows for constructing magnitudes of associative queries more than any core metadata system, and yet preserve their simplicity of access to the user.

Each FR of the model is interpreted in paths of properties taken from the underlying CRM. Main task of this thesis has been to search throughout the model, find and combine matching properties to the meaning of the respective FR. To illustrate it with an example consider the FR “refers to”. This is a typical FR that can be used with a combination of FCs as domain and range classes. In our example consider the “Thing refers to Thing” that connects Things with their subject, the Things they refer to or depict or describe. The CIDOC-CRM has two basic properties to describe these: P67.refers to and P52F.depicts. These two properties used in flat relationships would return desired results but would lack in recall. Combination of properties is the recipe for providing better results. For this reason we also include in the path “Things that carry (P128F.carries) things that refer to Things”. To add a ‘taste’ of intelligence we use the value of property propagation through transitive properties. So, also copies of things (P130F.shows features of) that refer to things are included, as we accept that copies have a high possibility of sharing subjects. We also include part-whole relationships as we accept that if a part of a Thing1 refers to another Thing2, then also the Thing1 refers to Thing2 and also to every part of Thing2. Finally we take advantage of the derivation chains that propagate the subject of the

initial Thing to the objects created through the derivation chain. So, when a Thing1 that refers to a Thing2 is digitized and anew thing is created, the new thing refers both to Thing1 and Thing2 and of course to their parts as described previously. Then also the objects that are created using as source this thing, will also share the same subjects. All these are valid if we consider subject preserving procedures. In this way, we have managed to include not only the relative to the meaning of the FR CRM-properties but also to enrich the result set using valuable reasoning rules. Following the described procedure we have generated the path in Figure 1. The path is edited in the paths' language, which is created for the purposes of this proposal and is described formally in chapter 4.

The design of the Fundamental Relationships has however a probabilistic character. That means that each of the properties included in a Fundamental Relationship participate in the relationship with a probability. Even so, for the sake of recall we add to each of the FRs all the possible schema properties that seem to bear a probability of leading to relevant to the query results. For example, the occurrence of an event in a place only entails a probability that also the sub-events that occurred in the context of the event also occurred in the same place. In the design of the FRs such inferences are included as they are very likely to contribute to the successful recall of the query.

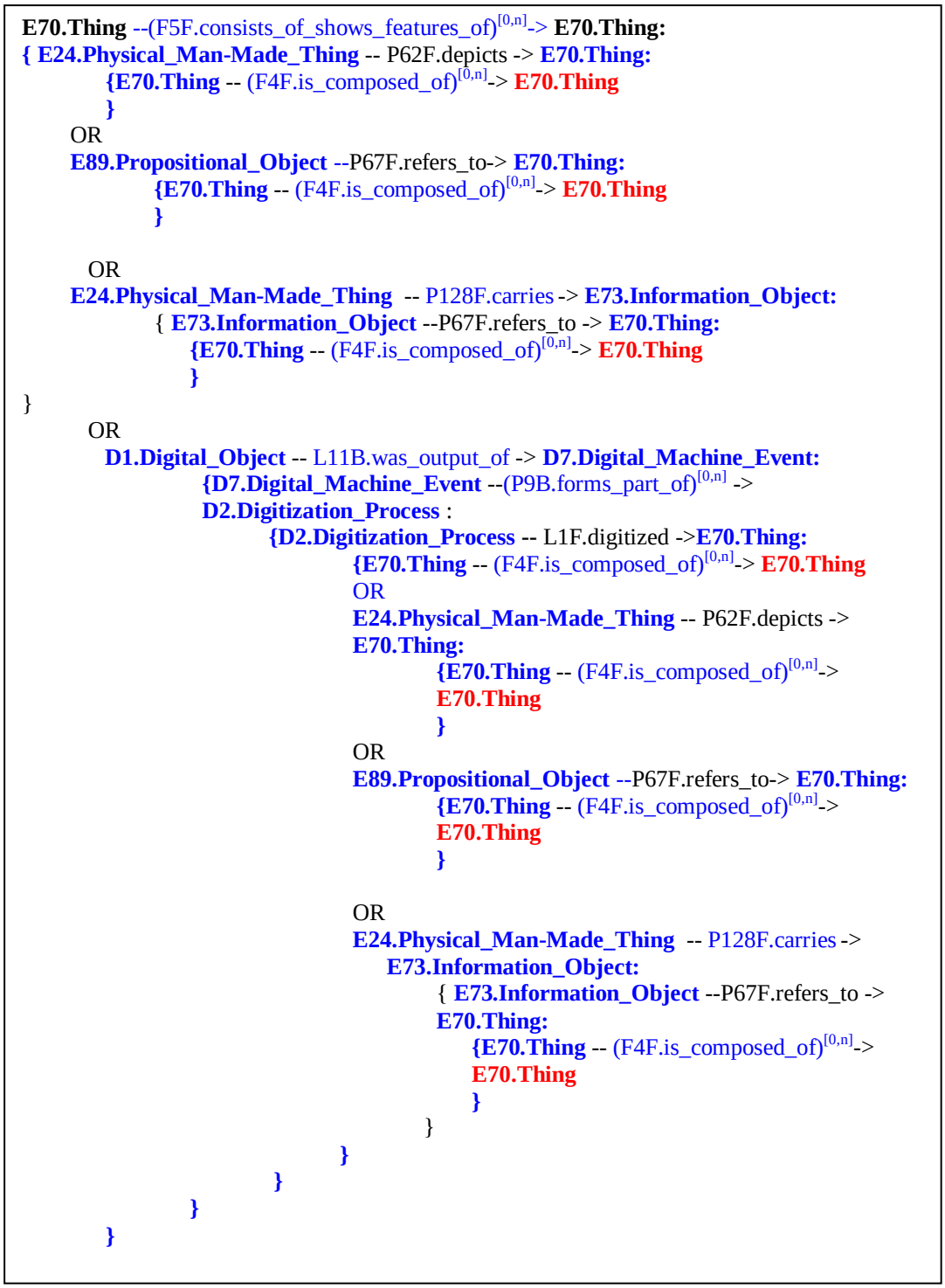

Figure 1: 'Thing refers to Thing' path

CHAPTER 4

Technical Approach

This chapter provides the technical details on the implementation of the proposed querying method. The implementation can be split into 3 phases:

a) design of an appropriate language for the formulation of the paths of the Fundamental Relationships that is rich enough to express the needs of the users but also simple enough to be used by non-experts like curators

b) design and implementation of a tool for aiding the user in the process of constructing correct paths, translating them to SPARQL and providing him a number of other facilities

c) implementation of the proposed model in the concept of the CIDOC-CRM schema.

The following sections describe each of these phases.

We should point out that this work addresses two kinds of users:

a) the administrative user, who is the user of the paths' language and the configuration tool and who uses the framework in order to create the materialization of the proposed model in concepts of the underlying schema - in our case the CIDOC-CRM schema and

b) the end-user, who is the one who performs a query on the metadata using the proposed framework and who might be totally unaware of the technologies described in this section. Still he must know the proposed simplified querying model. About this kind of user, please refer to section 5.1.

4.1. Path Expressions Language

The Fundamental Relationships (FRs) framework is designed to be handled by non-technicians. These users are people who are aware of the schema under which the metadata is built, but who might as well have poor technical expertise. The usage of already existing semantic query languages, like SPARQL, demands that the user knows the complex semantics of it. Moreover, writing in SPARQL is not intuitive, not

easy to follow and not easy to understand especially for non-computer scientists [42] [43].

For this reason a new simpler language to formulate the query statements that correspond to each FR had to be invented. Requirement for this language is that it is easily written and readable by the users. Since the users are aware of triples patterns, as they are aware of RDF ontologies, triple patterns sound like a perfect option. In these triple patterns, we distinguish the subject from the predicate using "--" and the predicate from the object using "->":

Triple:= subject--predicate->object

An example triple is the following where we have marked with bold the classes, in order to make it more readable. According to name conventions of CIDOC-CRM, a term starting from "E" denotes that we refer to a class of the schema, while a term starting from "P" denotes that we refer to a property of the schema.

E18.Physical_Thing--P46F.is_composed_of->**E18.Physical_Thing**

More complex triple patterns would also allow for reflexivity and transitivity in the predicate, indicated by [0,n]; 0 means that the predicate may not occur in the triple pattern, while n means that it may occur infinite times. Transitivity is essential for the exploitation of the property propagation, e.g. through part/whole or derivation chains:

Triple:= subject--(predicate)[0,n]->object

As an example consider the following triple, with which a thing is connected to all the parts that it consists of, but not only the ones directly stated as parts of it; we infer that parts of parts are also parts of the thing.

E18.Physical_Thing--(P46F.is_composed_of)[0,n]->

E18.Physical_Thing

This is translated to:

E18.Physical_Thing *(when predicate occurs 0 times)*

or

E18.Physical_Thing--P46F.is_composed_of->**E18.Physical_Thing**--P46F.is_composed_of->**E18.Physical_Thing**

… P46F.is_composed_of-> **E18.Physical_Thing** *(when predicate occurs n times)*

where "…" indicates the infinite repetition of the pattern "E18.Physical_Thing--P46F.is_composed_of->E18.Physical_Thing". In this way we interconnect a Thing

with all its subparts. Similarly, transitivity is used for copies of Things. If we have for instance that thing2 is a copy of thing1 and thing3 is a copy of thing2, then we infer through transitivity that thing3 is also a copy of thing1.

More complex triple patterns are supported that allow disjunction at the predicate level. In this way the user can define alternative predicates in the same triple pattern, instead of defining different triples, using one predicate per triple. In this way the paths become more compact and can be more easily read and updated. The pattern to be followed in this case is:

Triple:= subject--{predicate OR predicate …}->object

and including reflexivity and transitivity in the predicates as indicatively shown in the next pattern:

Triple:= subject--{predicate[0,n] OR predicate …}->object

As example consider the following triple:

E70.Thing -- {P92B.was_brought_into_existence_by OR P16B.was_used_for }-> **E7.Activity**

This triple indicates that we can move from the domain class 'E70.Thing' to the range class 'E7.Activity' following two ways (i.e. predicates): the "P92B.was_brought_into_existence_by" OR the "P16B.was_used_for". Instead of writing two different triples, the user is allowed to combine predicates that share the same domain and range. The rest of the cases (that combine predicates that share the same domain but not the same range class) are described latter in this section.

Patterns like the aforementioned can be combined to form paths. Paths are sequences of triples, in the simplest case like:

Path:=Triple:{Triple:{Triple:{Triple….}}}

The symbol ":" indicates that the current triple (the triple before the ":") is continued by the path that is enclosed in the next opening-closing brackets "{"and "}". The syntax allows for any triple following the triple pattern to be put in the path. Nevertheless, in order for the path to make sense and be semantically correct, the object of the triple before ":" must hold a sub-class relation (either way) with the subject of the next triple, after ":". For example, let's see the following path:

E24.Physical_Man-Made_Thing -- P128F.carries ->

E73.Information_Object:

{**E73.Information_Object** --P67F.refers_to -> **E70.Thing}**

With this path we aim at getting things that refer to other things. So, we have a path starting with the FC Thing and ending to the FC Thing. In the middle, using intermediate classes we can formulate a path that will lead to the correct result. So in the example, we "hop" to the information objects carried by the thing in domain and then to the things that these information objects refer to. To build this path, we actually connect the object of the first triple, with the subject of the second one, which can either be the same class (as in this example), or hold a sub class relationship with it (as in the next example).

In the next example, we demonstrate the case where the next subject is a super-class of the previous object. Here, a thing is created in an activity event (E7.Activity) and this activity may form part of wider events (E5.Event) which may or may not be activities. Although an E7.Activity is a E5.Event and the user could use E5.Event instead of E7.Activity, it is not wrong to write it in the way that this example shows. So, we permit also this case:

E28.Conceptual_Object--P94B.was_created_by -> **E7.Activity:**

{**E5.Event**--(P9B.forms_part_of)[0,n] -> **E5.Event }**

More complex paths would also allow for splits in the path. Our path expressions allow for branching at triple level, using the OR operator:

Path:=Triple:{Triple:{Triple:{Triple….}}

OR

Triple:{Triple:{Triple….}}

}

At this point we have referred to two different usages of the OR operator: among predicates and among triples. OR among predicates regards predicates that share the domain and range classes and can be as well translated as OR among triples. But this would be a redundancy, since the OR among predicates is the shortest way of two ways saying the same thing. This redundancy would not be in favor of convenience either, since when the user wants to alter something he has written, he would have to change more parts of the path. Path patters that include OR at triple level, aim at branching predicates that share the same domain class but not the same range class. This is also the reason why we can't use OR among predicates in this case.

In our model implementation on the CIDOC-CRM, branching plays a significant role. As we have said, each Fundamental Relationship is mapped to

different paths over the CIDOC-CRM schema. All these paths, that concatenated form the path of one FR, start from the same class and end to the same class, but may follow different routes using a different property sequence. The different paths that we follow are actually the different interpretations that we give to a Fundamental Relationship. For example a Thing from Place, may be translated as a Thing located in Place OR a Thing created in Place. We name *sub-paths* the different paths that we follow in order to create the total FR path. Sub-paths are mostly important when creating specializations for the FRs, thus when we want to check if a path can be defined as sub-relationship of a FR. Section 4.2.3.4 provides more details on this matter.

A practical example is displayed in the following path, where we have marked again with bold the classes. In this path, in order to find things that are *from* a place, we include those that are located in that place and those that were created at the place; but we also include the things that are part of things that are located or we created at that place. To achieve this, we can either use the direct relationship, P53F.has_former_or_current_location that leads to the place of the Thing, or we can do so by retrieving the Place of the creation event of the Thing. Since the P53F.has_former_or_current_location and the P94B.was_created_by don't share the same range class (E53.Place for the first, E5.Event for the second) we can't use OR in predicates. We have to create different 'triple branches' and for this reason we use the aforementioned syntax. Note also an issue that we mentioned before, about the linking of the successive triples. Here the subject E24.Physical_Man-Made_Thing of the second triple is a subclass of the first triple's object E70.Thing and thus it is correct to use it to continue the path. So, the path that the user writes is the following:

E70.Thing -- P46B.forms_part_of -> **E70.Thing:**
{E24.Physical_Man-Made_Thing --
P53F.has_former_or_current_location -> **E53.Place**
OR
E70.Thing -- P94B.was_created_by -> **E5.Event:**
{E5.Event -- P7F.took_place_at -> **E53.Place }**
}

This path can be split into two sub-paths, as shown below. This split is essential for breaking a path into the parts it consists of, in order to find the different interpretations that are given to some FR. Later, when the user wants to create a FR specialization, he will be able to check if indeed his path is a sub-path of of the FR. This is performed as

described later in Chapter 4, in the sub-relations detector of the 'FR configuration tool'. Yet again the split of an FR to its parts is the first step performed in quite all the function of the tool. The user does not build the separated sub-paths himself, as this would be unpractical for writing and updating reasons.

E70.Thing -- P46B.forms_part_of -> **E70.Thing:**

{E70.Thing -- P53F.has_former_or_current_location -> **E53.Place}**

and

E70.Thing -- P46B.forms_part_of -> **E70.Thing:**

{E70.Thing -- P94B.was_created_by -> **E5.Event:**

{E5.Event -- P7F.took_place_at -> **E53.Place }**

}

This introductory description was done in a simplified format of the paths' language we have created. To provide the formal description of the language, the next section provides the definition of the context-free grammar[21] it follows and which is described in Extended Backus–Naur Form (EBNF)[22] notation.

4.1.1. Syntax

This section provides the formal structure of the syntax followed in order to create the paths that are used in the generated language. For this reason, we define a context free grammar, containing the production rules that must be followed in order to create correct (valid) paths. The paths' language we use is a language based on this context free grammar.

DEFINITION 1:

We define an ontology schema (RDF schema) *O* as the finite set:

O = (C, P)

where C is the set of classes and P is the set of properties of the schema. Classes are the nodes in the *RDF graph* constituting the RDF schema and properties are the arcs

[21] http://en.wikipedia.org/wiki/Context-free_grammar
[22] http://en.wikipedia.org/wiki/Ebnf

connecting the nodes. An RDF graph is a set of RDF triples. An RDF triple has three parts:

1.subject: a node (class) of the RDF graph

2.predicate: an arc (property) of the RDF graph

3.object: a node (class) of the RDF graph

DEFINITION 2:

Let O = (O_1, O_2,…,O_n), $n \geq 1$ be the set of ontology schemata that we are covering, where $O_n = (C_n, P_n)$ as defined in Definition 1. The set of the ontology schemata O can then be re-written as the set consisting of the set of all classes and the set of all properties from the individual schemata:

O = (C, P), where $C = C_1 \cup C_2 \cup ... \cup C_n$ and $P = P_1 \cup P_2 \cup ... \cup P_n$.

We define a context free grammar G as the 4-tuple:

$G=(V, \Sigma, R, S)$ where

V={path, triple, predicate_expr, class, predicate, predicate_trans }, is the set of variables used in the grammar

Σ ={ ‘c’,’ p’, ‘:’, ‘--‘, ’->’, ‘{‘, ‘}’ , ‘(‘ , ‘)’ , ‘[0,n]’ , ‘OR’}, are the symbols used in the grammar where $c \in C$ and $p \in P$, $C, P \in O$.

R is a finite relation from V to V∪Σ ($V \xrightarrow{R} V \cup \Sigma$), represented by the rule set defined later in this section

S = $path \in V$, is the starting symbol

The set R consists of the following production rules:

path = triple (1)

path = triple, ‘:’,’{‘, path, ‘}’ (2)

path = triple, ‘:’, ‘{‘ ,path, ‘ OR ‘, path, ‘}’ (3)

triple = class,’--‘, predicate_expr,’->’, class (4)

predicate_expr = predicate | ‘{‘, predicate, ‘ OR ‘, predicate,’}’ (5)

predicate = ‘p’ | predicate_trans (6)

predicate_trans = ‘(‘,’p’,’)’,’[0,n]’ (7)

class = ’c’ (8)

DEFINITION 3:

We define as *paths' language* the context free language of the grammar G, as the set

$$L(G) = \{w \in \Sigma^* : S^* \Rightarrow w\}$$

where the symbol $\overset{*}{\Rightarrow}$ indicates the repetitive rule application, as explained by the next formula:

$$\forall\, u, v \in (V \cup \Sigma)^*, u \overset{*}{\Rightarrow} v \text{ if } u_1, u_2, \dots u_k \in (V \cup \Sigma)^*, k \geq 0$$

$$\text{such that } u \Rightarrow u_1 \Rightarrow u_2 \dots \Rightarrow u_k$$

This definition indicates that beginning from the start symbol we can end to a terminal symbol using repetitively the syntactic rules. Thus using these rules, the user can read or build paths using terms (classes and properties) of the schema and other terminal symbols defined in Σ. Appendix A can be advised for some practical examples of the usage of the language.

Let's see now an example of building a path using the rules defined for this grammar. Starting with the start symbol 'path' and using the 3rd rule:

path:=triple ':' '{' path 'OR' path'}' $\overset{(1)}{\Rightarrow}$

path:=triple ':' '{' triple 'OR' path'}' $\overset{(2),(4),(8)}{\Longrightarrow}$

path:='E70.Thing' '--' predicate_expr '->' 'E70.Thing' ':' '{' 'E24.Physical_Man-Made_Thing' '--' predicate_expr '->' E53.Place 'OR' path'}' '}' $\overset{(4),(5),(6),(7),(8)}{\Longrightarrow}$

path:='E70.Thing' '--' '(P46B.forms_part_of)[0,n]' '->' 'E70.Thing' ':' '{' 'E24.Physical_Man-Made_Thing' '--' 'P53F.has_former_or_current_location' '->' E53.Place 'OR' 'E70.Thing' '--' 'P94B.was_created_by' '->' 'E5.Event' ':' '{' 'E5.Event' '--' 'P7F.took_place_at' '->' 'E53.Place' '}' '}'

So we have built a path that finally consists only of terminal symbols by applying repetitively the aforementioned rules.

4.1.2. Expressive Power - Semantics

By the expressive power of the paths' language, we mean the set of all paths expressible in that language, adapting the respective definition for querying language as in [51]. To define the expressive power of our language, we display in this section the available operators and other existing features that enhance the expressivity of the

language. In addition to what the language can do, we also state what the language (still) can't do, thus the existing limitations.

In our language we provide explicit operators for union ('OR') among paths and among predicates and join (':') between successive triples. Also we provide the possibility to define reflexive and transitive properties ('[0,n]'). One by one these are examined in the following.

- Union among predicates

Union among predicates is a useful operation that enables the user to construct compact paths, as he is able to add more than one predicate in a triple, instead of building different triples.

Consider two predicates: $P_1, P_2 \in P\_expr$, where $P\_expr = P \cup P_r$ and P_r is the set of reflexive and transitive properties defined in the format provided by the grammar rules: (P)[0,n]. The union operation among the two predicates is:

$$P_1 \text{UNION} P_2 = \{P_1 \text{ OR } P_2\}.$$

Semantics: Let D_1=dom(P_1), D_2=dom(P_2), R_1=ran(P_1) and R_2=ran(P_2) where dom(P) represents the domain class of P and ran(P) represents the range class of P. Let also sub(D_1) be the sub-class set of D_1, sub(D_2) be the sub-class set of D_2, sub(R_1) be the sub-class set of R_1 and sub(R_2) be the sub-class set of R_2. We remind the reader that $D \in \text{sub}(D)$. Then in order for the usage of OR among predicates to be valid, we demand that

- $D_1 \in \text{sub}(D_2) \,||\, D_2 \in \text{sub}(D_1)$
- $R_1 \in \text{sub}(R_2) \,||\, R_2 \in \text{sub}(R_1)$

These constraints are essential for the validity of the triple, since the subject and object will be the same for all predicates in the pattern.

- Union among paths

This operation enables the union of two or more paths. Let two paths: $path_1$, $path_2 \in V \cup \Sigma$. Then:

$$path_1 \text{UNION} path_2 = \{path_1 \text{ OR } path_2\}$$

Semantics: $path_1$ and $path_2$ may be paths consisting of more paths. Let $object_1$ be the object of the last triple of $path_1$ and $object_2$ of the $path_2$ respectively. As before let sub($object_1$) be the sub-class set of $object_1$ and sub($object_2$) be the sub-class set of $object_2$. The following constraint must occur in order to have a valid usage of the OR operator among the two paths:

- $object_1 \in sub(object_2) \,||\, object_2 \in sub(object_1)$

This constraint demands that the two paths end to the same "category", so that either path's last object is super or sub class of the other.

- Join among triples

Join among triples is used in order to create sequences of them forming a path. Let two triples: $t_1, t_2 \in V \cup \Sigma$. Then, the join of the two triples is:

t_1 JOIN $t_2 = t_1:\{t_2\}$

Semantics: Let $object_1$ be the object of the triple t_1 and $subject_2$ the subject of the triple t_2. As before let $sub(object_1)$ be the sub-class set of $object_1$ and $sub(subject_2)$ be the sub-class set of $subject_2$ In order to maintain the continuity of the created path, we demand that:

- $object_1 \in sub(subject_2) \,||\, subject_2 \in sub(object_1)$

With this constraint we ensure that $triple_2$ is a natural "continuation" of $triple_1$.

- Reflexive and transitive properties

The expressivity of the language is enriched by including except for simple predicates also transitive and reflexive ones. Let a property $p \in P$. The reflexive and transitive form of it is: (p)[0,n].

Semantics: Let t=(subject, (p)[0,n], object) a triple containing the reflexive and transitive property p. In order for a property p to be defined as transitive, the domain and range of it must be the same class. So

- dom(p)=ran(p)

The transitivity of the property can not be restrained to a desired level; it goes as deep as defined in the metadata and that is the meaning on n in the pattern "[0,n]". 0 represents the reflexivity, thus the subject of t coincides with the object of t. So triple t is actually translated to

(1) subject

(2) (subject, p, object)

Take for example the following path containing a reflexive and transitive triple:

Path=a -- (p)[0,n] -> b:{c -- l -> d}

This is translated to

Path= {c -- l -> d OR a -- p -> b:{c -- l -> d}}

After describing the expressive potential of our language we can identify that certain operations have not yet been defined.

- Negation

Negation (NOT) is not defined in our language yet. There is not the possibility to describe

- a negative triple: NOT {a -- p -> b}
- a negative path: NOT {{a -- p -> b:{c -- e -> f}}}
- a negative class: NOT{a}

- Conjunction among paths

Conjunction in our language is defined for connecting sequential triples using the symbol ":". The semantics behind the usage of the operator have been described in the previous lines. It has not been defined though for connecting paths, in the way that "UNION among paths" is defined. Such a case is for example (using the AND operator):

```
E70.Thing -- P46B.forms_part_of -> E70.Thing: {
        E70.Thing-- P53F.has_former_or_current_location -> E53.Place
        AND
        E70.Thing -- P94B.was_created_by -> E65.Creation: {
                E65.Creation--P7F.took_place_at -> E53.Place
                }
        }

}
```

4.2. Fundamental Relationships configuration tool

4.2.1. Description

The process of building the paths for the Fundamental Relationships is complex, thus error prone, since it requires that the (administrative) user searches throughout the complex schema in order to find and combine the appropriate properties he needs for the FR interpretation. This poses the possibility of using the schema properties in a wrong way, or constructing chains of triples that do not

constitute continuous paths. Moreover, the user might use the grammar of the paths' language in a wrong way creating syntactical errors. Manually debugging the Fundamental Relationships paths would enforce extra difficulty and would diminish the usefulness of the framework.

Except for checking the correctness of the path, there are a number of collateral actions that can be performed on the paths and which manually would be time-consuming and frustrating. Checking the coverage of the schema; finding if there is a sub-relationship connection of the path with other Fundamental Relationships; checking of which rules on the database could be replaced in which Fundamental Relationships, transformation of the path to SPARQL query statement; these all are actions that are expected to be more or less frequently performed on the Fundamental Relationships and which can cost time and effort for the user if performed manually.

To overcome these limitations, we have designed and implemented the 'Fundamental Relationship configuration' tool. This tool provides the user with the possibility to perform automatically or semi-automatically all the aforementioned actions, saving not only time and effort but also ensuring the soundness of the result.

In the next sub-sections we describe the functionality of the tool and provide some technical details when necessary.

4.2.2. Technical Details-Prerequisites

Before continuing with the technical details, we remind again that this tool is to be used by the administrative user, defined in the beginning of this chapter. This user is supposed to know the schema under which the metadata are built.

The Fundamental Relationships configuration tool is a java based component, running with java 7.

It connects with a user-defined sesame repository[23], which is defined by the user who must provide the repository server and the repository name as Figure 4 shows. This repository may be the repository containing the metadata that are to be queried or an empty from metadata repository. In both cases, the ontology schema(s) that the paths are derived from must have been loaded in the sesame repository. Of course the user must make sure that the repository is up and running.

[23] http://www.openrdf.org/about.jsp

As stated before, we make use of inferences (rules) in order to make the querying process more efficient. These rules are specified in an OWLIM[24] reasoner embedded on the sesame repository and are deductions creating new implicit relationships among instances of the repository. An example of a rule that we have defined is the following (written in the OWLIM proposed format; L declares that we are using a property of the CIDOC-CRM digital schema):

b <crmdig:L21F.used_as_derivation_source> c

b <crmdig:L22F.created_derivative> d

---

d <crm:F1F.is_derivative_of> c

This rule means that if an event (= the domain class of L21F property) b has as derivation source the object (= the range class of L21F) c and also has created a derivative object (= the range class of L22F) d, it is deducted that the object d is a derivative of the object c. For this reason, a new implicit relationship will be created among the instances d and c, making the information of the metadata repository richer. Moreover, rules are used in order to shrink the paths contained in a query and thus in order to make the query statement shorter. This is useful especially for patterns that appear frequently in the FRs. Long paths, thus long SPARQL statements are likely to cause memory exceptions during query execution, so a solution is to create rules in the repository for these paths. On the other hand, using rules extensively would dynamically cause over-loading of the repository, especially when we are talking about big metadata repositories due to the creation and addition of new metadata. Moreover the updating of the rules is a quite difficult process, so rules should be used taking into consideration both the advantages and the disadvantages.

As mentioned above, we use the custom made rules in order to create shorter paths, thus shorter queries. For this reason, the tool needs access to the custom OWLIM rules that are defined in the repository where the metadata are kept and the user must provide it, identifying in the configuration file, the location of the respective .pie file (i.e. the file where the rules are defined in the repository).

The Fundamental Relationships paths are stored as string files that are organized in folders, as shown in the folder hierarchy of Figure 2. On the first level, the organization is by Fundamental Category. On second level the FRs are organized

[24] http://www.ontotext.com/owlim

in “Fundamental Category - Fundamental Category” folders. Each folder in this level stores the defined FRs for the two categories (in domain and range). For example in the folder ACTOR-THING we store the FRs that have as domain the FC ‘ACTOR’ and as range the FC ‘THING’. It is essential for the functionality of the tool, to maintain the organization as specified.

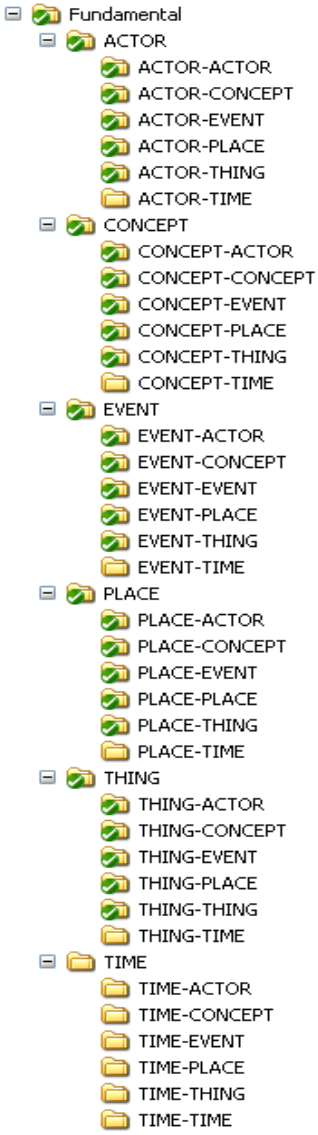

Figure 2: FCs-FRs folder hierarchy

A screen shot of the interface to this tool can be seen in Figure 3. Basically the input to this tool is put in the *Query Path* field (if not loaded from a file). The input is a path, written following the proposed grammar rules. Depending on the selected action to be performed on the input path, the respective result is then shown in the *Results* field. Other actions that do not necessarily require an input path are hidden in

the links *File*, *Special Cases* and *General Actions*. In the next sub-sections we describe the most profound functions of this tool.

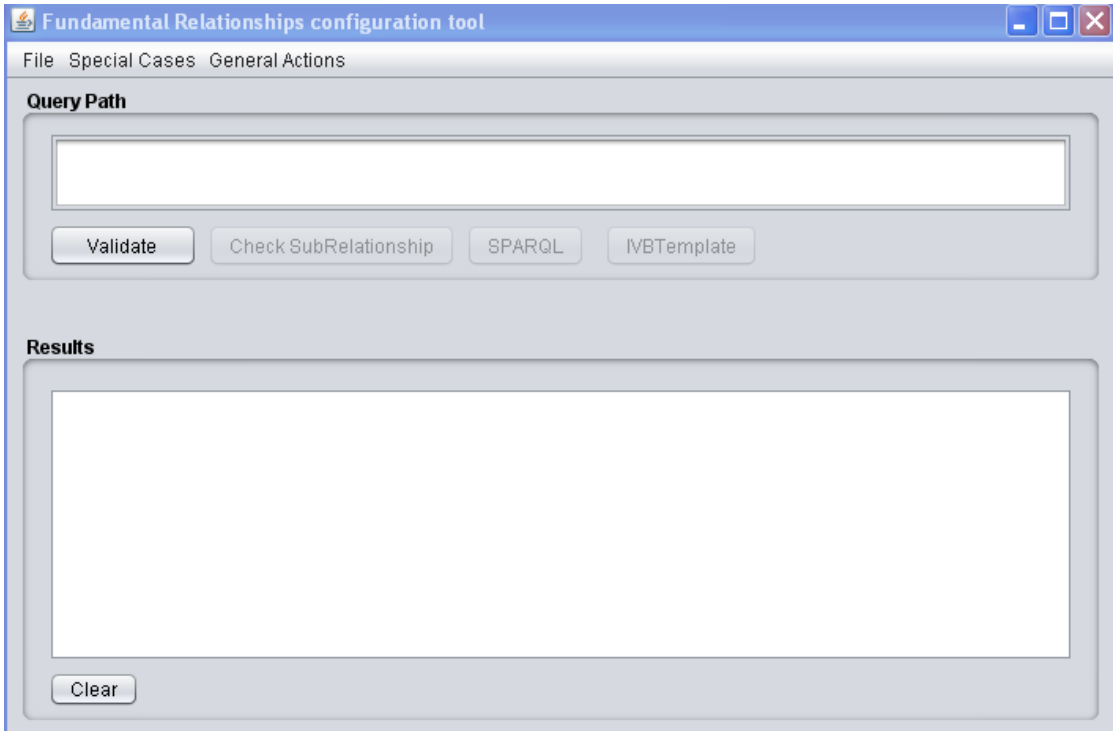

Figure 3: Fundamental Relationships configuration tool

4.2.3. Functionality

This section describes the provided functionality of the Fundamental Relationships configuration tool. It displays the available functions, lists the custom prerequisites for each one and describes briefly the idea behind the solution. In order to make the description more elaborate, we will use a running example throughout the functionality description.

This running example begins with a user that has written a path following the proposed grammar. This is the path corresponding to the specialization "Thing is located in Place" of the FR "Thing from Place". In order to create it, the user included the transitive property F4B.is_component_of. This is a property resulting from the application of a rule that unites all the Thing part-whole CIDOC-CRM properties into one, for easier reference. Then, to find the place where the thing is located, the user includes the P53F.has_former_or_current_location and the P54F.has_current_permanent_location which are properties that lead from a Thing to a Place. To include even more instances in the result set, the user also includes part-whole transitivity for the Place in range. In other words, to find what things are placed

in a specific location, we retrieve all Things and the parts of the Things that have current or former location or current permanent location the specified Place, or parts of this Place. This path is displayed below:

E70.Thing -- (F4B.is_component_of)[0,n] -> **E70.Thing:**

{E18.Physical_Thing -- {P53F.has_former_or_current_location OR P54F.has_current_permanent_location}->**E53.Place :**

{E53.Place --(P89F.falls_within)[0,n] ->**E53.Place**

}

}

First step when launching the tool, is to define the sesame repository (server location, repository name), as shown in Figure 4. The defined repository must be up and running in order for the tool to connect. In any other case, an error will appear and the tool will not be usable.

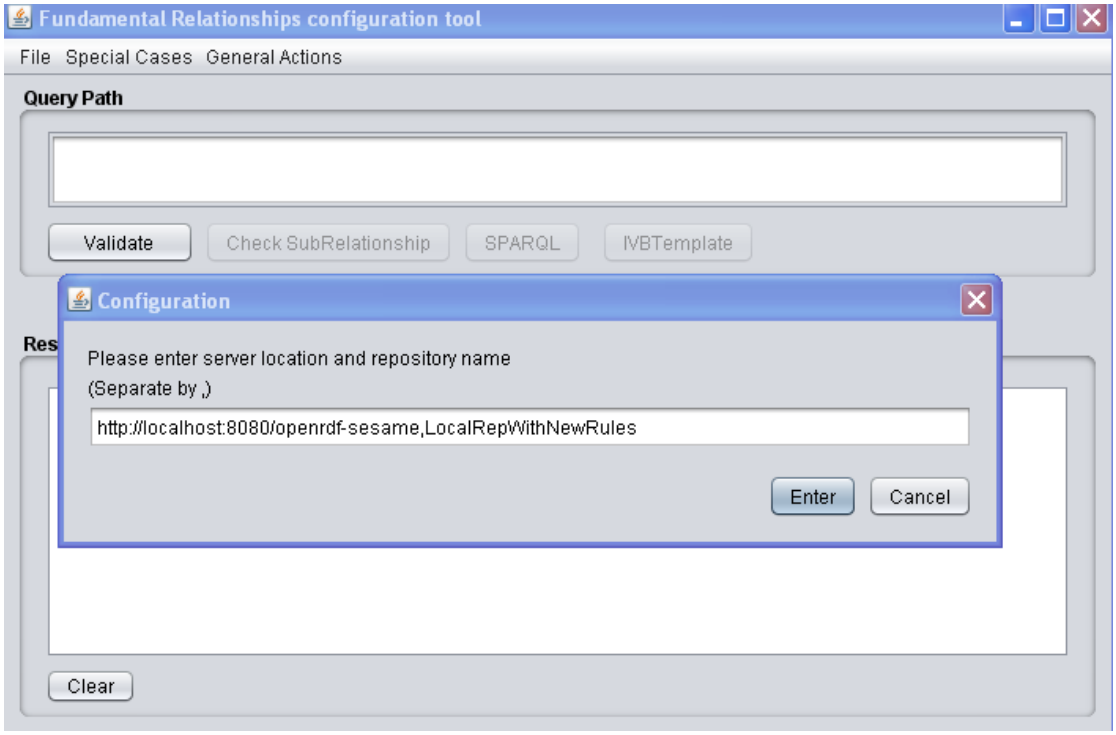

Figure 4: Defining the repository for the FR configuration tool

4.2.3.1. Path Validation

Now the user has connected to the tool and is ready to use it. First thing he shall want to do is to ensure that his path is correct both syntactically and semantically according to the schema. To do so, the user either writes the path directly to the Query

Path field, or loads it from a file, using the option *File→Load File* and then presses the *Validate* button.

Given the path string, the program will firstly check for syntactic errors, thus for misusage of the proposed grammar e.g. forgetting to close some parenthesis or bracket. When the syntax is correct, it will check for consistency with the schema e.g. using wrong range class for a predicate. If errors are found the validation is false, and the respective error message appears at the *Results* field. If the validation is correct, the path will appear at the *Results* screen written in an intended list format, in order to be more readable. In the next figures, we can see the results:

a) from a successful validation when the user provides the correct path given above (Figure 5)

b) from a consistency error (Figure 6) when the user provides the following path, using E53.Place instead of E70.Thing for range variable class for the predicate: F4B.is_component_of, while it expected E70.Thing:

E70.Thing -- (F4B.is_component_of)[0,n] ->**E53.Place:**
{E18.Physical_Thing -- {P53F.has_former_or_current_location OR P54F.has_current_permanent_location}->**E53.Place :**
{E53.Place --(P89F.falls_within)[0,n]->**E53.Place**
} **}**

c) from a syntactic error (Figure 7) when the user provides the following path forgetting to separate the subject from the predicate with -- as required by the grammar rules:

E70.Thing (F4B.is_component_of)[0,n] ->**E70.Thing:**
{E18.Physical_Thing -- {P53F.has_former_or_current_location OR P54F.has_current_permanent_location}->**E53.Place :**
{E53.Place --(P89F.falls_within)[0,n] ->**E53.Place**
} **}**

Note, that when a consistency error occurs, like in case b, about providing invalid classes for domain or range of a predicate, it might as well be a "valid" error. This happens when we have multiple instantiation cases. As multiple instantiation we mean the case when an instance can belong to more than one different classes of the schema, eg an instance of the class E70.Thing can also be documented as an instance of the class E53.Place. So, the user is given the opportunity to "legalize" the error, by

indicating to the system about the multiple instantiation case that they want to enable. Generally, CIDOC-CRM allows for multiple instantiation. On the other hand, if the error corresponds to usage of a disjoint class, the user is advised to remove this case from the respective configuration file, in order to be able to use it in the preferred way. A class is disjoint with another class, when instances of the one can't be defined also as instances of the other. More about multiple instantiation and disjoint cases will be discussed in section 4.2.3.7.

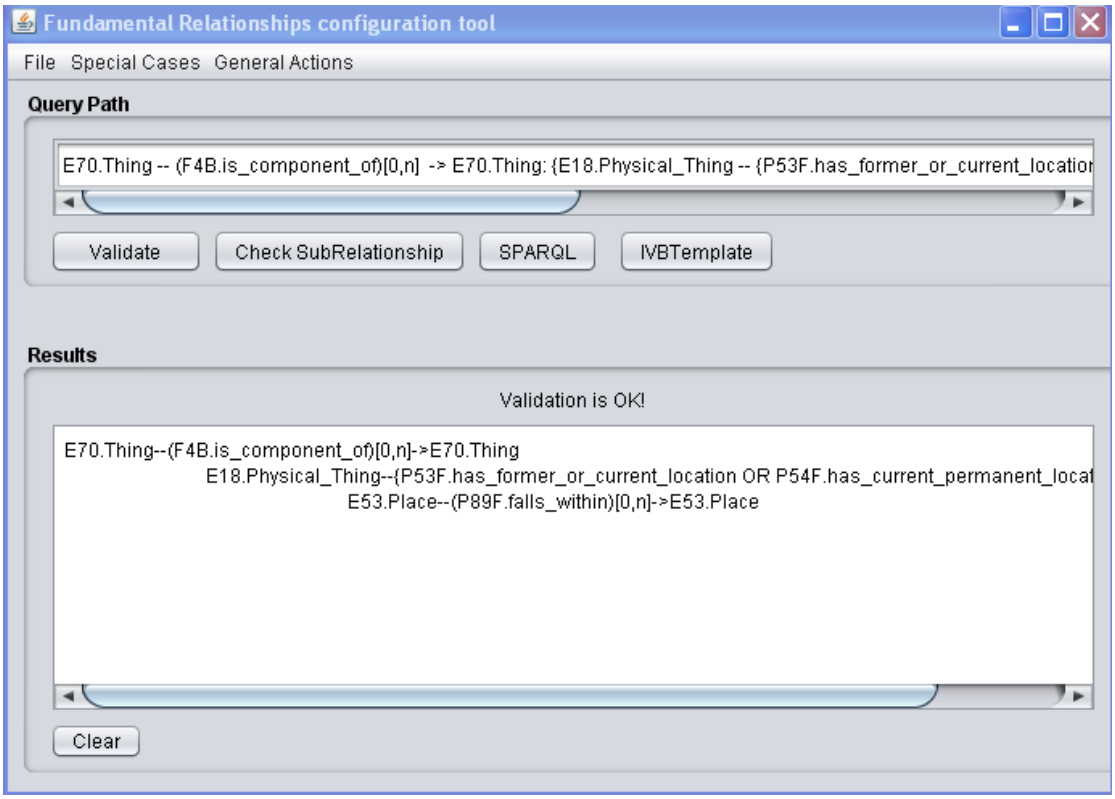

Figure 5: Example of a correct validation result

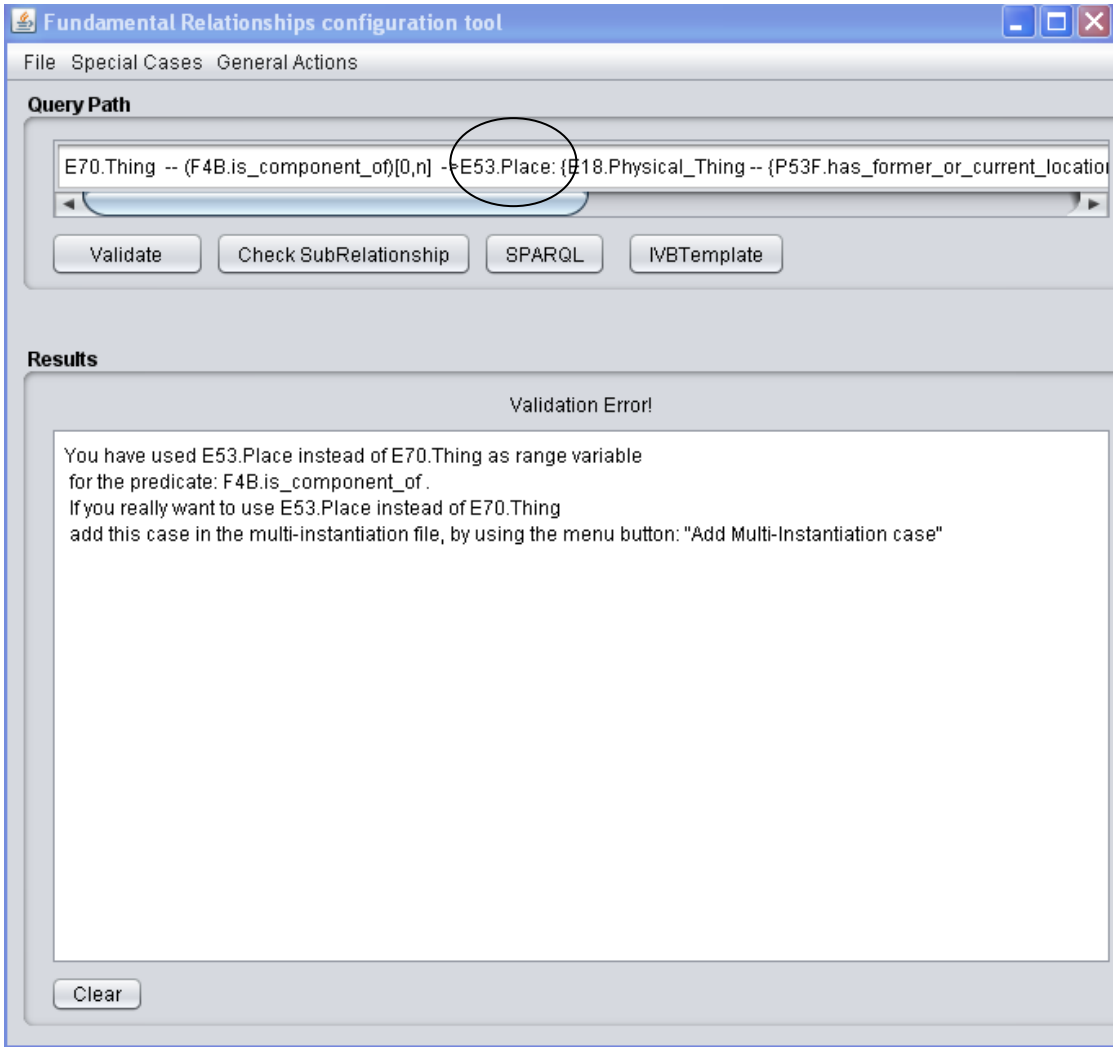

Figure 6: Example of a consistency error

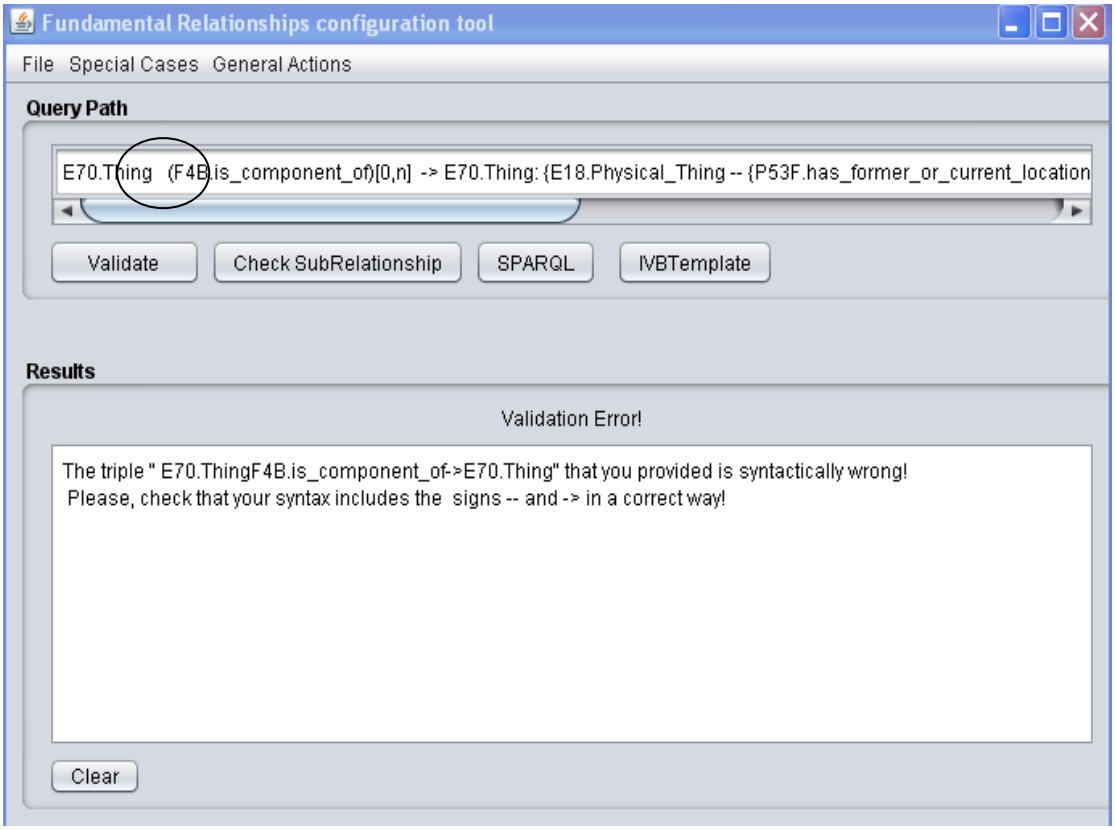

Figure 7: Example of a syntactic error

When the validation is successful, the user might want to save the path as a Fundamental Relationship. In order to save the path as a Fundamental Relationship, the user MUST use the option *File→Save* and provide the name of the Fundamental Relationship in the pop-up that appears. The range and domain FCs are set automatically. The default storage place is indicated by firstly the domain FC and then the domain-range FCs, navigating into the respective folders of the provided set-up package. The path is saved under the filename:"<domainFC><FundamentalRelationship><rangeFC>Path.txt". Figure 8 and Figure 9 illuminate the procedure.

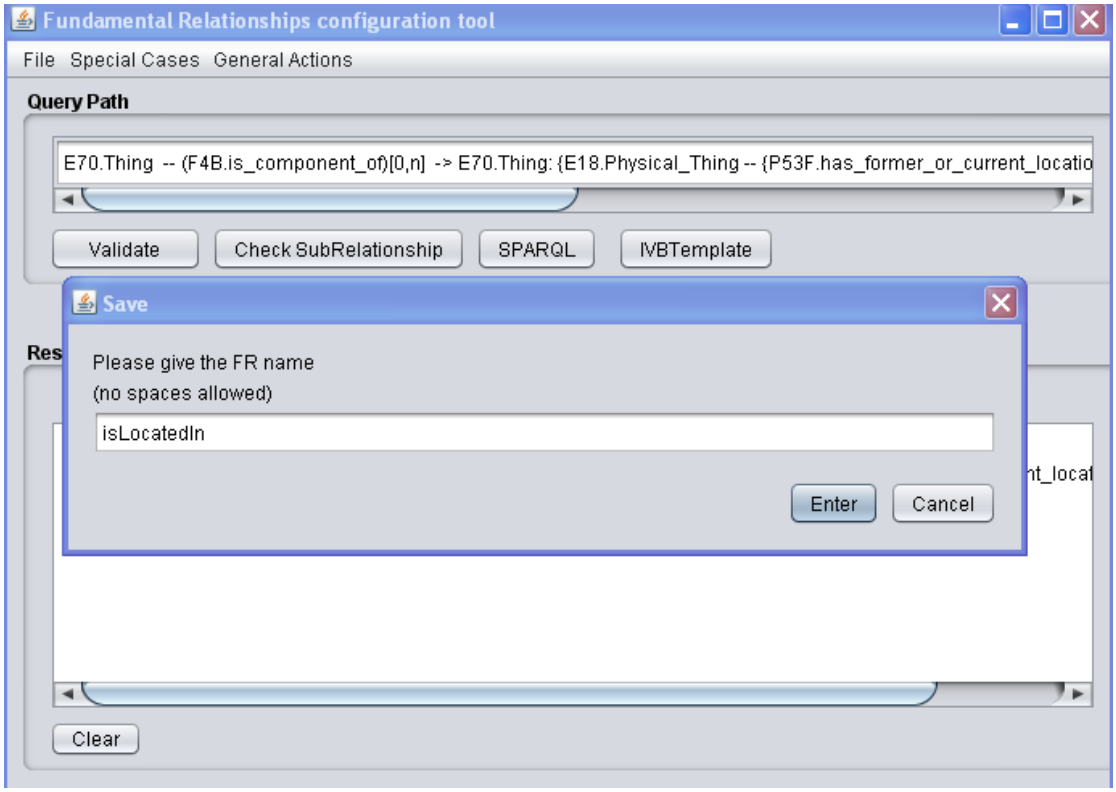

Figure 8: Define FR name for saving

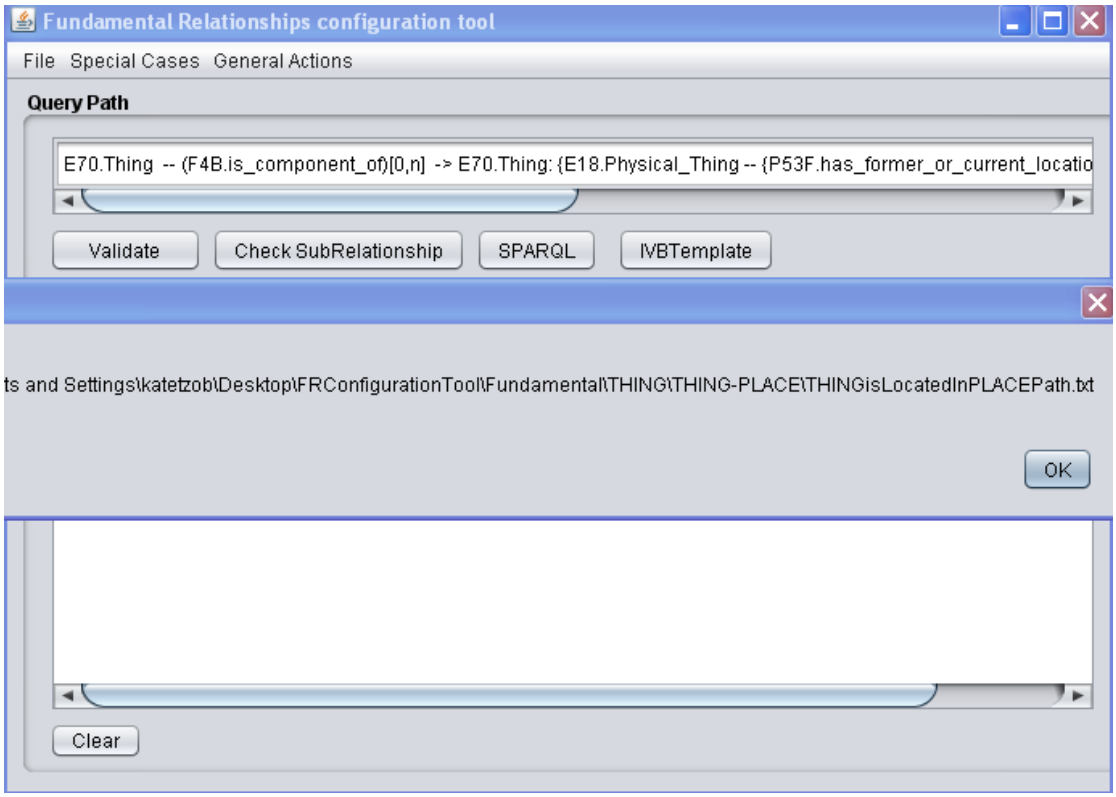

Figure 9: FR saved under displayed path

The step of successful path validation is necessary in order for the user to proceed with the other functions performed on the path. If and only if the validation is valid, can the user proceed with the other functions since the respective buttons are disabled in any other case.

4.2.3.2. Path to SPARQL Translation

When the user has checked the provided path for correctness and validation has been successful, he may want to create the respective SPARQL query statement from it. To do so, he uses the *SPARQL* button. When the translation is complete, the SPARQL statement is displayed in the Results field. In this SPARQL statement the asked-for variable is the $StartVar and along with it any existing labels of it, returned in the $Label variable. The parameter must be set in the $Endvar variable. The respective SPARQL for the path given in this example is:

```
select distinct $StartVar $Label {
$StartVar  rdf:type crm:E70.Thing.
optional { $StartVar crmdig:L4F.has_preferred_label $Label. }
{
{ $StartVar crm:P53F.has_former_or_current_location $var0}
UNION{ $StartVar crm:P54F.has_current_permanent_location $var0}
 $var0 crm:P89F.falls_within $Endvar.}
UNION{$StartVar  crm:F4B.is_component_of $var1.
{ { $var1 crm:P53F.has_former_or_current_location $Endvar.
}UNION{ $var1 crm:P54F.has_current_permanent_location $Endvar.}
}
UNION{{ $var1 crm:P53F.has_former_or_current_location $var2
}UNION{ $var1 crm:P54F.has_current_permanent_location $var2
} $var2 crm:P89F.falls_within $Endvar.}
 }
 UNION {
{ { $StartVar crm:P53F.has_former_or_current_location $Endvar.
}UNION{ $StartVar crm:P54F.has_current_permanent_location $Endvar. }
}}}
```

The translation of the user's path to SPARQL query language is performed following an algorithm that is designed taking under consideration the paths' language grammar. In general, joins among triples (':') are translated to SPARQL joins ('.') and disjunctions ('OR') are translated to SPARQL disjunctions ('UNION'). The

transformation o f a triple to SPARQL, entails the transformation of the domain and range classes of the triple to different variables in SPARQL. The predicate is used in the SPARQL after placing in the front the respective namespace prefix (either “crm:” or “crmdig:” in our case of CIDOC-CRM and CIDOC-CRMdig). In order to maintain the continuity of the triple chain, the variable used as range in a triple, will be used as domain for the next triple in chain. Briefly this algorithm takes as input a path in paths’ language and gives as output a SPARQL statement and is described in natural language in the next lines, omitting some very specific options:

We consider the functions with the following signatures implemented:

- Set<sub-path> FindSubpaths(path P) : calculates the sub-paths of the path P and returns them in a set. More about sub-paths see in 4.2.3.4.

```
Path P = path1;
Set<Path> pathSet = FindSubpaths(P);
integer varNo=0;
startVar =varNo;

Set< Set<triple> > newPathSet;
For each path in pathSet:
        Set<triple> tripleSet = SplitPathToTriples(path);
        Set<triple> newTripleSet;
        For each triple in tripleSet:
                if (firstTriple of path) startVar=”StartVar”;
                        else startvar=varNo;
                if (lastTriple of path) endVar= “Endvar”;
                        else endVar=++ varNo;
                String predicate = findPrefixedPredicate(triple);
                if (!predicate.contains(“OR”))
                        triple = startVar + predicate + endVar;
                else
                        List<predicate> predList= splitPredicates(predicate);
                        triple= ‘{ ‘ + startVar + predList[0]+ endVar +’}’;
                        while(predList.hasNext()){
                                pred= predList.next();
                                triple+= ‘UNION{‘ + startVar + pred + endVar +
                                ‘}’;
                        }
                newTripleSet.add(triple);
        newPathSet.add(newTripleSet);
Sparql = “select $Startvar $label optional { $StartVar
crmdig:L4F.has_preferred_label  $Label. }”
For each tripleSet in newPathSet:
        if (first tripleSet)
                Sparql+=”{”;
                        for each triple in tripleSet
                                Sparql+=triple;
                Sparql+=”}”;
```

```
        else
                Sparql+="UNION{";
                        for each triple in tripleSet
                                Sparql+=triple;
                Sparql+="}";
Sparql+="}";
```

If the user wants to save the output SPARQL then he can use the option *File→Save* and provide the name of the Fundamental Relationship. The domain FC and range FC are set automatically. The storage location for the SPARQLs is the folder *SPARQLs* in the set up package and the filename under which the SPARQL is stored is:

<domainFC><FundamentalRelationship><rangeFC>Sparql.txt

Figure 10 shows an example of path to SPARQL translation.

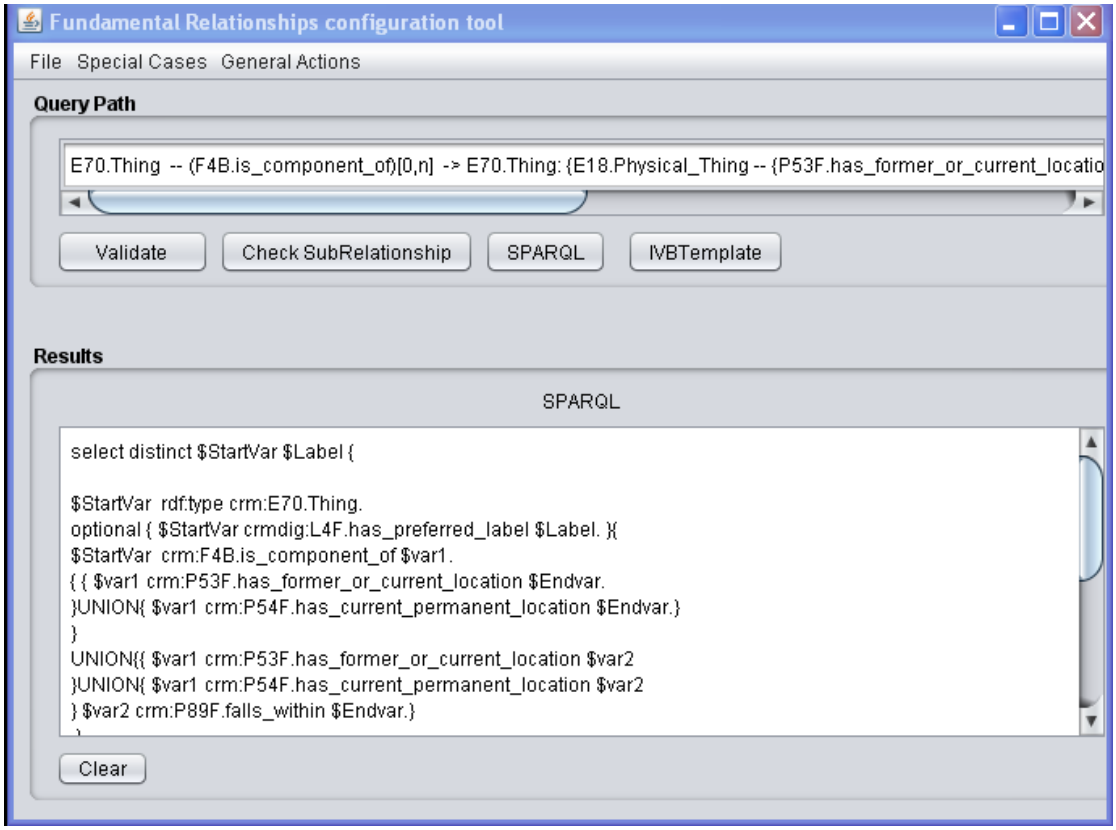

Figure 10: Example of a path to SPARQL translation

4.2.3.3. Path to IVB-Template Translation

This function is built following the same concept as the path to SPARQL translation. It is a specialized function for users of the 3D-COFORM project, who are going to incorporate the SPARQLs into the Integrated Viewer Browser (IVB) [44]

component. For more information about this component, please refer to the Chapter 5: Application and Experiments.

The IVB component requires that the SPARQLs are written in a specific XML formatting, including the domain and range FC and also the name of the FR. For this reason the user, after the press of the *IVBTemplate* button, will be asked for the name of the FR (the domain and range FCs are completed automatically as in the other "save" cases). The IVB template will then appear in the Results field.

If the user also wants to save this template, he may use the option *File→ Save*. As they have already defined the name of the FR for their template, the save will be done automatically and the template will then be found in the folder *IVBTemplates* of the setup package. The name of the saved IVBTemplate will be: <domainFC><FundamentalRelationship><rangeFC>Template.txt

Figure 11 shows an example of path to IVBTemplate translation.

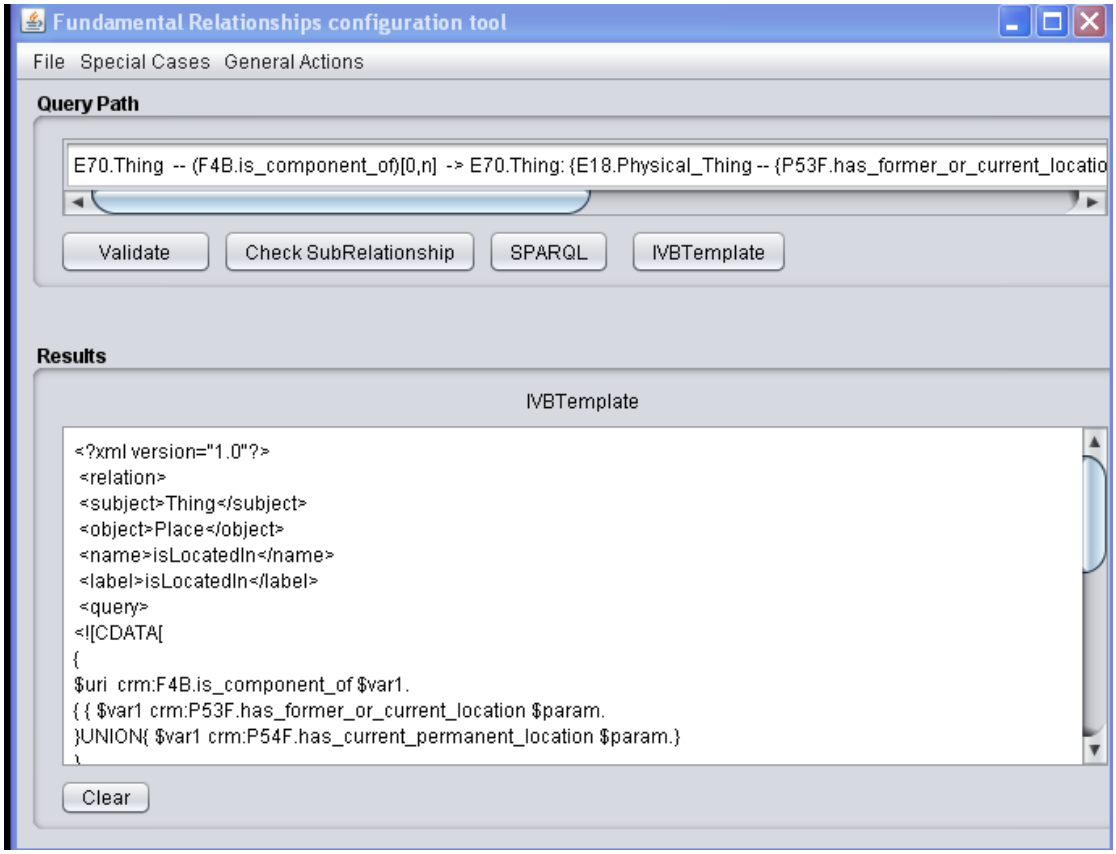

Figure 11: Example of a path to IVB-Template translation

4.2.3.4. Sub-relationship Detector

In section 3.2, where we display the design of the Fundamental Relationships, we mention that we try to include in each FR all the possible interpretations of it over

the CIDOC-CRM schema. In this way we achieve high recall rates. However, this generic design might conclude also to low precision rates. To face this problem we introduce specializations of the FRs or in other words we define *sub-relationships*. A sub-relationship returns a sub-set of the results that the respective FR returns, so will make the query more precise.

When the user writes a path in order to create a sub-relationship of an existing FR or in another case when he wants to find out if his path is already included in a FR, an automatic sub-relationship detection mechanism would be helpful. The sub-relationship detector provides such functionality, that detects sub-relationship relations among the user provided path, and all the already defined Fundamental Relationships. Of course, a path can be a sub-relationship of a Fundamental Relationship only if the first and the second share the same domain and range FCs. This observation is exploited for optimizing the performance of this task.

To check if the provided path (in our example case the provided path "Thing is located in Place") is a sub-relationship, the user can use the *Check SubRelationship* button. When the sub-relationship check is performed, the FRs which include this path as sub-relationship, will be returned to the user in the *Results* field. Figure 12 shows the results of this check; the defined by the user path is sub-relationship of the "Thing from Place" FR.

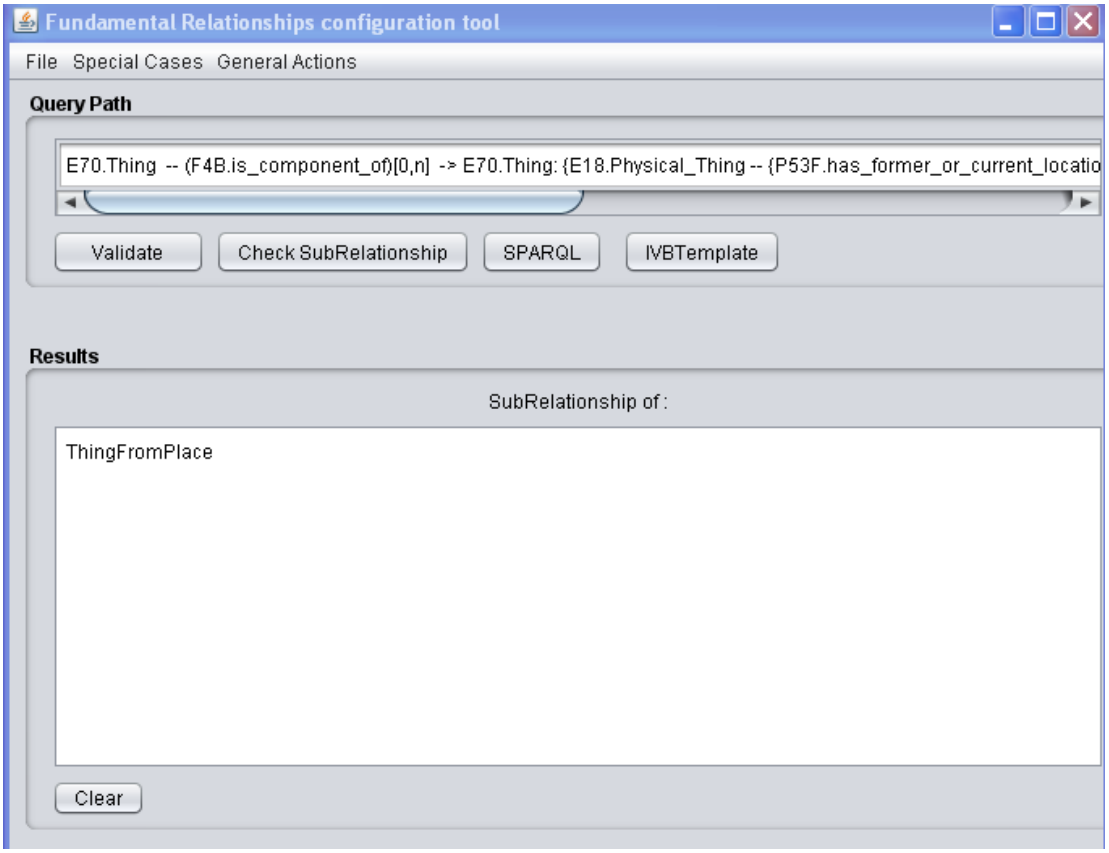

Figure 12: Example of a successful sub-relation finding

To find of which FRs the given path is sub-relationship, we use the notion of the *sub-path* which was introduced in the previous section and is further defined in Definitions 4 and 5 and Task 1.

DEFINITION 4:

Given a path P, a *sub-path* P_x of P is a simple path that has the same domain and range classes with P and is included in P. Domain class is the class with which the path starts and range class is the class to which the path ends. If we consider a path to be a set of vertices and edges, with vertices to represent classes and edges to represent predicates, a *simple* path is a path where each class-node is a start point for at most one predicate and end point for at most one predicate. A path P_x *is included* in a path P, if and only if the sequence of predicates that connect the domain class with the range class in the P_x, also occurs in P.

DEFINITION 5:

Given a path P we define as *sub-path set*, the finite set S:

$$\begin{cases} S = \{P_1, \dots, P_n\}, \text{where } P_x \text{ is a subpath of P}, x \in [1, n], n \in Z^* & \text{if } n > 1 \\ S = \{P\} & \text{if } n = 1 \end{cases}$$

Task 1 is used to define the way in which a path P is split into sub-paths creating the set of sub-paths. This is used as the base of the sub-relationship detector function.

Task 1:

Given a path P, find which sub-paths it consists of.

Let P be a path constructed following the rules in the grammar of Definition 2.

We replace the notion of triple with the notion of the predicate_expr, as here the only part of the triple that is meaningful for the decision is the predicate part. The class in domain and range are implied. So the rules 1,2 and 3 are simplified to:

path = predicate_expr (1)

path = predicate_expr, ":" ,"{", path, "}" (2)

path = predicate_expr, ":", "{" ,path, " OR ", path, "}" (3)

The rest of the rules are kept as they are. In the path P, we make the following change that concerns the transitivity and reflexivity, to imply that we either include (P) or not include (1) the predicate P:

predicate_trans = "(","P",")","[0,n]" → P OR 1

In this way, we simplify the disjunction cases to occur only in "triple level" since now a triple is represented by only a predicate. Different triples are delimited with the use of ':' .

Next step, is to separate the different paths that exist in the path P. To do so we proceed as in the following:

Let a column vector

$$A = \begin{matrix} a_1 \\ \cdots, j \in Z^*, \ a_i \neq a_m, \forall i \neq m \\ a_j \end{matrix} .$$

For each OR block of predicates encountered in the path P, we create a column vector $A_{i=0..k}$, k the number of OR statements.

$$\{P_1 \text{ OR } P_2 \dots \text{OR } P_n\} \rightarrow A = \begin{matrix} P_1 \\ \dots \\ P_n \end{matrix}$$

One single P_l in the path P is considered as an array with one value.

$P_l \rightarrow A = [P_l]$

If we erase from the path all the brackets, and replace the ':' sign with the multiplication sign '*', we are going to end to a format like:

$P=A_1*A_2{\cdot\cdot}*A_s$, where s=number of(*) - 1.

Definition

We define as multiplication among column vectors, notated with A*B, where

$A = \begin{matrix} a_1 \\ \cdots \\ a_n \end{matrix}$ and $B = \begin{matrix} b_1 \\ \ldots \\ b_m \end{matrix}$, with n≥m , the function that results into a new column vector C,

of size n*m where:

$$C = \begin{matrix} c_1 = a_1 b_1 \\ \ldots \\ c_{n*m} = a_n b_m \end{matrix}$$

Using the definition above, we end to that P is a column vector representing the different sub-paths set that it consists of.

$P=\{P_1,P_2..P_z\}$, $z \in Z^*$

To add a practical example of the previous theory, let us consider the path:

P= E53.Place --P89B.contains -> E53.Place : {E53.Place -- {P122F.borders_with OR P121F.overlaps_with }-> E53.Place:{E53.Place --P89F.falls_within -> E53.Place}}

This path, following the grammar rules' simplification (1) to (3) can be re-written to the following path, using only predicates:

P=P89B.contains:{{ P122F.borders_with OR P121F.overlaps_with }:

{P89F.falls_within }} (A)

Next step is two replace the predicate expressions with the respective vectors, which are the following:

P89B.contains $\rightarrow A_1 = P89B.\text{contains}$

P122F.borders_with OR P121F.overlaps_with $\rightarrow A_2 = \begin{matrix} P122F.\text{borders\_with} \\ P121F.\text{overlaps\_with} \end{matrix}$

P89F.falls_within $\rightarrow A_3 = P89F.\text{falls\_within}$

So, replacing A_1, A_2 and A_3 in (A) we get the following expression for the path P:

P= A_1:{{ A_2}: {A_3}}

If we remove the brackets, replace the ':' sign with the '*' sign and use the definition for the multiplication of column vectors that we have described formerly we get:

$$P= A_1^* A_2^* A_3 \xrightarrow[A2*A3]{} P= A_1^* \begin{matrix} P122F.\,borders\_with\ P89F.\,falls\_within \\ P121F.\,overlaps\_with\ \ P89F.\,falls\_within \end{matrix} \xrightarrow[A1*(A2*A3)]{}$$

$$P= \begin{matrix} P89B.\,contains\ P122F.\,borders\_with\ P89F.\,falls\_within \\ P89B.\,contains\ P121F.\,overlaps\_with\ \ P89F.\,falls\_within \end{matrix}$$

So, we have ended to a 2X1 vector P that contains as elements the different sub-paths of the path P; that is in our example:

P_1 = P89B. contains: P122F. borders_with: P89F. falls_within

and

P_2 = P89B. contains: P121F. overlaps_with: P89F. falls_within

Now, in order to apply the sub-paths theory to solve the sub-relationships detection, we follow the following idea:

Let F be an FR and P a path. As described above the FR F is also a path. So the path P and the path F can be written as sets of sub-paths following the definition 5.

$F=\{F_1,\ldots,F_n\}$, $n \in Z^*$

$P=\{P_1,\ldots,P_m\}$, $m \in I^*$ $m \in Z^*$

In order for the P to be sub-relationship of F, we set the following constraints:

domain(F)=domain(P) and range(F)=range(P) (1)

$n \geq m$ (2)

$\forall P_k = p_1 \ldots p_x \in P, \exists F_k = f_1 \cdots f_x, k \leq m \text{ s.t } p_i = f_i, i \leq x$ (3)

where domain(x) represents the domain class of x and range(x) represents the range class of x.

In other words, F and P must share the same domain and range classes, as indicated in the first constraint. F must have more or the same number of sub-paths than P as indicated in the second constraint. The thrid constraint demands that for a given sub-path P_k of P, that is translated to a chain of properties as described in Definition 5s, there must be a sub-path F_k of F, whose property chain contain the property chain of P_k. Contains means that either the properties in the same place in the chains are identical or the one contained in the P_k is subProperty of the one contained in the F_k.

So, the notion of equation = that is used between the properties of the two different paths, has the two following meanings:

$$p_i = f_i \Rightarrow \begin{cases} p_i \equiv f_i \\ p_i \text{ is subproperty of } f_i \end{cases} \quad (4)$$

To further clarify the adopted solution, let's see how the tool decided that the provided by the user path is a sub-relationship of the already defined FR "Thing from Place".

Here, P = **E70.Thing** -- (F4B.is_component_of)[0,n] -> **E70.Thing:**
{E18.Physical_Thing --
{P53F.has_former_or_current_location OR
P54F.has_current_permanent_location}->**E53.Place :**
{E53.Place --(P89F.falls_within)[0,n] ->**E53.Place**
}
}

and F= **E70.Thing** -- (F4B.is_component_of)[0,n] -> **E70.Thing:**
{E70.Thing--{P53F.has_former_or_current_location OR
P54F.has_current_permanent_location}-> **E53.Place**:
{E53.Place --(P89F.falls_within)[0,n]-> **E53.Place**
}
OR
E70.Thing -- P92B.was_brought_into_existence_by ->
E63.Beginning_of_Existence :
{E63.Beginning_of_Existence --
(P9B.forms_part_of)[0,n]-> **E5.Event** :
{E5.Event -- P7F.took_place_at -> **E53.Place**:
{E53.Place --(P89F.falls_within**)[0,n]->**
E53.Place }
}
}
}

Note: The path presented here as F, is only a part of the whole path of the "FR Thing from Place" but for the sake of the reader's convenience, this part is sufficient to describe the problem.

If we split the aforementioned paths to sub-paths set, we will get the following tables, in Figure 13 and Figure 14. In these figures, each line represents a sub-path and each column represents the subsequent triple. Notice that all paths begin with the FC Thing and end with the FC Place.

Sub-Paths of P		
E70.Thing -- F4B.is_component_of -> **E70.Thing**	**E18.Physical_Thing** -- P53F.has_former_or_current_location-> **E53.Place**	**E53.Place** -- P89F.falls_within -> **E53.Plac**e
E70.Thing -- F4B.is_component_of -> **E70.Thing**	**E18.Physical_Thing** -- P54F.has_current_permanent_location-> **E53.Place**	**E53.Place** -- P89F.falls_within -> **E53.Plac**e
E70.Thing -- F4B.is_component_of -> **E70.Thing**	**E18.Physical_Thing** -- P53F.has_former_or_current_location-> **E53.Place**	
E70.Thing -- F4B.is_component_of -> **E70.Thing**	**E18.Physical_Thing** -- P54F.has_current_permanent_location-> **E53.Place**	
E18.Physical_Thing -- P53F.has_former_or_current_location-> **E53.Place**	**E53.Place** --P89F.falls_within -> **E53.Plac**e	
E18.Physical_Thing -- P54F.has_current_permanent_location-> **E53.Place**	**E53.Place** --P89F.falls_within -> **E53.Plac**e	
E18.Physical_Thing -- P53F.has_former_or_current_location-> **E53.Place**		
E18.Physical_Thing -- P54F.has_current_permanent_location-> **E53.Place**		

Figure 13: Sub-paths of P

Sub-Paths of F				
E70.Thing -- F4B.is_component_of -> **E70.Thing**	**E18.Physical_Thing** -- P53F.has_former_or_current_location- >**E3.Place**	**E53.Place** -- P89F.falls_within -> **E53.Plac**e		
E70.Thing -- F4B.is_component_of -> **E70.Thing**	**E18.Physical_Thing** -- P54F.has_current_permanent_location-> **E53.Place**	**E53.Place** -- P89F.falls_within -> **E53.Plac**e		
E70.Thing -- F4B.is_component_of -> **E70.Thing**	**E18.Physical_Thing** -- P53F.has_former_or_current_location-> **E53.Place**			
E70.Thing -- F4B.is_component_of -> **E70.Thing**	**E18.Physical_Thing** -- P54F.has_current_permanent_location- >**E53.Place**			
E18.Physical_Thing -- P53F.has_former_or_current_location-> **E53.Place**	**E53.Place** --P89F.falls_within - >**E53.Plac**e			
E18.Physical_Thing -- P54F.has_current_permanent_location-> **E53.Place**	**E53.Place** --P89F.falls_within - >**E53.Plac**e			
E18.Physical_Thing -- P53F.has_former_or_current_location-> **E53.Place**				
E18.Physical_Thing -- P54F.has_current_permanent_location-> **E53.Place**				
E70.Thing -- F4B.is_component_of -> **E70.Thing**	**E70.Thing** -- P92B.was_brought_into_existence_by -> **E63.Beginning_of_Existence**	**E63.Beginning_of_Existence** -- P9B.forms_part_of-> **E5.Event**	**E5.Event** -- P7F.took_place_at-> **E53.Place**	**E53.Place** -- P89F.falls_within-> **E53.Place**
E70.Thing -- F4B.is_component_of -> **E70.Thing**	**E70.Thing** -- P92B.was_brought_into_existence_by -> **E63.Beginning_of_Existence**	**E5.Event** -- P7F.took_place_at-> **E53.Place**	**E53.Place** -- P89F.falls_within-> **E53.Plac**e	
E70.Thing -- F4B.is_component_of -> **E70.Thing**	**E70.Thing** -- P92B.was_brought_into_existence_by -> **E63.Beginning_of_Existence**	**E5.Event** -- P7F.took_place_at-> **E53.Place**		
E70.Thing -- P92B.was_brought_into_existence_by-> **E63.Beginning_of_Existence**	**E63.Beginning_of_Existence** -- P9B.forms_part_of-> **E5.Event**	**E5.Event** -- P7F.took_place_at-> **E53.Place**	**E53.Place** -- P89F.falls_within-> **E53.Plac**e	
E70.Thing -- P92B.was_brought_into_existence_by-> **E63.Beginning_of_Existence**	**E63.Beginning_of_Existence** -- P9B.forms_part_of-> **E5.Event**	**E5.Event** -- P7F.took_place_at-> **E53.Place**		
E70.Thing -- P92B.was_brought_into_existence_by-> **E63.Beginning_of_Existence**	**E5.Event** -- P7F.took_place_at-> **E53.Place**	**E53.Place** -- P89F.falls_within -> **E53.Place**		
E70.Thing -- P92B.was_brought_into_existence_by-> **E63.Beginning_of_Existence**	**E5.Event** -- P7F.took_place_at-> **E53.Place**			

Figure 14: Sub-paths of F

Now that we have the sub-paths sets, we have to verify if the three constraints (1), (2) and (3) declared above occur. Firstly both paths have the E70.Thing as domain class and the E53.Place as range class. The second constraint is satisfied, since we indeed have that the number of sub-paths of F (=16) is bigger than the number of the

sub-paths of P (=9). For the third constraint, we can see from the tables above that the sub-paths of P appear in the first 9 sub-paths of the sub-paths' table of F. So, both constraints are true and we can define that P is a sub-relationship of F.

Note here, that if instead of the last sub-path of P, the "E18.Physical_Thing -- P53F.has_former_or_current_location->E53.Place" we had for instance the sub-path "E18.Physical_Thing -- P55F.has_current_location-> E53.Place" again P would be a sub-relationship of F. This happens because P55F.has_current_location is a sub-property of P53F.has_former_or_current_location, a case that is also considered as property-equality according to the formula (4). As "E18.Physical_Thing -- P55F.has_current_location-> E53.Place" of P appears at the same level as "E18.Physical_Thing --P53F.has_former_or_current_location->E53.Place" of F and the respective sub-paths are of the same length (=1), the constraint (3) is again satisfied.

The algorithm behind this solution, is based on checking the constraints 1-3. We consider the functions with the following signatures implemented:

- Set<sub-path> FindSubpaths(path P) : calculates the sub-paths of the path P and returns them in a set
- Class range(path P) : Finds the range class (the last class of the path) and returns it as an object of "Class" type
- Class domain(path P) : Finds the domain class (the first class of the path) and returns it as an object of "Class" type

Also we consider the following types:

- sub-path := List<properties> : the sub-path is represented as a list of subsequent properties
- Class := String : this is a string representing a class of the CRM model

```
Path P=path1;
Path F=path2;
Boolean found=false, allFound=true;  //allFound for all subpaths, found for all triples
if(range(P)==range(F) && domain(P)==domain(F)){
        subpathSetP= FindSubpaths(P);
        subpathSetF= FindSubpaths(F);
        if(subpathSetF.size() >= subpathSetP.size()){
                for(Sub-path p: subpathSetP){
                allFound=false;
                 changF:for(Sub-path f: subpathSetF){
                                found=false;
                                for (int i=0; i<= p.size();i++){
                                        if(isEqual(p[i], f[i])){
                                            found=true;
                                            if(i==p.size)
                                                {allFound=true; break changeF;}
                                        }
                                        else found=false;
                                        if (!found) break;
                        }       }
                        if(!allFound) break;
                }
                if (allFound) return true;
                else return false;
        }else return false;
}else return false;

Boolean isEqual(predicate p, predicate f){
        if(p==f) return true;
        if (p.isSubpropertyOf f) return true;
        else return false;
}
```

4.2.3.5. Schema-Coverage Checker

The Fundamental Categories and Relationships model aims at providing an abstract schema of the CIDOC-CRM schema. For this reason, we need to check if the new model covers the more specialized one, in other words to check in what degree the properties of the CIDOC-CRM schema are included in the Fundamental Relationships specified. For a schema property to be covered by the proposed model, it must occur that either the property itself or a super-property of it is included in the model.

The schema coverage checker provides this opportunity. This function does not require any input from the user. In order to perform it, they must select the option *General Actions→Check Schema Coverage*. Then in the *Results* field the uncovered properties of the schema will appear.

Some other interesting observations from the usage of this function can be remarked.

- Spot errors in the schema: From our experience using this function, we observed that certain properties of the schema had incomplete declaration for example "crm:P14F.1_carried_out_by_as_author" should have been declared as rdf:subPropertyOf "crm:P14B.performed"
- Spot errors at the metadata: Also we observed errors at the stored metadata, as they included properties with wrong namespaces for instance: "crm:L22F.created_derivative". L22F.created_derivative is a property from the CIDOC-CRM digital and should have the namespace 'crmdig' instead of 'crm'. Thus, the correct choice would be "crmdig: L22F.created_derivative".

Figure 15 shows an example of schema coverage check.

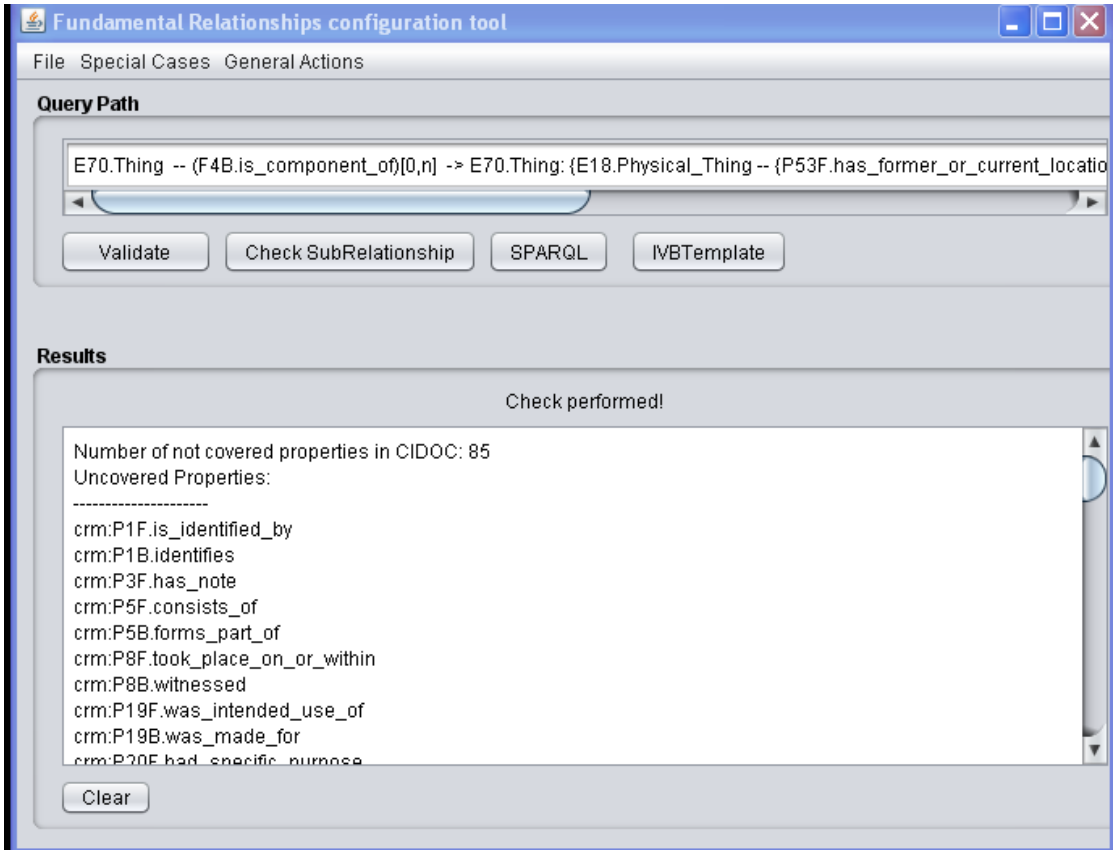

Figure 15: Example of finding schema uncovered properties

4.2.3.6. Rules Detector

It is not rare that for reasoning and performance reasons, extra custom rules will be added in the repository in some time in the future after the FCs-FRs model has

been defined. In this way, inferences are created that can be used also in queries in order to make them more effective. It is essential then, that we provide the user with some maintenance aid, in order to save them effort and time on searching the Fundamental Relationships for possible alterations due to new reasoning.

To meet this need we have created the rules detector function, which can be found under the *General Actions→Check for new rules*. This function reads the custom crm rules of the repository and searches in all the FR paths in order to detect possible matches of the rules *premises*. As premises we define the pattern that must occur in the metadata and which leads to the creation of the *consequences.* Consequences are the inferences created in the repository, thus the implicit triples [45]. So, if a rule's premises are found in some FRs, these FRs are indicated to the user with extra information about the path that can be replaced in the FR by the new triple indicated by the rule consequences. In order to find if a rule premise occurs in a path, the program transforms the premises to paths' language format. Then the path is split into the sub-paths it contains, as described in section 4.3.2.4. In this case, the subpath set contains only one subpath, as the rules describe one sequence of triples that must occur in order to create the inference. For every defined FR, we create the subpath set it corresponds to and check whether the path of the rule exists within any subpath of a given FR. If it exists, then the name of the FR is returned together with the consequences, written again in paths' language.

This function searches in all the paths of the FRs to find matching patterns, which is a time-consuming process. So, a message appears in order to aware the user about this. Figure 16 shows such a check, where rules can be replaced in some FRs' paths.

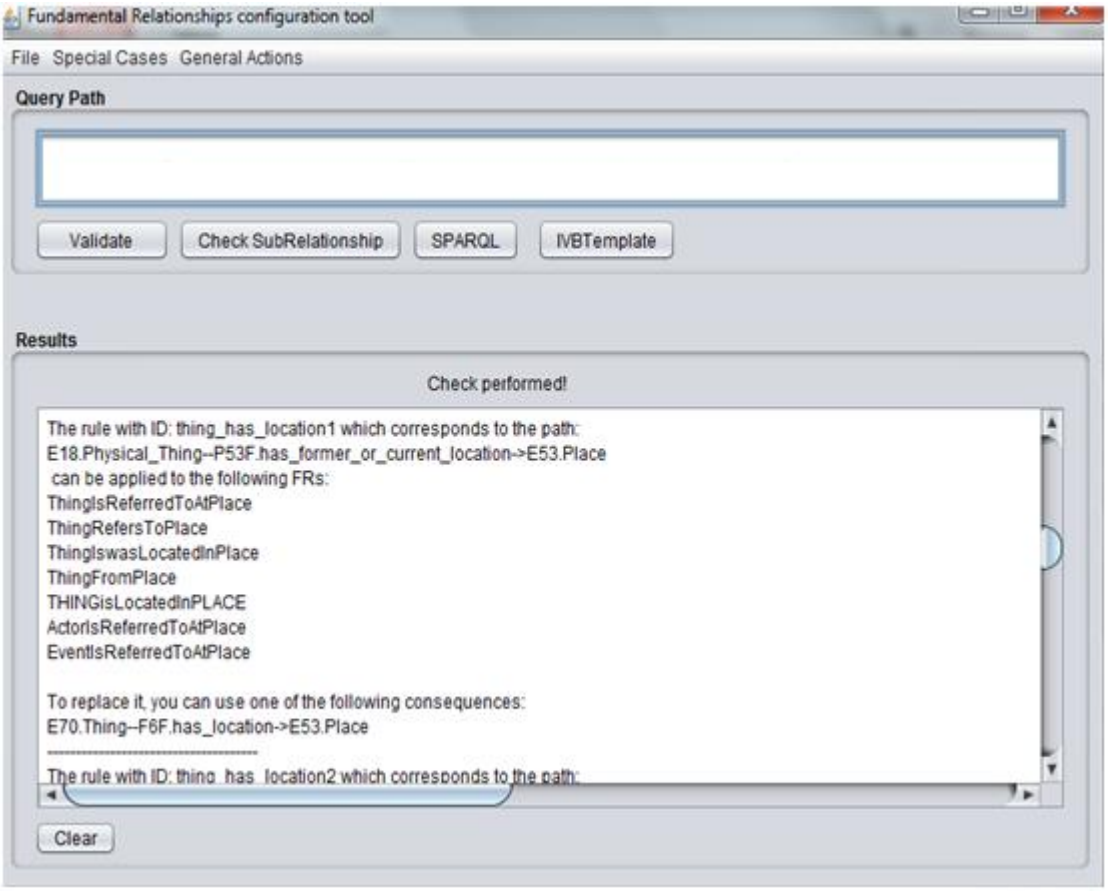

Figure 16: Example of finding that a rule has been defined in the repository and can be changed in some paths.

4.2.3.7. Multiple Instantiation and Disjoint Cases Handler

When using the CIDOC-CRM schema, the user is allowed to declare that one object identifier (URI or UUID) belongs to more than one classes. This is an act of multiple instantiation. Multiple instantiation greatly simplifies the picture of the way the world works, as we can omit for certain cases the class hierarchy that means complexity. In the same concept, in the mapping of the FCs-FRs model to CRM paths we allow multiple instantiation, although the user must declare which classes can be used in the place of the other; in other words which classes' instances can also be instances of which other classes. The multiple instantiation is hierarchical, so it is inherited to the subclasses of the involved classes as well.

In the validation process when a consistency error is found in the path, i.e. when instead of an expected classA a classB is found (see case b from 4.2.3.1 paragraph), there will appear a suggestion message for the user: If they wish to accept

this “invalidity” they may declare that classA can be replaced by classB, thus instances of classA can also be instances of classB. If such a declaration has already been made, the program will not display the message and will proceed normally.

In our example case the user instead of using the ‘E70.Thing’ class, he used the ‘E53.Place’. So, if he wants to state that this was done intentionally and they wish to use E53.Place in the place of E70.Thing he must add this multiple instantiation case. In order for the user to add or remove multiple instantiation cases, he shall use the option *Special Cases→Add Multiple Instantiation Case* or the *Special Cases-> Remove Multiple Instantiation Case* respectivelty. Figure 17 shows the example when the user states that he wants to use E53.Place instead of E70.Thing. So he has to write in the pop-up window: “E70.Thing:E53.Place”. In an analogous manner the multiple instantiation removal is performed.

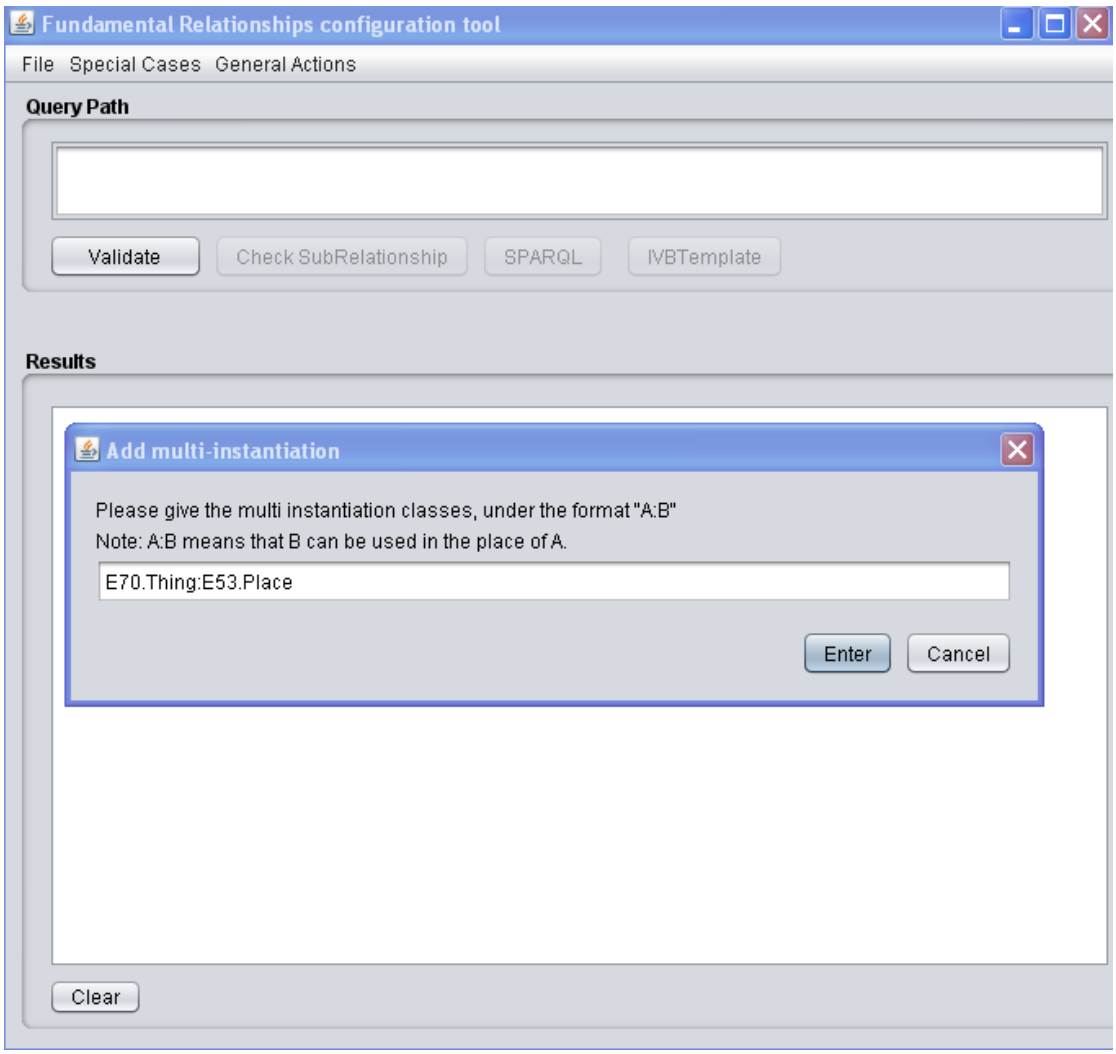

Figure 17: Example of defining a multi-instantiation case

There have been already defined some multiple instantiation cases, driven from the frequent use of the instances of the one as instances of the other. These are:

- E53.Place:E19.Physical_Object, E39.Actors. Physical objects are often referred to as Places for example museums, organizations etc
- E5.Event:E52.Time-Span. Time spans can be considered as events for example 1821 frequently denotes the Greek revolution of 1821
- E52.Time-Span:E5.Event. Often instead of specifying the time spans, we use instead the event for example we may say: "When Nikos was born, Thessaloniki was burned" meaning on 1917 and not the actual birth event of Nikos
- E7.Activity:E6.Destruction. A destruction in the CIDOC-CRM schema is considered not to be an activity, thus can't be performed by someone. Nevertheless, it is useful many times that we link it with an actor that responsible for it, so we may use it also as an activity.

Along with the multiple instantiation cases, we also enable the option for disjoint cases. If a class is declared to be disjoint with some other class, this means that the two classes can't share instances. The disjoint property is hierarchical and symmetrical, meaning that it is inherited to the sub-classes of the classes and is also valid vice a versa. That is if classA is disjoint with classB then also classB is disjoint with classA.

Disjointness puts restrictions to which classes can be multiple instantiated with other classes. So, if a classA has been declared disjoint with classB, this means that even if classB and classA have been declared as a multiple instantiation case, they can't be used so. Disjointness is more powerful than multiple instantiation.

In order for the user to add or remove multiple instantiation cases, he shall use the option *Special Cases→Add Disjointness Case* or the *Special Cases→Remove Disjointness Case.* Figure 18 shows the example of how the user can define that E53.Place is disjoint with E70.Thing. Note that he must run twice the case, writing E53.Place: E70.Thing and vice a versa E70.Thing: E53.Place. Disjoint case removal can be done in an analogous way.

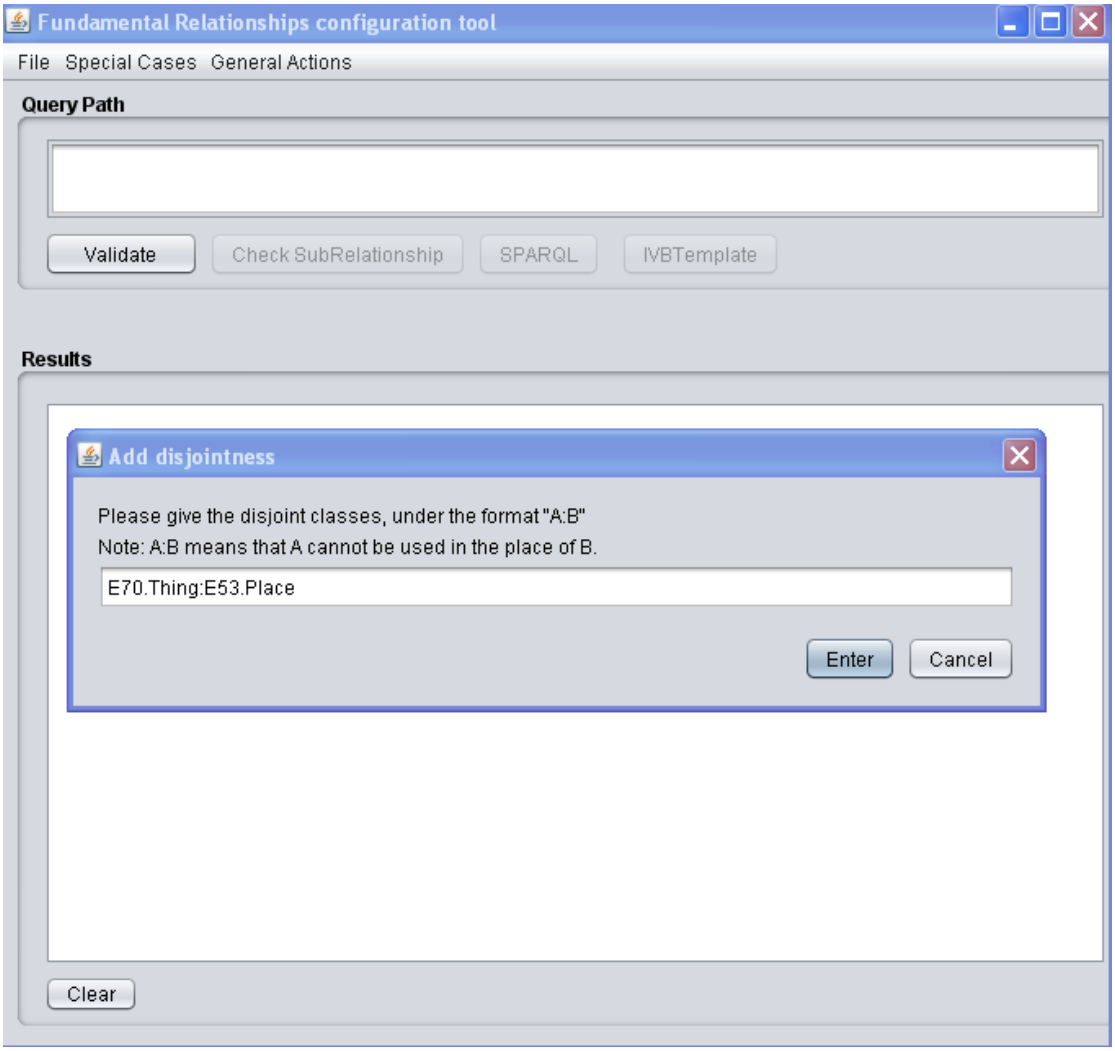

Figure 18: Example of finding a disjointness case

We have already defined for the FCs-FRs models the following disjoint cases. These have been derived from the CIDOC-CRM proposal for declaration of all disjoint classes[25].

- E2.Temporal_Entity:E77.Persistent_Item, E53.Place, E54.Dimension, E52.Time-Span, E59.Primitive_Value
- E77.Persistent_Item:E2.Temporal_Entity, E53.Place, E54.Dimension, E52.Time-Span, E59.Primitive_Value
- E53.Place:E54.Dimension, E52.Time-Span, E59.Primitive_Value, E77.Persistent_Item, E2.Temporal_Entity
- E54.Dimension:E52.Time-Span, E59.Primitive_Value, E53.Place, E77.Persistent_Item, E2.Temporal_Entity

[25] http://www.cidoc-crm.org/issues.php?id=92

- E52.Time-Span:E59.Primitive_Value, E2.Temporal_Entity, E77.Persistent_Item, E53.Place, E54.Dimension
- E39.Actor:E24.Physical_Man-Made_Thing, E73.Information_Object, E41.Appellation, E51.Contact_Point, E55.Type
- E24.Physical_Man-Made_Thing:E39.Actor
- E51.Contact_Point:E39.Actor
- E73.Information_Object:E39.Actor
- E41.Appellation:E39.Actor
- E18.Physical_Thing:E28.Conceptual_Object
- E28.Conceptual_Object:E18.Physical_Thing
- E26.Physical_Feature:E19.Physical_Object
- E19.Physical_Object:E26.Physical_Feature
- E55.Type:E30.Right,E39.Actor
- E30.Right:E55.Type
- E56.Language:E57.Material,E58.Measurement_Unit
- E58.Measurement_Unit:E56.Language
- E57.Material:E58.Measurement_Unit, E56.Language

4.3. Fundamental Relationships Materialization on CIDOC-CRM

As stated before the proposed model, even though it can be customized for a wide range of different discourses, it has been designed for the cultural heritage domain. CIDOC-CRM is an adequate model to describe this complex world and is mostly used as said before in many integration projects. For this reason, we have chosen to implement this framework in the context of the CIDOC-CRM [46] schema and the CIDOC-CRMdig [9], an extension of the first aiming at describing the digitization of objects. The main objective is to include all relative - and if applicable all - the properties of the schema in paths representing the FRs.

In order to create the corresponding paths of the FRs, we have used the paths' language with the aid of the Fundamental configuration tool. Let's now see an example of a Fundamental Relationship materialized in CIDOC-CRM: the "Thing is about or refers to Thing." The respective created path is the one that follows:

```
E70.Thing --(F5F.consists_of_shows_features_of)[0,n]-> E70.Thing:
  { E24.Physical_Man-Made_Thing -- P62F.depicts -> E70.Thing:
        {E70.Thing -- (F4F.is_composed_of)[0,n]-> E70.Thing
        }
   OR
    E89.Propositional_Objec--P67F.refers_to-> E70.Thing:
        {E70.Thing -- (F4F.is_composed_of)[0,n]-> E70.Thing
        }
   OR
   E24.Physical_Man-Made_Thin -- P128F.carries -> E73.Information_Object:
        { E73.Information_Object--P67F.refers_to -> E70.Thing:
            {E70.Thing -- (F4F.is_composed_of)[0,n]-> E70.Thing
            }
         }
   OR
   D1.Digital_Object -- L11B.was_output_of -> D7.Digital_Machine_Event:
         {D7.Digital_Machine_Event--(P9B.forms_part_of)[0,n] ->  D2.Digitization_Process:
               {D2.Digitization_Process -- L1F.digitized ->E70.Thing:
                     {E70.Thing -- (F4F.is_composed_of)[0,n]-> E70.Thing
                     OR
                     E24.Physical_Man-Made_Thing-- P62F.depicts -> E70.Thing:
                           {E70.Thing -- (F4F.is_composed_of)[0,n]-> E70.Thing
                           }
                     OR
                     E89.Propositional_Object--P67F.refers_to-> E70.Thing:
                           {E70.Thing -- (F4F.is_composed_of)[0,n]-> E70.Thing
                           }
                     OR
                     E24.Physical_Man-Made_Thing -- P128F.carries ->
                        E73.Information_Object:
                           { E73.Information_Object --P67F.refers_to ->  E70.Thing:
                              {E70.Thing -- (F4F.is_composed_of)[0,n]-> E70.Thing
                              }
                           }
                  }
            }
         }
    }
```

Starting from the domain FC “Thing” and ending at the range FC “Thing”, we have searched in the schema(s) for all the possible interpretations of the “is about or refers to” FR. This process concludes to the properties “P62F.depicts” and “P67F.refers_to” and also a chain of the properties “P128F.carries” and “P67F.refers_to” is included. Nevertheless, a restriction to only a flat usage of these properties would yield low recall. To overcome this undesired outcome, we add some extra reasoning to construct inferences, taking into consideration the propagation of the subject (theme) of an object through part/whole relationships, object copies and derivation chains. So, we infer that if a thing1 that has a part that refers to a thing2, then also thing1 refers to thing2; if a thing1 shows features of another thing that refers to thing2 then also thing1 refers to thing2; if a thing2 has been digitized and the

derivative is thing1, then the thing1 refers to thing2; if a thing that refers to thing2 has been digitized and the derivative is thing1, then the thing1 also refers to thing2. In all these cases above thing2 can also be a part of the explicitly referred to thing. If we include all these possible paths from domain to range FCs we end to the complex path above. As we have already stated though, we accept that there is only a probability entailed in the participation of each of the above in the respective FR, but we accept it so that we improve the recall rate.

In the same manner we have constructed all the FRs combined with the applicable FCs in domain and range. The complete list of these paths is displayed in Appendix A.

CHAPTER 5

Application and Experiments

The proposed work has been accepted and is being implemented in the context of two big European projects, the 3D-COFORM Project and the Research Space Project. These projects follow different approaches for the implementation of the model, as described later in this chapter. Along with these applications, we have collected real scientific queries from the 'Centre for documentation of cultural and natural heritage'[26] and we have proven that they can be successfully mapped to our method. The use of the implemented software has also been verified in the context of the 3D-COFORM Project and the implementation of the model in the CIDOC-CRM schema. Finally, we have tested a series of the Fundamental Relationships. This chapter describes all the above, proving the value and feasibility of the method.

5.1. 3D-COFORM Project

The 3D-COFORM[27] project aims at providing integrated technologies to make the large-scale production of 3D models feasible for the systematic documentation and study of material cultural heritage. For that purpose, it combines leading-edge technologies for 3D model generation from acquired data (photographic or laser), generation of synthetic models and presentations. Underlying there is a scalable repository infrastructure (RI) used to manage integrated, distributed data and metadata about cultural-heritage objects themselves, digital representations of them, and scholarly and scientific annotations. The RI contains a metadata repository [44] [47] which is implemented on SESAME[28] with an OWLIM[29] reasoner on top. The repository provides the platform and semantics to manage objects, 3D models and other presentations alike. It also supports the scholarly discourse on recent and past object features in archaeology, sites and monuments management, museum

[26] http://www.cultnat.org/
[27] http://www.3d-coform.eu/
[28] http://www.openrdf.org/
[29] http://www.ontotext.com/owlim/

disciplines, and conservation. The reasoner is used to optimize the performance of the semantic queries performed in the system by pre-calculating frequent deductions and storing them physically in the network. The reasoner ensures that these deductions are updated following changes of the primary data.

The query system we have designed and describe in this thesis is part of the "Integrated Viewer and Browser"(IVB) component that is running on the RI, and which we are implementing together with partners of the project, in particular ISTI – CNR[30] in Pisa. The "Query Formulation Interface (QFI)" is the part of the component for formulating the queries against the metadata of the repository. This tool is aimed at end-users as we defined them in the introduction of chapter 4.

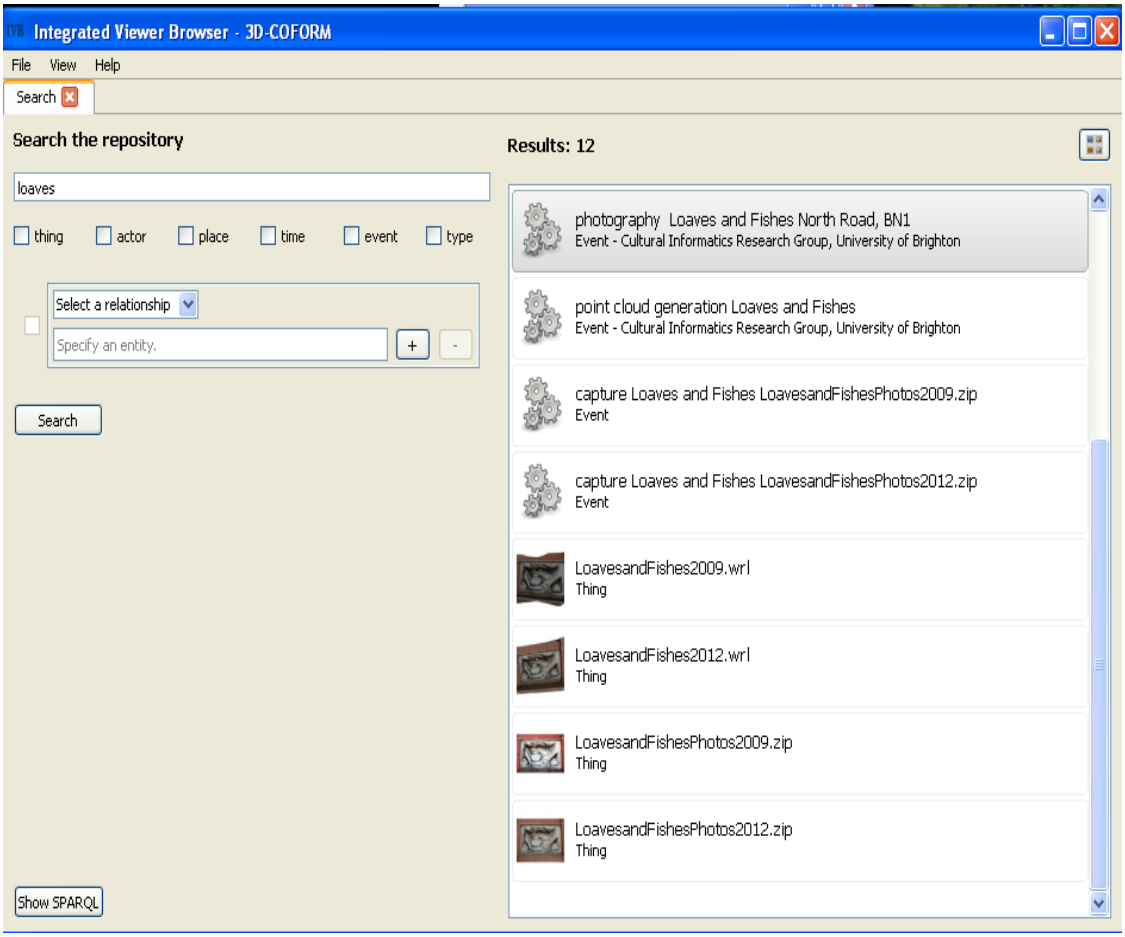

Figure 19: Free text search on QFI

This interface provides both free text search and search via the proposed FCs-FRs model. A user formulating a query in the system may first type in a keyword. A

[30] http://www.isti.cnr.it/

full-text search into all literals returns the associated nodes in the browser, together with minimal metadata and icons, as shown in Figure 19. Each node is marked by the FC it is an instance of. The free text query is based on the fact that in addition to the URIs (Uniform Resource Identifiers), all RDF nodes in the RI are assigned with textual (non-unique) labels with names or titles. This is becoming a good practice in RDF databases. Some also have descriptions in the form of an rdf:literal.

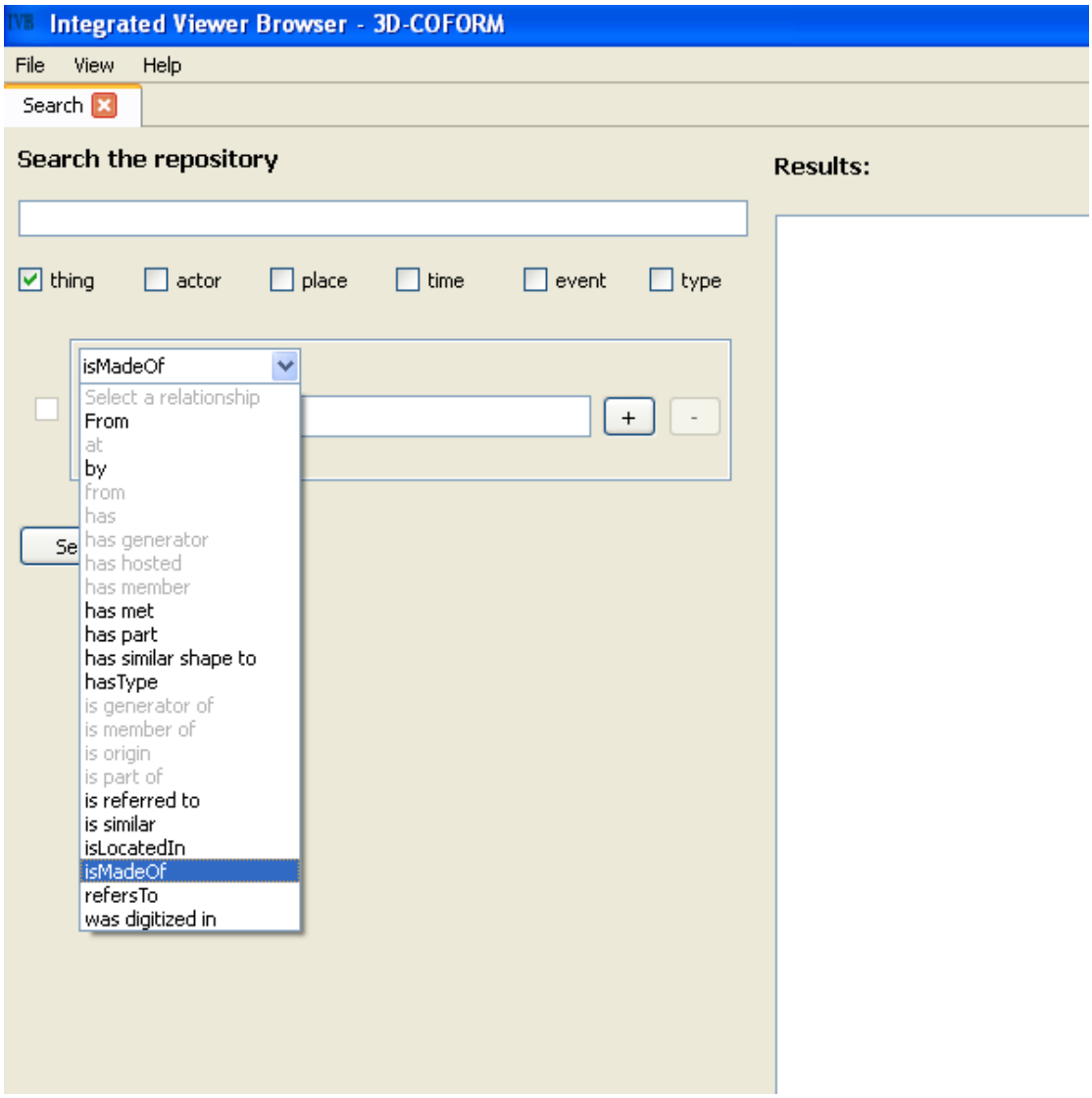

Figure 20: FRs for category Thing

The more precise querying method that the QFI provides is built upon the FCs and FRs. Like the design of the model proposes, a user must first "select" (in the sense of the Structured Query Language (SQL) "Select" statement) the FC from which the question should return instances. Then the user must compose a sort of "Where Clause." This consists of a flat list of properties (FRs) with the selected FC as

domain and with range values combined by AND. For each FC in domain a specific list of FRs are available, which are shown to the user in the dropdown list, like in Figure 20.

Figure 21 shows an example of querying. Here the user wants to search for "Things that are made of stone". As we see in the figure, he has checked the Thing FC for domain selection. Then he has selected from the available FR list with domain Thing, the "is made of" FR specialization. Since "Thing is made of x" expects range parameters that belong only to the Concept FC the typing look ahead mechanism proposes only instances of that FC. So, the user selects the desired stone type and performs the search.

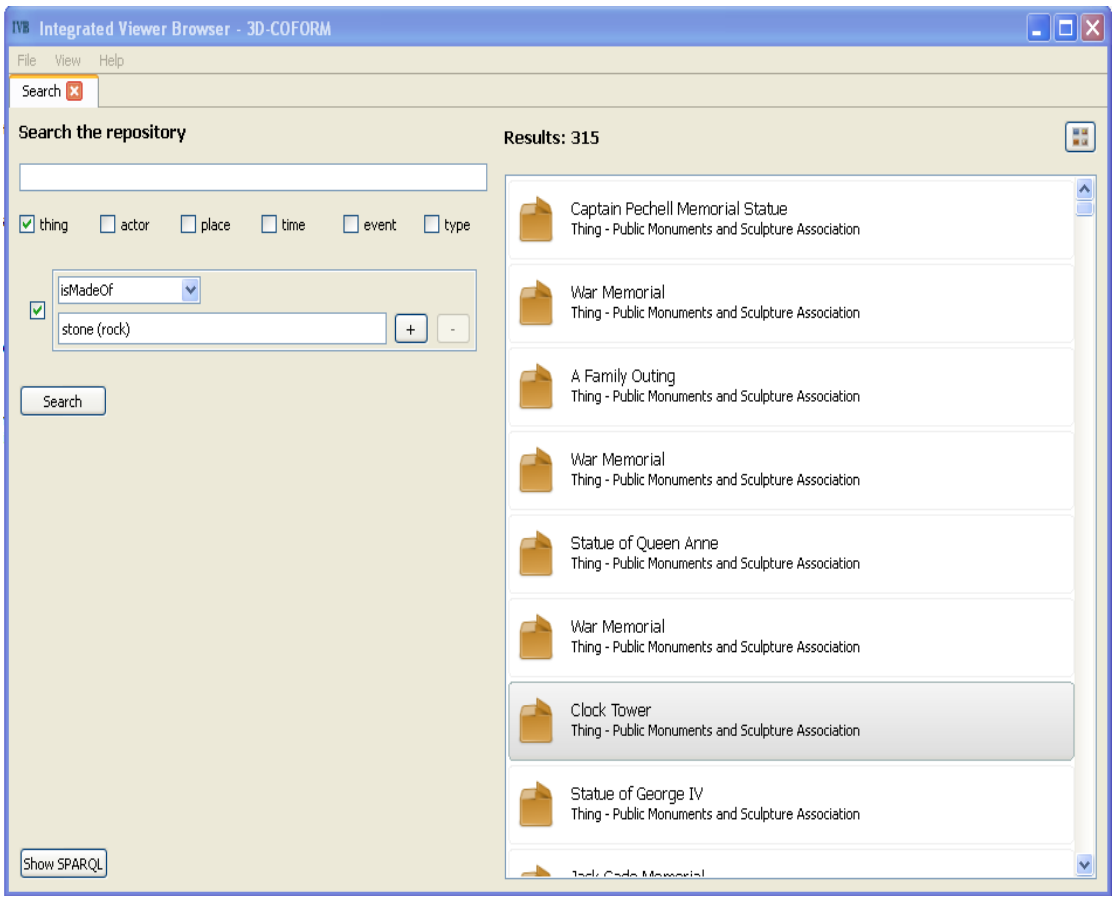

Figure 21: Results to the query

If we browse in the first result, the 'Captain Pechell Memorial Statue', we get information on this statue which is displayed in Figure 22. In this information however we see that the statue has not been assigned any 'stone' type. So, how was it included in the result list?

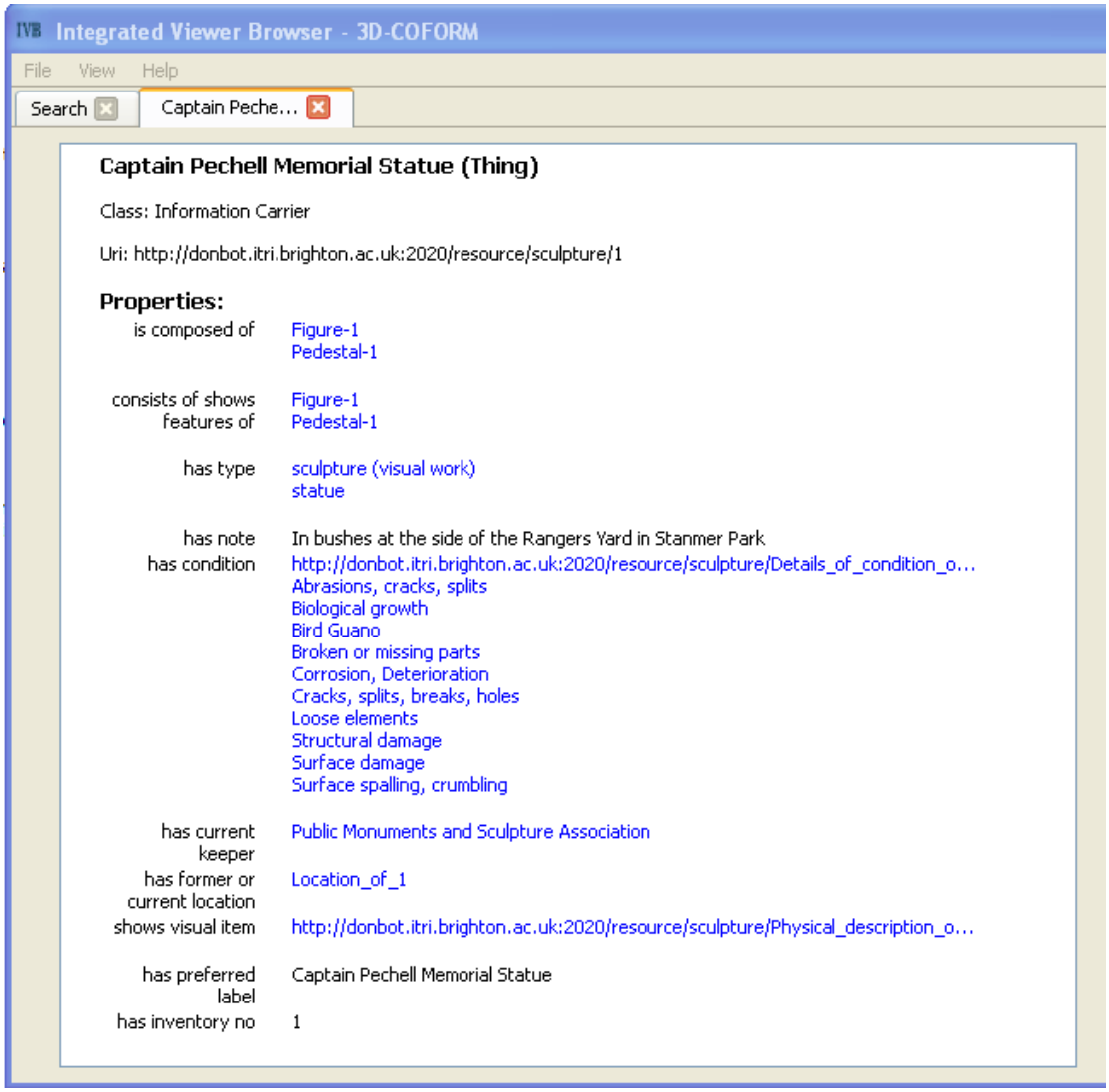

Figure 22: Browse to the first query result (Pechell statue)

The answer is found if we browse one level deeper, at the components of the statue. Figure 23 shows the information of the 'Figure-1', which is part of the Pechell statue. There we can see that this component has type stone and we infer that this is also the reason why the Pechel statue was returned as a relevant result. Since it has a part of stone, it is also considered to be made of stone. So, we have managed to increase recall by property propagation from parts to whole.

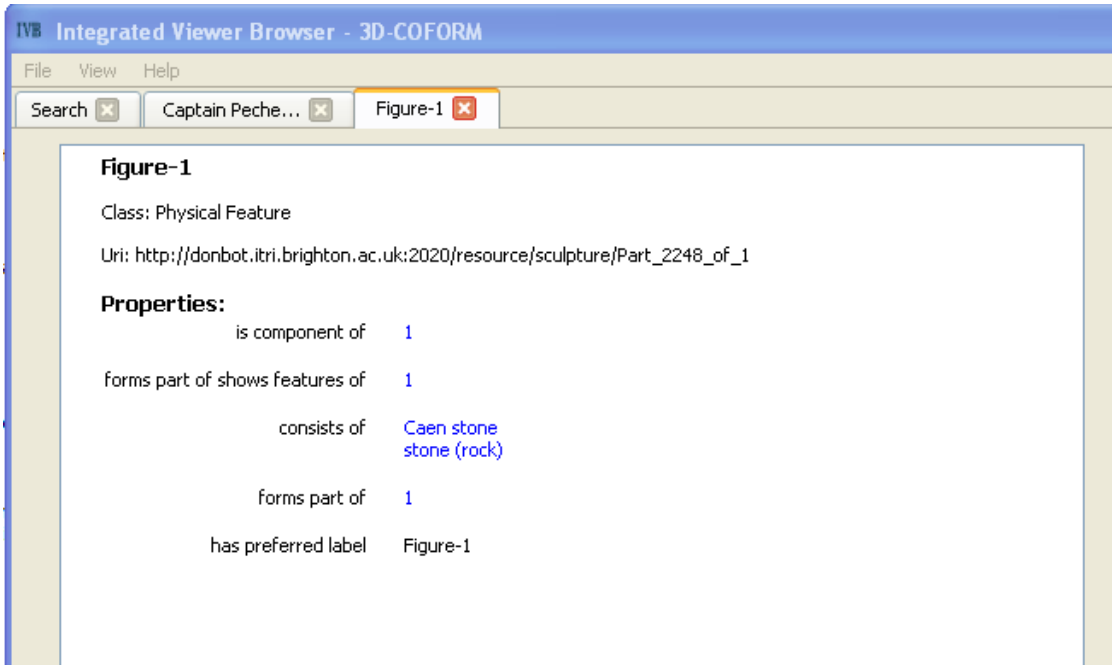

Figure 23: Browse to Figure-1 (Pechell statue's component)

5.2. Research Space Project

ResearchSpace Project[31] is a project aimed at supporting collaborative internet research, information sharing and web applications for the cultural heritage scholarly community. The ResearchSpace environment intends to provide a) Data and digital analysis tools b) Collaboration tools c) Semantic RDF data sources d) Data and digital management tools e) Internet design and authoring tools f) Web Publication. Semantic technology is at the core of the infrastructure because it provides an effective mechanism for research and collaboration across data provided by different organizations and projects. For the querying purposes on metadata structured following the CIDOC-CRM schema, the project has accepted the proposed in this thesis model.

The proposed framework uses the SPARQL language as the tool to perform the queries against the semantic networks. This is done by mapping the Fundamental Relationship that the user has selected into the respective SPARQL statement. Nevertheless, instead of using SPARQL statements the Research Space project translates the FR paths to OWLIM rules. In this way for each FR a set of new inferences are created in the repository and then are queried for.

[31] http://www.researchspace.org/

OWLIM [48] is a semantic repository - a software component for storing and manipulating huge quantities of RDF data. It provides the possibility of reasoning on the metadata, with the rule format and the semantics of its logical formalism. In this way inferences are built and enrich the meaningfulness and richness of the stored data. OWLIM reasoning is preferred to OWL-RL[32] as the first is more expressive allowing also definition of a property by conjunction.

Briefly, a rule definition in OWLIM consists of premises and corollaries, as we have mentioned again in chapter 2, that are RDF statements defined with subject, predicate and object. Each rule is defined by an Identifier. Thus, OWLIM rules can be used in order to create the rule representation of the Fundamental Relationships. Every Fundamental Relationship corresponds to a path of triples as described in Section 4.1. This path of triples can be used to form the premises of the rule, and the FR can be used to create the corollary of it. In this way, the big SPARQL statements that correspond to each FR's path are reduced to just a simple SPARQL statement, querying only the created one-triple corollaries corresponding to each FR. This could dynamically improve the performance of the query in terms of time.

Even though the alternative implementation described above can replace the proposed SPARQL statements implementation, yet again there are some factors to be taken into account. Firstly, regarding the OWLIM case, the generation of about 100 and more rules in the database, would also increase the load of the database, especially if we are talking for already a massive database. Furthermore, care should be taken for the maintenance of the method. If the interpretation of a FR is changed or if a change is performed on a path that is commonly used in many FRs that would also have impact to the implemented rules and an efficient solution for the update purposes must be provided.

5.3. Evaluation

5.3.1. Research questions

In this section we display a set of research questions and we prove that they can be mapped to our model in order to create semantic, associative queries. These

[32] http://www.w3.org/2001/sw/wiki/OWLRL

research questions are extracted from three examples of Egyptian artifacts and monuments that demonstrate the different approaches to reconstruct their original form provided in the "Report on different archeological approaches to reconstructing ancient Egyptian monuments and artifacts" created by an Mahmoud Ibrahim, an Egyptologist from the Center for Documentation of Cultural and Natural Heritage (CULTNAT)[33]. They have been analyzed in terms of queries to a repository that can support the answer of such questions, assuming that it consists of a hypothetical semantic network containing rich cultural information.

Attached to every query there is a graphical representation[34] of the queries with terms from the CIDOC-CRM that adopts the following notation: Boxes represent classes, the upper part of which is the name of the CIDOC CRM class and the lower part is the name of an instance of that class. Arrows denote properties and the name of the CIDOC-CRM property is printed over the arrow. Query parameters include terms, numbers, dates, strings. Variables are represented with the letters X, Y, Z, U, V, W and denote any node fitting the respective path. The result of the query is denoted with "?".

After each graph, we present how each query is mapped to our proposed model, proving its adequacy to represent research questions of the domain.

Query 1: Find the material of the **Monument A**

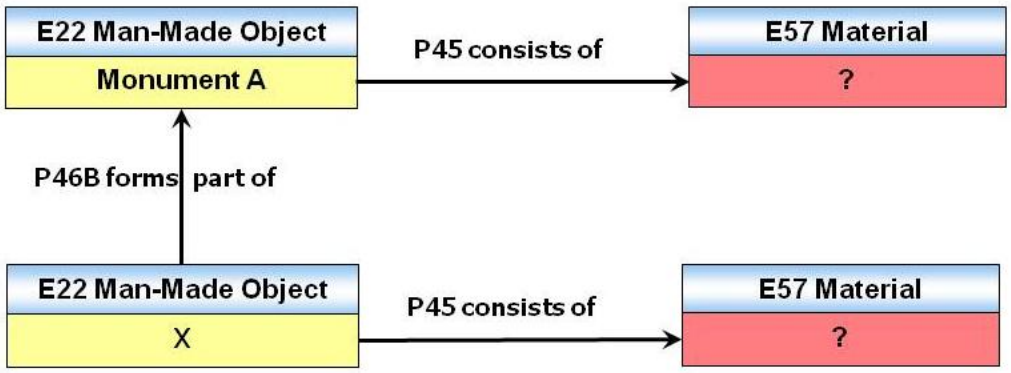

Figure 24: Query 1 graph

[33] http://www.cultnat.org/Cultnat.aspx
[34] The graphical representations are created by the archaeologist Lida Mpaxrami .

Figure 24 shows the respective query graph. This query searches for the material of the Thing Monument A. The material of the Monument A includes its explicitly stated materials but also the materials of its parts. This query could be answered by the "Concept is type of Thing" FR creating the query *Concept is type of Monument A*. The implementation of the FR with CIDOC-CRM terms can be seen in the following path:

E55.Type --(P127B.has_narrower_term)[0,n] -> **E55.Type:**
{E55.Type --P2B.is_type_of->**E70.Thing:**
{E70.Thing -- (F4B.is_component_of) [0,n] -> **E70.Thing**
}
OR
E57.Material – F45B.is_incorporated_in-> **E70.Thing:**
{E70.Thing -- (F4B.is_component_of) [0,n] -> **E70.Thing**
}
OR
E57.Material –P68B.use_foreseen_by->**E29.Design_or_Procedure:**
{ E29.Design_or_Procedure – P33B.used_by->**E7.Activity:**
{ E7.Activity -- P92F.brought_into_existence-> **E70.Thing:**
{E70.Thing -- (F4B.is_component_of) [0,n] ->
E70.Thing
}
}
}
OR
E57.Material --P126B.was_employed_in-> **E11.Modification:**
{ E7.Activity -- P92F.brought_into_existence-> **E70.Thing:**
{E70.Thing -- (F4B.is_component_of) [0,n] -> **E70.Thing**
}
}
OR
E57.Material – P2B.is_type_of -> **E3.Condition_State:**
{ E3.Condition_State --P44B.condition_of **-> E70.Thing:**
{E70.Thing -- (F4B.is_component_of)[0,n] -> **E70.Thing**
}
}
}

Query 2: Find statues that portray **Seti I**

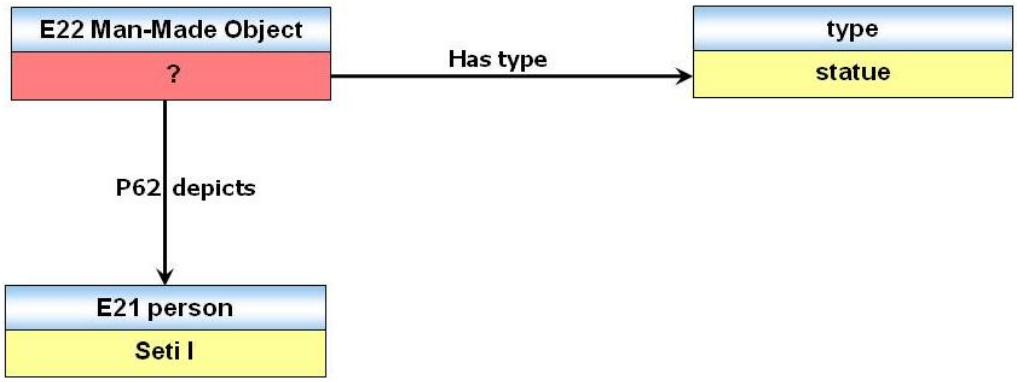

Figure 25: Query 2 graph

Figure 25 shows the respective query graph. This query searches for the statues which have as subject the person Seti I. This is a case of combined query. Here the domain variable FC (=Thing) has two constraint parameters. Firstly we demand that the thing has a certain type and then that it has a certain subject. The two FRs that can be used in conjunction are the "Thing has type Concept" and "Thing is about or refers to Actor", creating the query: *Thing has type statue* and *Thing is about or refers to Seti I*. The implementation of the FRs with CIDOC-CRM terms can be found in Appendix A.

Query 3: Find the things that are attached to the statue of Seti I.

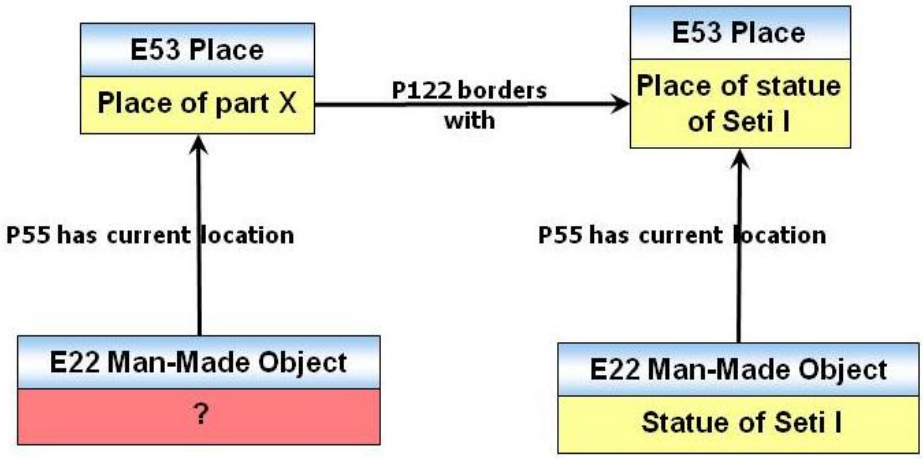

Figure 26: Query 3 graph

Figure 26 shows the respective query graph. This query searches for things that border with Statue of Seti I, or as the graph shows with the place that the statue is located in (and we assume it is not identified in the query). This is a case of multi-level query which creates chains of FRs, providing constraints for the range variable. So, we search for Things located in a Place that borders with a Place that hosts the statue of Seti I. In this case we use the specialization "Thing is/was located in Place" of the FR "Thing from Place" and the reverse of it "Place is location of Thing". This query using only the aforementioned specializations would be formed as in Table 2 (**bold** indicates the querying (domain) variable, *italics* indicates the specified parameters, **bold** red indicates the range variable that becomes domain for the next set of FRs, simple text indicates the FRs):

Thing	Is/was located in	**Place**	Borders/overlaps with	**Place**	is location of	*Statue of Seti I*

Table 2: FRs for Query 3

The implementation of the FRs with CIDOC-CRM terms can be found in Appendix A.

Query 4: Find funerary masks found in the **tomp of Psusennes I**

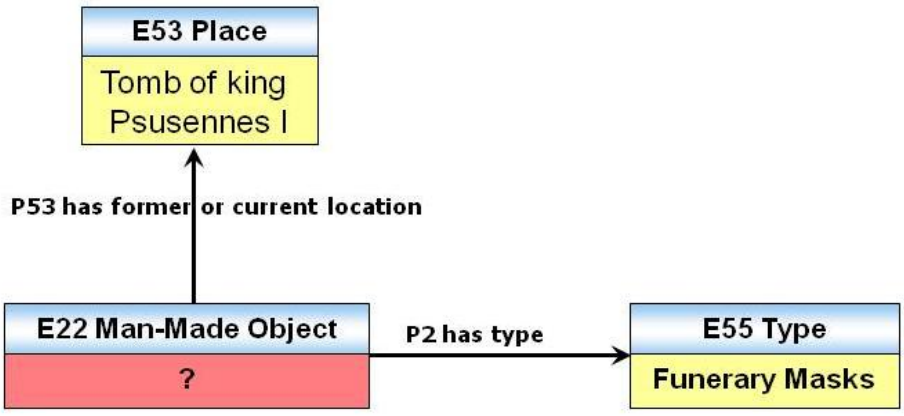

Figure 27: Query 4 graph

Figure 27 shows the respective query graph. This query searches for any funerary masks found in the tomb of Psusennes I. This is a case of combined query.

Here the domain variable FC (=Thing) has two constraint parameters. Firstly we demand that the thing has a certain type and then that it is found in a certain place. The two FRs that can be used in conjunction are the "Thing has type Concept" and "Thing from Place", creating the query: *Thing has type funerary mask* and *Thing from tomb of Psusennes I*. The implementation of the FRs with CIDOC-CRM terms can be found in Appendix A.

Query 5: Find images representing royal statues that were created during excavation events

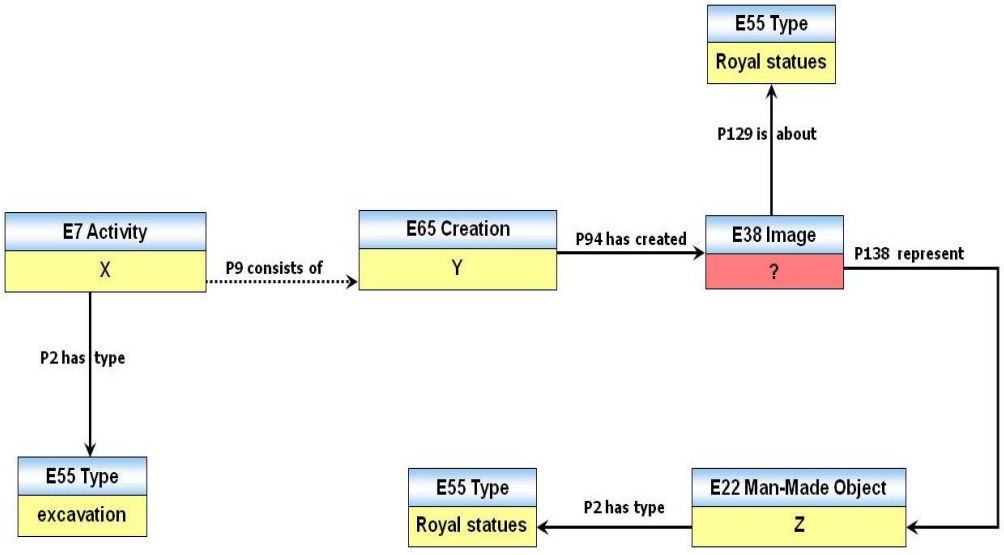

Figure 28: Query 5 graph

Figure 28 shows the respective query graph. This query searches for images that have as subject Things of type royal statues and that were created during an excavation event or most probably in a sub-event of an excavation event. This is a case of combined query which also uses as range the special case of "Thing-Concepts" and "Event-Concepts". Here the domain variable FC (=Thing) has two constraint parameters. Firstly we demand that the thing has a certain subject and then that it was created in a specific event. The two FRs that can be used in conjunction are the "Thing is about or refers to Thing" and the specialization "Thing crated in Event" of the FR "Thing from Event", creating the query: *Thing is about or refers to royal statues* and *Thing created in excavation*. The implementation of the FRs with CIDOC-CRM terms can be found in Appendix A.

Query 6: Find all the stones from Building A that have a previous use and are dated between 1971 - 1928 BC.

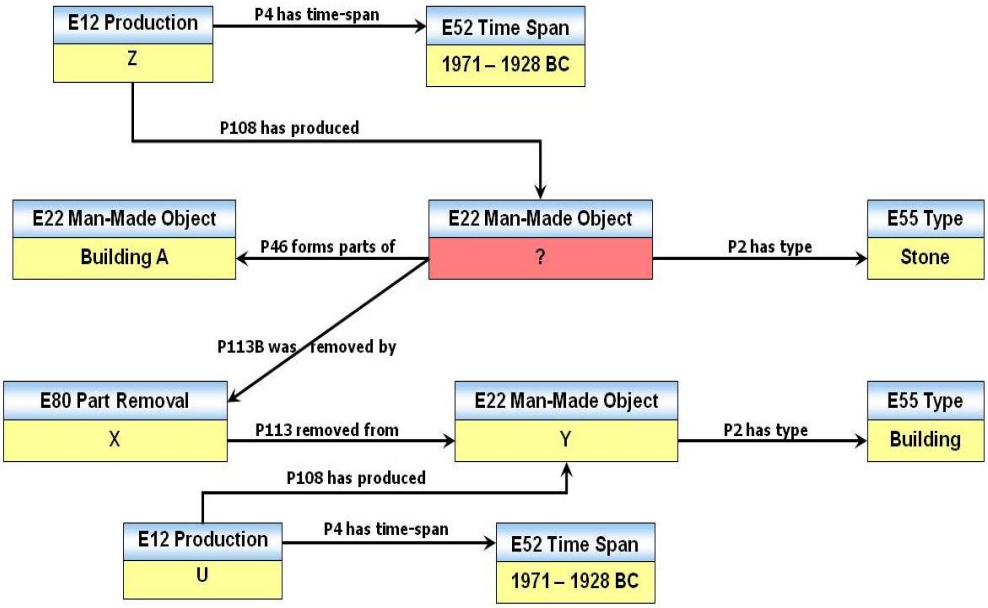

Figure 29: Query 6 graph

Figure 29 shows the respective query graph. This is a complex query that searches for things of type stone, that are part of a Building A and that were produced during 1971-1928 BC and were removed from a thing that has type Building and was produced during 1971-1928 BC. This is a case of combined and multi-level query which creates chains of FRs, providing constraints for the range variables. To explain it more, here the domain variable FC (=Thing) has 4 constraint parameters. Firstly we demand that the thing has a certain Type, then that it was created in a certain Time, then that it is part of another Thing and then that it was from another Thing. This Thing is of a certain Type and was created in a certain Time. This query using only FRs would be formed as in Table 3 (**bold** indicates the querying (domain) variable, *italics* indicates the specified parameters, **bold** red indicates the range variable that becomes domain for the next set of FRs, simple text indicates the FRs):

Thing	has type from from from	*Stone* *1971-1928 BC* *Building A* **Thing**	from has type	*1971-1928 BC* *building*

Table 3: FRs for Query 6

The implementation of the FRs with CIDOC-CRM terms can be found in Appendix A.

Query 7: Find the type themes of stone things.

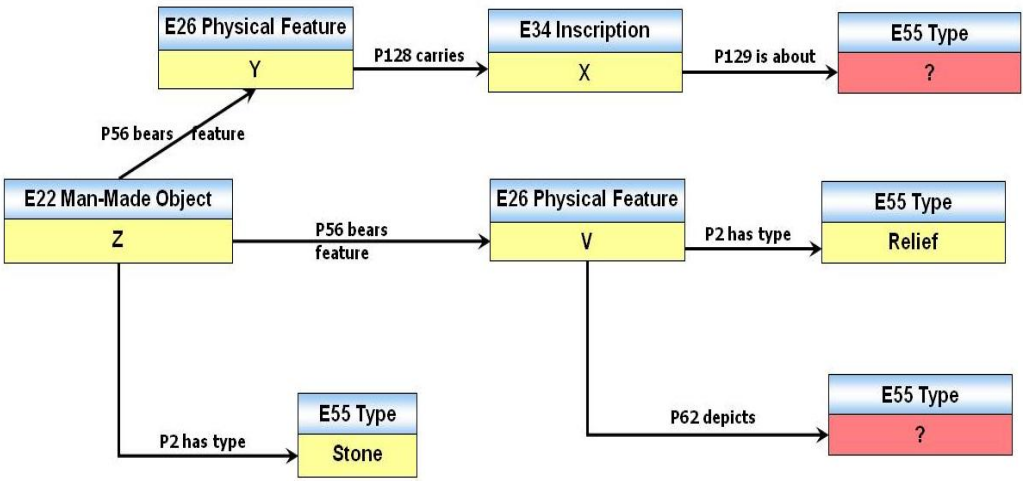

Figure 30: Query 7 graph

Figure 30 shows the respective query graph. This query searches for the concepts that are referred to by things of type stone. This is also a case where the range belongs to the special case of "Thing-Concepts". The FR that is used to represent this query is "Concept is referred to by Thing-Concept" creating the query: *Concept is referred to by stone-Things.* The implementation of the FR with CIDOC-CRM terms can be found in Appendix A.

Query 8: Find royal statues that are dated between 1305 – 1290, are intact and exhibit characteristics typical of middle kingdom

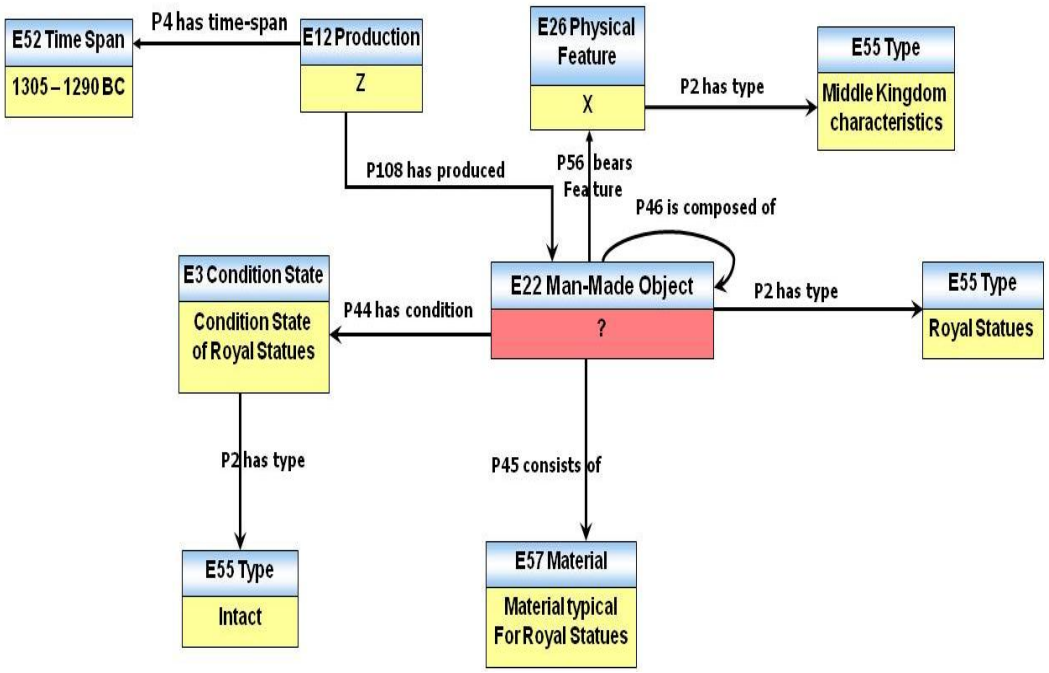

Figure 31: Query 8 graph

Figure 31 shows the respective query graph. This query searches for the royal statues which have 3 kind of types (intact, material typical for royal statues, middle kingdom characteristics) and are from 1305-1290 BC. This is a case of combined query. Here the domain variable FC (=Thing) has five constraint parameters. We demand that the thing has 4 certain types and that it was constructed in a certain time. The FRs that can be used in conjunction are the "Thing has type Concept" and "Thing has met Time", or even better their specializations "Thing is made of Concept" and "Thing created in Time" creating the query: *Thing has type royal statues* and *Thing has type intact* and *Thing is made of material typical for royal statues* and *Thing has type middle kingdom characteristics* and *Thing created in 1305-1290 BC.* The implementation of the FRs with CIDOC-CRM terms can be found in Appendix A.

By the analysis of these research queries that come from independent, objective source we have been able to prove that the proposed model is sufficient and capable of creating associative queries of the kind that archeologists and scientists from the field usually ask.

5.3.2. Test queries

To measure the performance of the system in terms of recall and precision (emphasizing in recall), we have run two set of queries. The queries were performed

on the metadata stored in the 3D-COFORM repository (0.5M triples) and which are built by partners of the project in order to describe digitization processes mainly on objects of cultural interest but also research information and other information on people, places, events and other things involved in the processes. By these tests, we did not only measure the recall and precision rates but also observed the behavior of long SPARQL statements on the Sesame repository, which we also report in this section.

Given that we don't know all the cases described in the metadata of the repository, we can only study some cases and collect the relevant objects according to specific queries. For this purpose we have created two different sets of queries based on two different cases.

The first set of queries is performed based on the most complex case we could demonstrate so far: the one referring to the "Ivory Panel A.15-1955," an object from the Victoria and Albert Museum that depicts the "Ascension" and also "Jesus Christ". The object was digitized twice, and the digital outcomes were further processed in order to produce the final object, which comprised dozens of processing steps and intermediate files. The queries that we performed for this case, were examples of the following FRs:

1. Thing is about or refers to Actor
2. Thing has type Concept
3. Thing has met Actor
4. Thing from Event
5. Thing by Actor
6. Event by Actor
7. Actor from Place

The actual queries, along with the number of relevant (expected) instances, the number of returned by the method objects and the number of the relevant returned by the method objects are given in Table 4.

Query	# of relevant objects	# of returned objects	# of relevant returned objects
1.Thing is about Jesus Christ	17	17	17
2.Thing has type tiff	8	19	8
3.Thing has met VAM	17	24	16
4.Thing from Minidome acquisition Event of Ivory Panel	5	10	5
5.Thing created in Minidome acquisition Event of Ivory Panel	5	5	5
6.Thing by Vam	17	21	17
7.Event by Vam	20	20	20
8.Actor from London	2	2	2

Table 4: Queries and relevant objects

Table 5 shows the values for the recall and precision rates for each query. The rates are calculated by the next equations:

$$Recall = \frac{E \cap \Sigma}{\Sigma}, \qquad Precision = \frac{E \cap \Sigma}{E}$$

where: E =Number of retrieved objects

and

Σ = Number of relevant objects

Query	Recall	Precision
1.Thing is about Jesus Christ	1	1
2.Thing has type tiff	1	0.5
3.Thing has met VAM	0.94	0.67
4.Thing from Minidome acquisition Event of Ivory Panel	1	0.5
5.Thing created in Minidome acquisition Event of Ivory Panel	1	1
6.Thing by Vam	1	0.81
7.Event by Vam	1	1
8.Actor from London	1	1

Table 5: Precision and Recall Rates per Query

If we calculate the average recall and the average precision we have that our method has

$$\overline{Recall} = \frac{7 * 1 + 0.94}{8} = 0.9925, \qquad \overline{Precision} = \frac{6.48}{8} = 0.81$$

So we can see that our method indeed demonstrates high average recall rate and relatively high average precision rate. This is achieved due to useful inferences included in the queries, and to the fact that we also included instances that have a high probability of belonging to the result set. This is something that harmonizes with our design, since it was meant to be recall oriented. Precision is lowered due to the generality of the FRs but also due to the fact that there are also more objects existing in the database with the desired properties, even though we consider them to be irrelevant for this case. For example take the Query 4: "Thing from Minidome acquisition Event of Ivory Panel". This is a generic question that would return all things *participating* in the event, even though the user actually wanted to retrieve things that were *created* in the event. For this reason, the precision rate is relatively low. The next query, Query5:"Thing created in Minidome acquisition Event of Ivory Panel" is a specialization of Query4; this query shows 100% precision. Thus, a possible way to increase precision if it falls beyond expectations is to use

specializations. Another alternative is to further *constrain* the query as described in the next set of test queries.

The aforementioned tests retained to individual usage of each FR, thus one FR was used to construct one query. In this part, we will demonstrate a test that regards the combined usage of FRs in order to create a more specific query.

For this reason we have created the query: "Thing from Brighton and has type sculpture(visual work) and is made of stone(rock)", step by step. For this query we assume as relevant only one object, the "Loaves and Fishes" sculpture and we want to observe how precision is reacting to the additional constraints. So we obtain the Table 6 that gather the results (relevant objects=1, retrieved relevant objects=1 for all queries):

Query	**Retrieved objects#**	**Precision**	**Recall**
Thing from Brighton	221	0.0045	1
Thing from Brighton and has type sculpture(visual work)	34	0.029	1
Thing from Brighton and has type sculpture(visual work) and is made of stone(rock)	18	0.05	1

Table 6: Combined query results

Asking combined queries like the one above, performed really well in terms of precision improvement, but up to a certain level. Adding more and more FRs to the query actually means adding more and more conjunctions in the SPARQL query statement, making it also very long. But Sesame Repository, has a limit for the length

of SPARQLS it can process. When the limit is reached, it responds with a memory exception error.

We observed that very long SPARQLs are created for FRs that correspond to very long paths, like the "Thing from Place" FR. So when the query contains at least one such a long FR, no more than one or two FRs, thus constraints, can be added. The created SPARQL reaches the limit of the Sesame Repository. We tried to upgrade the system memory to 8GB but still the problem remained.

The solution to this problem is to include even more inference rules to the database. These rules are derived from the FCs-FRs model and specifically from the paths that constitute long FRs. Every such path can be split into several sub-paths, which then can form new implicit relationships to the database. Using then these implicit relationships, we end with much simpler paths, and at querying time with shorter SPARQLs.

Take for example the FR: Thing from Place. This relationship unites the sub-paths: Thing is located in Place, Thing is created in Place, Thing is created by Actor with residence Place, Thing is created by Actor born/formed in Place. All these sub-paths can be included into the repository as rule premises (see section 4.2.3.6), constructing the consequences named after the sub-paths' name. These consequences alone then can be used to create the path for the FR:Thing from Place. In this way the corresponding SPARQL is dramatically shrunk. This consequence enables the combination of multiple FRs into one single SPARQL.

CHAPTER 6

Conclusions and Further Work

In this work we propose and implement a new method for querying semantic networks: For formulating queries, the user is presented a small list of configurable fundamental relationships and relevant specializations, easy to comprehend and memorize. These fundamental relationships abstract by rich deductions from an underlying semantic network of much more specialized metadata comprising mainly explicit event descriptions. In this way they simulate to the user a much simpler semantic network, which covers as many generic questions as possible with a high recall. Following the selection of a value for a query parameter, the user is presented with at most a dozen relationships, fewer than Dublin Core [6]. The specializations of the FRs allow for systematically increasing the precision of queries on demand, down to the level of detail of the underlying network ontology. Being so compact, our proposed model reliefs the user from the necessity to know by heart complex schemata or semantic query languages, providing an easy way to perform queries. On the other hand, we maintain the rich underlying metadata schema allowing for reasoning and information integration. Even though the design is mostly directed to the cultural heritage world, it can be customized to different scientific domains as well.

To realize this proposal, we have built a context-free language, the "paths' language" designed to be used mainly by non-expert users and have provided its syntax and semantics. Built upon this language the "Fundamental Relationships Configuration tool" aims at facilitating the implementation and maintenance of the abstract query model. The provided functionality includes among others the validation of paths build with the proposed language, the translation of the paths to SPARQL, the detection of FR specializations etc. Especially for the domain of cultural heritage we have implemented this model using as base the CIDOC-CRM schema and extensions of it for digital objects.

This proposal has been implemented in the context of the 3D-COFORM Project[35] while also the Research Space Project[36] has accepted and is in parallel working on a different implementation approach. The inclusion of the proposed querying method in two big European projects proves, up to a degree, its practical feasibility. In order to ensure its capability to express associative scientific queries we have worked on consolidating and refining the proposed model with respect to real user questions including practical 3D data management and scholarly queries from an independent source. Also, we have tested a set of the most profound Fundamental Relationships against real metadata and the results have been encouraging, showing high recall rates and adjustable precision rates.

Still, our method of selecting the FCs and FRs is mostly intuitive and further work can be done in the direction of standardizing the model. Moreover, since the model is basically implemented in the context of one still ongoing project there is space for further testing and verification. In our case, the model is implemented in two metadata schema-the CIDOC-CRM and the CIDOC-CRMdig. Testing and verification could be done also in more metadata repositories using the same or different schemata, of the same and/or different discourse.

The field allows also for more scientific work to be conducted. First of all it would be interesting to compare the proposed SPARQL implementation versus the OWLIM rules implementation of the Research Space Project, in terms of time and memory allocation efficiency. In another direction to further improve the precision, alterations can be done to the paths' language in order to make it more expressive, for example by enabling the possibility of including negation (filters) in crucial positions of the FRs. Till now, the model cannot use an FR to define other FRs, unless the FR is defined as schema property, e.g. through OWLIM rules. As further work, the FR configuration tool could include the option of including an FR into the definition of another FR, instead of having to re-write the path for this FR into the new definition. One extra enhancement on the FR configuration tool would be a graphical path formulation interface to facilitate the user in their process of building the paths for the FRs. As far as the application of the model in querying tools is concerned, along with

[35] http://www.3d-coform.eu/, duration 2008-2012
[36] http://www.researchspace.org/, deployment duration 2011/2012

the answers, a description on their provenance could also be displayed, in other words an explanation why each answer has been included in the result set.

CHAPTER 7

Bibliography

[1] Antoniou G. and F. van Harmelen. *A Semantic Web Primer*, Cambridge MA, MIT Press, 2004.

[2] Klyne, G. and Carroll, J. Resource Description Framework (RDF): Concepts and Abstract Syntax - W3C Recommendation. (http://www.w3.org/TR/rdfconcepts/).

[3] Signore O. The Semantic Web and cultural heritage: ontologies and technologies help in accessing Museum information. In*: Information Technology for the Virtual Museum*, 2006.

[4] Coburn E. and Light R. and McKenna R. and Stein R. and Vitzthum A. LIDO - Lightweight Information Describing Objects Version 1.0, ICOM-CIDOC, 2010.

[5] Fernandez M. and V. Lopez and M. Sabou and V. Uren and D. Vallet and E. Motta and P. Castells. Semantic Search meets the Web. In: *The IEEE International Conference on Semantic Computing*, pages 253–260, 2008.

[6] Hilman, D. Using Dublin Core - The Elements, 2006 (http://dublincore.org/documents/usageguide/elements.shtml.)

[7] VRA Core 4.0 Element Description, 2007 (http://www.loc.gov/standards/vracore/VRA_Core4_Element_Description.pdf).

[8] SPARQL Query Language for RDF, 2008 (http://www.w3.org/TR/rdf-sparql-query/).

[9] Theodoridou M. and Y. Tzitzikas and M. Doerr and Y. Marketakis and V. Melessanakis. Modeling and Querying Provenance by Extending CIDOC CR.M. Distributed and Parallel Databases. In: *Distributed and Parallel Databases*, (27)2, 2010.

[10] Moreau L. and J. Freire and J. Myers and J. Futrelle and P. Paulson. *The Open Provenance Model*. University of Southampton. 2007 (http://openprovenance.org/).

[11] Tzompanaki K. and M. Doerr. A New Framework For Querying Semantic Networks. In: *Proceedings of Museums and the Web 2012: the international conference for culture and heritage on-line*. San Diego, 2012.

[12] Tzompanaki K. and M. Doerr. A New Framework For Querying Semantic Networks.In: *CIDOC 2012 Conference*, Helsinki, 2012.

[13] Bailey J. and F. Bry and T.Furche and Schaffert S. Web and Semantic Web Query Languages: A Survey, In: Reasoning Web, First International Summer School 2005, Springer-Verlag, LNCS 3564, 2005.

[14] Prud'hommeaux E. and A. Seaborne. SPARQL Query Language for RDF, 2008 (http://www.w3.org/TR/rdf-sparql-query/).

[15] McGuinness D. and F. van Harmelen. OWL Web Ontology Language Overview, 2004 (http://www.w3.org/TR/owl-features/).

[16] W3C OWL Working Group. OWL 2 Web Ontology Language Overview, 2009 (http://www.w3.org/TR/owl2-overview/).

[17] T. R. Gruber. A translation approach to portable ontologies. In: *Knowledge Acquisition*, 5(2):199-220, 1993.

[18] Klein D. and F. Fensel and Harmelen and I. Horrocks. The relation between ontologies and XML Schemata. In: *Proceedings of the {ECAI}'00 workshop on applications of ontologies and problem-solving methods*, Berlin, 2000.

[19] Rosch E. *Principles of Categorization*. University of California, Berkeley. First publised in: Rosch, Eleanor and Lloyd, Barbara B. (eds), Cognition and categorization 27-48, 1978.

[20] Ribeiro-Neto B. and R. Baeza-Yates. *Modern Information Retrieval*. Addison Wesley Longman Publishing Co. Inc, 1999.

[21] Auer S. and C. Bizer and G. Kobilarov and J. Lehmann and R. Cyganiak and Z. Ives. DBpedia: a nucleus for a Web of open data. In: *Proceedings of 6th nternational Semantic Web Conference, 2nd Asian Semantic Web Conference*, 722–735, 2007.

[22] Manolis N. and Y. Tzitzikas. Interactive Exploration of Fuzzy RDF Knowledge Bases. In: *Procs of ESWC'11*, 2011.

[23] Russell, Alistair, Smart, Paul R., Braines, Dave and Shadbolt, Nigel R. NITELIGHT: A Graphical Tool for Semantic Query Construction. In: *Semantic Web User Interaction Workshop (SWUI 2008)*, Florence, Italy, 2008.

[24] Ding L. and T. Finin and A. Joshi and R. Pan and R.S. Cost and Y. Peng and P. Reddivari and V. Doshi and J. Sachs. Swoogle: a search and metadata engine for the semantic Web. In: *The thirteenth ACM international conference on Information and knowledge management*. New York, ACM. 652–659, 2004.

[25] Lopez V. and E. Motta and V. Uren. PowerAqua: Fishing the Semantic Web. In: *The Semantic Web: research and applications*, Volume 4011, 393–410, 2006 (http://technologies.kmi.open.ac.uk/poweraqua/).

[26] Lei Y. and V. Uren and E. Motta. SemSearch: A Search Engine for the Semantic Web. In: *Managing Knowledge in a World of Networks*, Springer LNCS, Volume 4248, 238–245, 2006.

[27] Sacco, G. M. and Y. Tzitzikas. *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*. The Information Retrieval Series, Vol. 25, Springer, 2009.

[28] Doerr M. and D. Iorizzo. The dream of a global knowledge network-A new approach. In: *ACM, Journal on Computing and Cultural Heritage*, 1(1), 1–23, 2008.

[29] Hyvoenen E.and T. Ruotsalo and T. Haeggstroem and M. Salminen and M. Junnila and M. Virkkilae and M. Haaramo and E. Maekelae and T. Kauppinen and K. Viljanen. CultureSampo-Finnish Culture on the SemanticWeb: The Vision and First Results. In: *12th Finnish Artificial Intelligence Conference STeP*, 2006 (http://www.kulttuurisampo.fi/index.shtml)

[30] Shaw M. and Betwiler L. and Noy N. and Brinkley J. and Suciu D. vSPARQL: A view definition language for the semantic web. In: *Journal of Biomedical Informatics*, Volume 44, 102–117, 2011.

[31] Magkanaraki A. and Tannen V. and Christophides V. and Plexousakis D. Viewing the semantic web through RVL lenses. In: *Proceedings of the International Semantic Web Conference*, 96–112, 2003.

[32] Karvounarakis G. and V. Christophides and D. Plexousakis. RQL: A Declarative Query Language for RDF. In: *Eleventh International World Wide Web Conference (WWW)*, Hawaii, 2002.

[33] Lagoze C. and J. Hunter. The ABC Ontology and Model. In: *Journal of Digital Information*, 2001.

[34] Gangemi A. and N. Guarino and C. Masolo and A. Oltramari and L. Schneider. Sweetening Ontologies with DOLCE. In: *EKAW 2002*, 166–178, Spain, 2002.

[35] Basic Formal Ontology. http://www.ifomis.org/bfo.

[36] Doerr M. and S. Gradmann and S. Hennicke and A. Isaac and C. Meghini and H. van de Sompel. The Europeana Data Model (EDM). In: *World Library and Information Congress: 76th IFLA General Conference and Assembly*, 2010 (http://www.europeana.eu/portal/).

[37] Bekiari Ch. and Ch.D. Gritzapi and D. Kalomoirakis. POLEMON: A Federated Database Management System for the Documentation, Management and Promotion of Cultural Heritage. In: *The 26th Conference on Computer Applications in Archaeology*, Barcelona,1998.

[38] Ranganathan S.R. *A Descriptive Account of Colon Classification*, Bangalore: Sarada Ranganathan Endowment for Library Science, 1965.

[39] Bearman D. and J. Perkins. *Standards Framework for the Computer Interchange of Museum Information*. Museum Computer Network, 1993 (http://old.cni.org/pub/CIMI/framework.html).

[40] Lakoff G. *Women, fire, and dangerous things: What categories reveal about the mind*, University of Chicago Press, 1987.

[41] Pustejovsky J. *The generative lexicon*. MIT Press, 1995.

[42] McCarthy L. and Vandervalk B. and Wilkinson M. SPARQL Assist language-neutral query composer. In: *BMC Bioinformatics 2012*, Volume 13, 2012.

[43] Lushan H. and Finin T. and Joshi A. GoRelations: An Intuitive Query System for DBpedia, In: *Proceedings of the Joint International Semantic Technology Conference*, Springer LNCS, 2011.

[44] Doerr M. and K. Tzompanaki and M. Theodoridou and Ch. Georgis and A. Axaridou and S. Havemann. A Repository for 3D Model Production and Interpretation in Culture and Beyond. In: *Proceedings of VAST 2010: 11th International Symposium on Virtual Reality, Archaeology and Cultural Heritage*, 97–104, Paris, 2010.

[45] OWLIM: Semantic Repository for RDF(S) and OWL-SwiftOWLIM/BigOWLIM, versions 2.9÷3.X, 2009 (http://www.askpetko.com/Samples/PrintManuals/OWLIMPrimer.pdf).

[46] Doerr M. The CIDOC CRM – An Ontological Approach to Semantic Interoperability of Metadata. In: *AI Magazine*, 24(3), 75–92, 2003 (http://www.cidoc-crm.org/index.html).

[47] Tzompanaki K. and M. Doerr and M. Theodoridou and S. Havemann. 3D-COFORM: A Large-Scale Digital Production Environment .In: *ERCIM News*, Volume 86, 18-19, 2011.

[48] Kiryakov A. and D. Ognyanov and D. Manov. OWLIM – a pragmatic semantic repository for OWL. In: *Proceedings of the Conference on Web Information Systems Engineering (WISE) Workshops*, 182–192, 2005.

[49] O'Neill E.T. FRBR: Functional Requirements for Bibliographic Records, Application of the Entity-Relationship Model to Humphry. In: *Library Resources & Technical Services*, 2002.

[50] Kakali C. and I. Lourdi and T. Stasinopoulou and L. Bountouri and C.Papatheodorou and M. Doerr and M. Gergatsoulis. Integrating Dublin Core metadata for cultural heritage collections using ontologies. In: *2007 Proc. Int'l Conf. on Dublin Core and Metadata Applications*, 2007

[51] Angles R. and C. Gutierrez. *The Expressive Power of SPARQL*. Technical Report TR/DCC-2008-5. Chile, 2008

Appendix A

In this appendix we display the materialization of the Fundamental Categories and Fundamental Relationships model, on the CIDOC-CRM and extensions of it for describing concepts from the digital world, the CIDOC-CRMdig. This appendix can also be found as a technical report under the internet address: http://www.ics.forth.gr/tech-reports/2012/2012.TR429_Intuitive_querying_CIDOC-CRM.pdf
.
Each Fundamental Category and each Fundamental Relationship is mapped to specific entities and grouping of properties of the CIDOC-CRM (http://www.cidoc-crm.org/rdfs/cidoc_crm_v5.0.2_english_label.rdfs) and the extension of it for Digital Things, the CIDOC-CRM Digital (http://www.ics.forth.gr/isl/rdfs/3D-COFORM_CRMdig.rdfs). For disputes that do not concern digital objects the part of the paths that include terms of the CIDOC-CRMdig may be omitted. In the same spirit, if other mappings are used more path-parts may be included in order to complete the needs of the certain case.

Notation
In this proposal we use as mentioned above two schemata, the CIDOC-CRM and the CIDOC-CRM digital. To refer to a class from the CIDOC-CRM we use the letter E (from Entity) and for the properties we use the P (from Property). Respectively we use the letter D to refer to a class from the CIDOC-CRM digital and the letter L to refer to the properties. For extentions to the already existing classes and relationships of the CIDOC-CRM we use the letter C for classes and F for properties. To elaborate this with examples:
E53.Place : a class from the CIDOC CRM
P62F.depicts: the forward link of a property of the CIDOC-CRM
P62B.is_depicted_by: the backward link of a property of the CIDOC-CRM
D1.Digital_Object :a class from the CIDOC-CRM Digital
L1F.digitized: the forward link of a property of the CIDOC-CRM Digital
L11B.was_output_of: the backward link of a property of the CIDOC-CRM Digital
Links that begin with the letter F are used to declare a rule. Since we make use of inferences, we have created in the repository a number of rules that replace certain paths, that either are too long or is used many frequently in the FRs. The notation of links that refer to rules, follow the same concept for forward and backward linking, that is:
F1F.is_derivative_of:the forward property created by the application of a rule
F1B.has_derivative: the backward property created by the application of a rule
For example, the property *F1F.is_derivative_of* hides the following rule applied in the system:

b <crmdig:L21F.used_as_derivation_source> c
b <crmdig:L22F.created_derivative> d
----
d <crm:F1F.is_derivative_of> c

This means that if an event b used as derivation source a digital object c and it created as a derivative the digital object d, then we can deduct that the digital object d is a derivative of c.
In the rest of this section we display a description of the notation we use to create the paths for the FRs. To construct a simple link that forms a triple we use ‘ -- ’ between the subject and the predicate and ‘ -> ’ between the predicate and the object. For example a simple triple is:
Subject -- predicate -> object
We use two types of links: direct and intermediate links. Intermediate are the links that do not connect the domain and range fundamental categories directly with one another, but through other (intermediate) entities. For presentation purposes we mark the intermediate links with blue color and the direct ones with black. We also mark with bold **blue** the intermediate classes. Classes that pose constraints (the range FC) in the path are noted with bold **red**.
For example, the semantic network might contain a direct relationship between a Thing and a Place such as:
E70.Thing-- P62F.depicts->**E53.Place**
And this means that an instance of E70 Thing refers to an instance of E53 Place.
But if the semantic network contains information about a Thing that shows features of another Thing that refers to a Place, then we may in general infer that also the first Thing depicts the Place even

though not explicitly noted. So we include in our query another intermediate relation, “shows features of”, among the first Thing and the Place:
E70.Thing--P130F.shows_features_of-> **E70.Thing:**
{ E70.Thing -> P62F.depicts-> **E53.Place**
}
In the example above an intermediate link is included so it is marked with blue and the intermediate class is marked with bold blue. To explain the rest of the notation consider the following example:
Subject1-- predicate1-> object1
In order to continue the path chain, and use the object1 as a subject for the next triple we add a ‘:’ after the object1 and enclose the next triple (or block of triples) in { } brackets. So this would be:
Subject1-- predicate1-> object1:
{object1 -- predicate2 ->object2
}
To make more illustrative the fact that the next triple is one level deeper we use a tab.
Note that the next triple can continue not only with the object1 itself but also with a sub-class or a super-class of it.
We mark with green links that have as destination a **Type entity** and are used when we refer to a category through its type, eg
E70.Thing-- P62F.depicts-> **E53.Place** [--P2F.has_type -> **E55.Type**]
This is not included in the official grammar, since it is a handling performed in the interface, and it is global for all categories (except for concept).
The use of **OR** operator can be applied either on predicates or on triples level.
When on predicates, we enclose the OR operands united with OR in brackets{ } eg
E73.Information_Object -- {P67F.refers_to OR P129F.is_about} -> **E53.Place**
When on triples again we enclose the block of triples in { } and we unite them with OR as in the following:

E70.Thing -- P130F.shows_features_of -> **E70.Thing:**
{E70.Thing -- P31B.was_modified_by -> **E5.Event**
OR
E70.Thing -- P94B.was_created_by -> **E5.Event**
}
The notation **(*relation*)$^{[0,n]}$** means that the relation within the parenthesis may occur 0 to n times **recursively and always in the same direction**. For example **(P130F.shows_features_of)$^{[0,n]}$** implies the path:
E70.Thing -- P130F.shows_features_of -> E70.Thing -- P130F.shows_features_of -> E70.Thing -- P130F.shows_features_of -> E70.Thing) ..
or just the E70.Thing if we have 0 occurrences of the relationship.
The **(*relation*)$^{[0,n]}$** notation does not imply a change in the traversing direction of a path. For example **(P130F.shows_features_of)$^{[0,n]}$ will never** imply the path**:**
E70.Thing--P130F.shows_features_of->E70.Thing--P130B.features_are_also_found_on-> E70.Thing--P130F.shows_features_of ->E70.Thing .
In the formal grammar we denote this notation as **(*relation*)[0,n]**, but we mean the same thing. This is done because our FR configuration tool handles simple strings, and interprets the $^{[0,n]}$ as [0,n]. Here, in order for the display to be more readable, we have used the $^{[0,n]}$ notion, for the same reasons that we have used different formatting for each part of the path (colors, bolds etc.)

THING

The Fundamental Category Thing is a general class that could be mapped to the CIDOC-CRM class E70.Thing. As such, this category comprises usable discrete, identifiable, instances of E77.Persistent Item that are documented as single units.
However, there are some subclasses that we do not consider to be members of the FC Thing, either because they are better categorized in another FC, or because for the current discourse in the concept of the 3D-COFORM project some E70.Thing sub-classes are of no special interest. So from the general category we exclude the classes E21.Person, E55.Type, E30.Right and E41.Appelation. So, the FC Thing could be mapped to the following:
Thing={E70.Thing NOT E21.Person NOT E55.Type NOT E30.Right NOT E41.Appelation}.
As in the paths' language we don't support negations, these disjunctions are formulated in the query interface. There, when instances of the FC Thing are asked for, there is a filtering at the returned instances for the disjoint types. In the future, we may expand the grammar to include also disjunctions.

Thing-Place

Instances of Place are geometrically identified by global coordinates or absolute reference systems. But practically one may intuitively refer to a Place by physical features that are or once have been located in a Place, such as cities or buildings. This is the reason why we also include physical features in the relationships that include the FC **Place**.

a. refers to or is about

Things often refer to or are about Places basically by their themes. So either they do so by directly referring to the Place or by being the "carrier" of the thing that refers to the Place. Copies of things that refer to or are about places also bear the same property.
Here we could make an expression that describes this combination of the interpretations of Place in order to skip the repetition of same blocks of paths :
Place= {**E26.Physical_Feature** -- P53F.has_former_or_current_location->**E53 Place**
OR
E53.Place
}
In this version we have replaced the two frequent sets of transitive properties, declaring part-whole relationships and "shows features of" relationships with the respective rule
F5F.consists_of_shows_features_of which is also a transitive property.

There properties that define the spatial containment of two Places:

- P89F.falls_within (P89B.contains)

P89F.falls_within is a super-property of *P88B.forms_part_of* . P88 also defines contextual containment relationship between the Places. Here the more general relationship P89 is enough so we use this one in the path.
In CIDOC-CRM properties that define the "refer to" or "is about" relationship are:

- P62F.depicts
- P67F.refers_to

The respective general fundamental relationship "*Thing refers to or is about Place"* is:

E70.Thing --(F5F.consists_of_shows_features_of)$^{[0,n]}$-> **E70.Thing:**
{E24.Physical_Man-Made_Thing -- P62F.depicts **-> E53.Place**:
{E53.Place--(P89B.contains)**)**$^{[0,n]}$ **-> E53.Place** [--P2F.has_type ->
E55.Type]
}

OR
E89.Propositional_Object -- P67F.refers_to-> **E53.Place**:
{E53.Place--(P89B.contains)**)**$^{[0,n]}$ **-> E53.Place** [--P2F.has_type ->
E55.Type]
}

OR
E24.Physical_Man-Made_Thing -- P62F.depicts **-> E26.Physical_Feature:**

{E26.Physical_Feature--P53F.has_former_or_current_location ->
E53.Place:
{E53.Place--(P89B.contains)**[0,n]** -> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
OR
E89.Propositional_Object --P67F.refers_to -> **E26.Physical_Feature:**
{E26.Physical_Feature--P53F.has_former_or_current_location ->
E53.Place:
{E53.Place--(P89B.contains)**[0,n]** -> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries-> **E73.Information_Object:**
{**E73.Information_Object** -- P67F.refers_to -> **E26.Physical_Feature:**
{E26.Physical_Feature--P53F.has_former_or_current_location ->
E53.Place:
{E53.Place--(P89B.contains)**[0,n]** -> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
OR
E73.Information_Object -- P67F.refers_to -> **E53.Place**:
{E53.Place--(P89B.contains)**[0,n]** -> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
OR
D1.Digital_Object -- L11B.was_output_of -> **D7.Digital_Machine_Event:**
{D7.Digital_Machine_Event --(P9B.forms_part_of)[0,n] ->
D2.Digitization_Process :
{D2.Digitization_Process -- L1F.digitized -> **E18.Physical_Thing:**
{E26.Physical_Feature--P53F.has_former_or_current_location ->
E53.Place:
{E53.Place--(P89B.contains)**[0,n]** -> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P62F.depicts ->
E53.Place:
{E53.Place--(P89B.contains)**[0,n]** -> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{ **E73.Information_Object** --P67F.refers_to ->
E53.Place:
{E53.Place--(P89B.contains)**[0,n]** -> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}
}
}

b. is referred to at

There are cases that one is interested in Things that are referred to at a certain Place. Reference implies the presence or creation of a Thing that refers to the Thing(s) in interest, as even speech is a human product, thus a Thing. So, when we talk about something, write about something or in any other way mention something the means we do it is mapped to a Thing that we create.
In this manner we connect the Thing and Place FCs not only directly but also through other Things and through Events.
In this relationship we do not use the "shows features of" property because if someone refers to a Thing, this does not mean that they also refer to any copies of it.
Moreover, we do not include here the Physical Feature notion. Mainly the reference at a Place is done with the P53F.has_former_or_current_location property of an instance of **E18.Physical_Thing** which is a super-property of the E26.Physical_Featuer. Then we also express the "is referred to at" FR through creation events that are directly connected with Places with the property P7F.took_place_at.
The property P8F.took_place_on_or_within cannot be used since it connects an event with a Thing that cannot surely be placed in a Place with no respect to a time period.
In CIDOC-CRM properties that define the "is referred to" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by
-

The respective general fundamental relationship "*Thing is referred to at Place"* is:

E70.Thing -- (F4B.is_component_of)$^{[0,n]}$ -> **E70.Thing:**
{E70.Thing -- P67B.is_referred_to_by -> **E89.Propositional_Object:**
{E89.Propositional_Object -- P94B.was_created_by -> **E65.Creation**:
{E65.Creation --(P9B.forms_part_of)$^{[0,n]}$-> **E5.Event:**
{E65.Creation -- P7F.took_place_at ->**E53.Place:**
{E53.Place --(P89F.falls_within)$^{[0,n]}$-> E53.Place
[--P2F.has_type -> **E55.Type**]
}
}
}
OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{E18.Physical_Thing -- P53F.has_former_or_current_location ->
E53.Place :
{E53.Place --(P89F.falls_within)$^{[0,n]}$-> E53.Place
[--P2F.has_type -> **E55.Type**]
}
}
}
OR
E70.Thing -- P62B.is_depicted_by -> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- P53F.has_former_or_current_location ->
E53.Place:
{E53.Place --(P89F.falls_within)$^{[0,n]}$-> E53.Place
[--P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P108B.was_produced_by -> **E12.Production**:
{ E12.Production --(P9B.forms_part_of)$^{[0,n]}$-> **E5.Event:**
{ E5.Event -- P7F.took_place_at ->**E53.Place**:
{E53.Place --(P89F.falls_within)$^{[0,n]}$-> E53.Place [--
P2F.has_type -> **E55.Type**]
}
}
}
}
}

c. from

This relation returns Things that have as origin a given Place. For the Thing category, we make the general assumption that the Thing as a whole entity and its parts separately are created in the same place.
The relation, "P53F.has_former_or_current_location" is a **deduction** of "P7F.took_place_at". It is obvious that since the Event took place in a certain Place, the included objects also were once located in the same Place.
The "from (history)" relation will group the following relations:

1. The former or current location of the THING or its components.
2. The place where the THING was acquired or found.
3. The place associated (birth, residence) with an Actor who carried out the creation or production of the THING or (residence) who is the keeper or owner of the THING
4. The Place from where the THING was moved and to where the THING was moved. We can both say "Caryatid from the Parthenon" and "Caryatid from the British Museum" (as a place) and it can be differently translated as "Caryatid moved from the Parthenon" and "Caryatid moved to the British Museum".

The link (**E4 Period** --*P8 took place on or within (witnessed)* ->**E19 Physical Object**) describes that an event happens on or within a THING in which case the THING acts as place information but it does not describe any information about the origin or active participation of the THING.
The same applies for the (**E5 Event** -- *P12 occurred in the presence of (was present at)* ->**E77 Persistent Item**) link which describes the participation of a THING in an Event but it does not include any information about the origin or active participation of the THING. So these two properties cannot be included in the "from" relationship.
In CIDOC-CRM properties that define the "from" relationship are:

- P53F.has_former_or_current_location
- P7F.took_place_at
- P74F.has_current_or_former_residence
- P54F.has_current_permanent_location
- P25F.moved with P26F.moved_to and P27F.moved_from
- P24B.changed_ownership_through

The respective general fundamental relationship "*Thing from Place"* is:

E70.Thing -- (F4B.is_component_of)$^{[0,n]}$ -> **E70.Thing:**
{E70.Thing--{P53F.has_former_or_current_location OR
P54F.has_current_permanent_location}-> **E53.Place**:
{E53.Place --(P89F.falls_within**)$^{[0,n]}$-> E53.Place** [--P2F.has_type -> **E55.Type**]
}
OR
E70.Thing -- P92B.was_brought_into_existence_by -> **E63.Beginning_of_Existence** :
{E63.Beginning_of_Existence -- (P9B.forms_part_of) $^{[0,n]}$ -> **E5.Event** :
{ E5.Event -- P7F.took_place_at -> **E53.Place**:
{E53.Place --(P89F.falls_within**)$^{[0,n]}$-> E53.Place** [--
P2F.has_type -> **E55.Type**]
}
OR
E7.Activity --P14F.carried_out_by -> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)$^{[0,n]}$ ->
E39.Actor:
{E39.Actor -- P74F.has_current_or_former_residence ->
E53.Place:
{E53.Place --(P89F.falls_within**)$^{[0,n]}$-> E53.Place**
[--P2F.has_type -> **E55.Type**]
}
OR
E39.Actor --P92B.was_brought_into_existence_by->
E63.Beginning_of_Existence:
{E63.Beginning_of_Existence -- (P9B.forms_part_of)
$^{[0,n]}$ -> **E5.Event:**

{E5.Event -- P7F.took_place_at -> E53.Place:
{E53.Place --(P89F.falls_within)[0,n] ->
E53.Place [--P2F.has_type ->
E55.Type]
}
}
}
}
}
}
}
OR
E19.Physical_Object --P25B.moved_by -> E9.Move :
{ E9.Move -- {P26F.moved_to OR P27F.moved_from} -> E53.Place:
{E53.Place --(P89F.falls_within)[0,n]-> E53.Place [--P2F.has_type ->
E55.Type]
}
}
OR
E19.Physical_Object -- P24B.changed_ownership_through -> E8.Acquisition:
{ E8.Acquisition-- (P9B.forms_part_of)[0,n] -> E5.Event:
{ E5.Event -- P7F.took_place_at -> E53.Place :
{E53.Place --(P89F.falls_within)[0,n]-> E53.Place [--P2F.has_type
-> E55.Type]
}
}
}

OR
E18.Physical_Thing –{P51F.has_former_or_current_owner OR
P49F.has_former_or_current_keeper OR P109B.is_current_or_former_curator_of}->
E39.Actor:
{E39.Actor -- (P107B.is_current_or_former_member_of)[0,n] -> E39.Actor:
{E39.Actor -- P74F.has_current_or_former_residence -> E53.Place:
{E53.Place --(P89F.falls_within)[0,n]-> E53.Place [--P2F.has_type
-> E55.Type]
}
}
}
}

Specializations:

In the fundamental relationship *from (history)* we can also define some sub-fundamental relationships, which contain more restricted information. This information is however commonly asked for by the users and a more specialized query may return better results as far as precision is concerned, based to what they actually want to know about. So we can define the following:

- Created in

This specialized relationship is the most important and most commonly asked, as it is mainly by the creation event of a Thing that we determine its origin. So taken an area or more generally a Place as granted, one may ask for the Things that were created, modified or produced there.

The respective specialized fundamental relationship "*Thing created in Place"* is:

E70.Thing -- (F4B.is_component_of)[0,n] -> E70.Thing:
{E70.Thing -- P92B.was_brought_into_existence_by -> E63.Beginning_of_Existence:
{ E63.Beginning_of_Existence -- (P9B.forms_part_of)[0,n] -> E5.Event:
{ E5.Event -- P7F.took_place_at-> E53.Place :
{E53.Place --(P89F.falls_within)[0,n]-> E53.Place [--P2F.has_type
-> E55.Type]
}
}

}
}

- Found or acquired at

Especially in the archaeological field of study, one is interested in Things found in a certain Place. This place could be a *from (history)* for this thing especially when the creation place is unknown. The same applies for the place where a Thing is acquired, which means the place where the transfer of legal ownership of a Thing takes place.
The respective specialized fundamental relationship "*Thing found or acquired at Place"* is:

E70.Thing -- (F4B.is_component_of)$^{[0,n]}$ -> **E70.Thing:**
{E19.Physical_Object -- P12B.was_present_at -> **C2.Finding_Event:**
{ C2.Finding_Event -- (P9B.forms_part_of)$^{[0,n]}$ -> **E5.Event:**
{ E5.Event -- P7F.took_place_at -> **E53.Place :**
{E53.Place --(P89F.falls_within**)$^{[0,n]}$-> E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
}
}
OR
E19.Physical_Object -- P24B.changed_ownership_through -> **E8.Acquisition:**
{ **E8.Acquisition**-- (P9B.forms_part_of)$^{[0,n]}$ -> **E5.Event:**
{ E5.Event -- P7F.took_place_at -> **E53.Place :**
{E53.Place --(P89F.falls_within**)$^{[0,n]}$-> E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
}
}

- Was created/produced by person from

This specialized relationship gives the user the possibility to restrict the results of the "*from*" FR to only the Things that we created by an Actor from a specific Place.
The respective specialized fundamental relationship "*Thing created/produced by person from Place"* is:

E70.Thing -- (F4B.is_component_of)$^{[0,n]}$ -> **E70.Thing:**
{E70.Thing -- P92B.was_brought_into_existence_by -> **E63.Beginning_of_Existence:**
{ **E63.Beginning_of_Existence** -- (P9B.forms_part_of)$^{[0,n]}$ -> **E7.Activity:**
{ E7.Activity -- P14F.carried_out_by -> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)$^{[0,n]}$-> **E39.Actor:**
{ E39.Actor -- P74F.has_current_or_former_residence -> **E53.Place :**
{E53.Place --(P89F.falls_within**)$^{[0,n]}$-> E53.Place** [--P2F.has_type -> **E55.Type**]
}
OR
E39.Actor -- P92B.was_brought_into_existence_by ->
E63.Beginning_of_Existence :
{ E63.Beginning_of_Existence -- P7F.took_place_at ->
E53.Place:
{E53.Place --(P89F.falls_within**)$^{[0,n]}$-> E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}
}

- Is/was located in

One may be interested in Things that are or ever have been located in a certain Place.
The respective specialized fundamental relationship "*Thing is/was located in Place"* is:

E70.Thing -- (F4B.is_component_of)$^{[0,n]}$ -> **E70.Thing:**
{E18.Physical_Thing -- P53F.has_former_or_current_location ->**E53.Place :**
{E53.Place --(P89F.falls_within**)$^{[0,n]}$-> E53.Place** [--P2F.has_type -> **E55.Type**]
}
OR
E18.Physical_Thing –{P51F.has_former_or_current_owner OR
P49F.has_former_or_current_keeper}-> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)$^{[0,n]}$ -> **E39.Actor:**
{E39.Actor -- P74F.has_current_or_former_residence -> **E53.Place:**
{E53.Place --(P89F.falls_within**)$^{[0,n]}$-> E53.Place** [--P2F.has_type
-> **E55.Type**]
}
}
}
}

- Moved from

With this specialization one may search for things moved from one Place.

E70.Thing -- (F4B.is_component_of)$^{[0,n]}$ -> **E70.Thing:**
{**E19.Physical_Object** --P25B.moved_by -> **E9.Move** :
{ **E9.Move** -- P27F.moved_from -> **E53.Place**:
{**E53.Place** --(P89F.falls_within)$^{[0,n]}$-> **E53.Place** [--P2F.has_type ->
E55.Type]
}
}
}

- Moved to

With this specialization one may search for things moved to one Place.

E70.Thing -- (F4B.is_component_of)$^{[0,n]}$ -> **E70.Thing**:
{**E19.Physical_Thing** --P25B.moved_by -> **E9.Move** :
{ **E9.Move** -- P26F.moved_to-> **E53.Place**:
{**E53.Place** --(P89F.falls_within)$^{[0,n]}$-> **E53.Place** [--P2F.has_type ->
E55.Type]
}
}
}

Thing-Thing

a. has met

This is a general relationship meaning that a Thing has been in the same place at the same time with another Thing. This can further imply that a Thing is part of another Thing, or two Things have been present at the same Event.
The respective general fundamental relationship "*Thing has met Thing"* is:

E70.Thing -- (P46B.forms_part_of)$^{[0,n]}$ -> **E18.Physical_Thing:**
{**E18.Physical_Thing** -- P12B.was_present_at -> **E5.Event :**
{ E5.Event -- (P9B.forms_part_of) $^{[0,n]}$ -> **E5.Event**:

{ **E5.Event** -- P12F.occurred_in_the_presence_of -> **E70.Thing:**
{**E70.Thing** --(P46F.is_composed_of) [0,n] -> **E70.Thing** [--P2F.has_type-> **E55.Type**]
}
}
}
}

b. refers to or is about

A Thing can have as a theme or refer to another Thing, or a Thing may be created having as source a Thing that refers to another Thing. In the latter case we suppose subject preserving output events that inherit the "digitized" link. Also a Thing can be a copy of a Thing or bear similarity with a Thing that refers to another Thing and in this way they refer to the same Thing. These relationships can be modeled in the general relationship "is about".

In CIDOC-CRM properties that define the "refer to" or "is about" relationship are :

- P62F.depicts
- P67F.refers_to

The respective general fundamental relationship "*Thing refers to or is about Thing"* is:

E70.Thing --(F5F.consists_of_shows_features_of)[0,n]-> **E70.Thing:**
{ **E24.Physical_Man-Made_Thing** -- P62F.depicts -> **E70.Thing:**
{**E70.Thing** -- (F4F.is_composed_of)[0,n]-> **E70.Thing** [-- P2F.has_type -> **E55.Type**]
}
OR
E89.Propositional_Object --P67F.refers_to-> **E70.Thing:**
{**E70.Thing** -- (F4F.is_composed_of)[0,n]-> **E70.Thing** [-- P2F.has_type -> **E55.Type**]
}

OR
E24.Physical_Man-Made_Thing -- P128F.carries -> **E73.Information_Object:**
{ **E73.Information_Object** --P67F.refers_to -> **E70.Thing:**
{**E70.Thing** -- (F4F.is_composed_of)[0,n]-> **E70.Thing** [-- P2F.has_type -> **E55.Type**]
}
}
OR
D1.Digital_Object -- L11B.was_output_of -> **D7.Digital_Machine_Event:**
{**D7.Digital_Machine_Event** --(P9B.forms_part_of)[0,n] ->
D2.Digitization_Process :
{**D2.Digitization_Process** -- L1F.digitized ->**E70.Thing:**
{**E70.Thing** -- (F4F.is_composed_of)[0,n]-> **E70.Thing** [-- P2F.has_type -> **E55.Type**]
OR
E24.Physical_Man-Made_Thing -- P62F.depicts -> **E70.Thing:**
{**E70.Thing** -- (F4F.is_composed_of)[0,n]-> **E70.Thing** [-- P2F.has_type -> **E55.Type**]
}
OR
E89.Propositional_Object --P67F.refers_to-> **E70.Thing:**
{**E70.Thing** -- (F4F.is_composed_of)[0,n]-> **E70.Thing** [-- P2F.has_type -> **E55.Type**]
}

OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{ **E73.Information_Object** --P67F.refers_to ->
E70.Thing:

{E70.Thing -- (F4F.is_composed_of)[0,n]-> **E70.Thing**
[-- P2F.has_type -> **E55.Type**]
}
}
}
}
}

c. is referred to by

A Thing may be referred to by Things that have as theme or subject the Thing, or that refer to or are about the Thing. We may even expand the reference to Things that have as a part the Thing in reference.
In CIDOC-CRM properties that define the "is referred to by" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Thing is referred to by Thing"* is:

E70.Thing -- (F4B.is_component_of)[0,n] -> **E70.Thing:**
{E70.Thing--P67B.is_referred_to_by -> **E89.Propositional_Object:**
{ E89.Propositional_Object --(P148B.is_component_of)[0,n]->
E89.Propositional_Object[--P2F.has_type -> **E55.Type**]
OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] ->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing[--P2F.has_type -> **E55.Type**]
OR
E24.Physical_Man-Made_Thing -- L1B.was_digitized_by->
D2.Digitization_Process:
{D2.Digitization_Process--(P9F.consists_of)[0,n]->
D7.Digital_Machine_Event:
{D7.Digital_Machine_Event-- L11F.had_output->
E70.Thing:
{E70.Thing--
(F5B.forms_part_of_shows_features_of)[0,n]->
E70.Thing [--P2F.has_type -> **E55.Type**]
}
}
}
}
}
}
OR
E70.Thing -- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing** :
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] -> **E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing[--P2F.has_type -> **E55.Type**]
OR
E24.Physical_Man-Made_Thing -- L1B.was_digitized_by->
D2.Digitization_Process:
{D2.Digitization_Process--(P9F.consists_of)[0,n]->
D7.Digital_Machine_Event:
{D7.Digital_Machine_Event-- L11F.had_output-> **E70.Thing:**
{E70.Thing--(F5B.forms_part_of_shows_features_of)[0,n]-
> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
}

}
}
}
OR
E70.Thing -- L1B.was_digitized_by-> **D2.Digitization_Process:**
{D2.Digitization_Process--(P9F.consists_of)[0,n]-> **D7.Digital_Machine_Event:**
{D7.Digital_Machine_Event-- L11F.had_output-> **E70.Thing:**
{ E70.Thing--(F5B.forms_part_of_shows_features_of) [0,n]-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
}
}
}

d. from

This relationship is about the origin of a Thing in regard with another Thing. This may include the primitive constituents of a Thing (*is part of* relationship), the Thing transformed into another Thing, or the Things removed from another Thing. Here we can not include the parts added to a Thing as these parts do not change the identity of the Thing. It is more suitable to include this property to the *has part* FR.
So, in this FR the Thing in the range is the wider concept of the two Things.
In CIDOC-CRM properties that define the "from" relationship are :

- P46B.forms_part_of
- P106B. forms_part_of
- P148B.is_component_of

The respective general fundamental relationship "*Thing from Thing"* is:

E70.Thing -- (F4B.is_component_of)[0,n] -> **E70.Thing:**
{ **E70.Thing** [--P2F.has_type -> **E55.Type**]
OR
E18.Physical_Thing -- P123B.resulted_from -> **E81.Transformation :**
{ E81.Transformation -- (P9B.forms_part_of) [0,n] -> **E81.Transformation**:
{ E81.Transformation -- P124F.transformed -> **E70.Thing *:***
{E70.Thing -- (F4B.is_component_of)[0,n] -> **E70.Thing** [-- P2F.has_type -> **E55.Type**]
}
}
}
OR
E24.Physical_Man-Made_Thing -- P31B.was_modified_by-> **E11.Modification:**
{ E11.Modification-- (P9B.forms_part_of) [0,n] -> **E7.Activity**:
{
E7.Activity -- P110F.augmented-> **E70.Thing** [-- P2F.has_type -> **E55.Type**]
OR
E7.Activity -- P112F.diminished-> **E18.Physical_Thing** [-- P2F.has_type -> **E55.Type**]
}
}
}

Specializations:

In the fundamental relationship *from (history)* we can also define some sub-fundamental relationships, which contain more restricted information. This information is however commonly asked for by the users and a more specialized query may return better results in terms of precision based to what they actually want to know about.

- Is part of

In this specialization the user can restrict the results of the *from* FR to only the parts that the given Thing consists of.

The respective specialized fundamental relationship "*Thing is part of Thing"* is:

E70.Thing -- (F4B.is_component_of)[0,n] -> **E70.Thing** [-- P2F.has_type -> **E55.Type**]

e. has part

This is a FR that indicates the part-whole relationship of a Thing with other Things. More specifically given a Thing-part we can find the Thing(s)-whole to which the Thing-part belongs. It is the reverse relationship of *Thing is part of Thing.*

The respective general fundamental relationship "*Thing has part Thing"* is:

E70.Thing--(F4F.is_composed_of)[0,n]-> **E70.Thing:**
{**E70.Thing** [-- P2F.has_type -> **E55.Type**]
OR
E24.Physical_Man-Made_Thing -- P110B.was_augmented_by -> **E79.Part_Addition:**
{E79.Part_Addition -- (P9B.forms_part_of) [0,n] -> **E79.Part_Addition**:
{E79.Part_Addition-- P111F.added -> **E18.Physical_Thing** [-- P2F.has_type -> **E55.Type**]
}
}
}

f. is similar or same with

Similarity between two instances of the category Thing, may mean that one Thing is a copy of another, so they share the same features, or they are anchors of an annotation that indicates similarity.
The respective general fundamental relationship "*Thing is similar or same with Thing"* is:

E70.Thing -- { (P130F.shows_features_of) [0,n] OR (P130B.features_are_also_found_on [0,n] } -> **E70.Thing:**
{ E70.Thing [-- P2F.has_type -> **E55.Type**]
OR
E70.Thing -- L54B.is_same-as -> **D38.Same-as :**
{ D38.Same-as -- L54F.is_same-as ->**E70.Thing** [--P2F.has_type -> **E55.Type**]
}
}

Thing-Actor

a. has met

This is a general relationship meaning that a Thing has been in the same place at the same time with an Actor. This can further imply a Thing and an Actor have been present at the same Event. Here it is not right to use further inference for the Group the Actor belongs to, since it is not so probable that the Group has also been in the same event as the Actors have.
We assume that when a Thing as a whole appears at an event, then its parts also are present in the same event.
The respective general fundamental relationship "*Thing has met Actor"* is:

E70.Thing -- (P46B.forms_part_of) [0,n] -> **E70.Thing:**
{ **E18.Physical_Thing** -- P12B.was_present_at -> **E5.Event :**
{ E5.Event -- (P9B.forms_part_of) [0,n] -> **E5.Event**:
{ E5.Event -- P12F.occurred_in_the_presence_of -> **E39.Actor:**
{ E39.Actor -- (P107B.is_current_or_former_member_of)[0,n] ->
E39.Actor [--P2F.has_type -> **E55.Type**]
}

}
}
}

b. Is referred to by

An actor may refer to a Thing by material or immaterial Things they create and which refer to some Thing. If the Thing that is referenced is part of a bigger whole, then we assume that the reference also is valid for the bigger Thing.
In CIDOC-CRM properties that define the "is referred by" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Thing is referred to by Actor"* is:

E70.Thing -- (F4B.is_component_of)[0,n] -> **E70.Thing:**
{E70.Thing-- P67B.is_referred_to_by -> **E89.Propositional_Object:**
{E89.Propositional_Object -- P94B.was_created_by -> **E65.Creation:**
{ E65.Creation -- (P9B.forms_part_of) [0,n] -> **E65.Creation**:
{ E65.Creation-- P14F.carried_out_by -> **E39.Actor:**
{ E39.Actor -- (P107B.is_current_or_former_member_of)[0,n] -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] ->
E24.Physical_Man-Made_Thing:
{ E24.Physical_Man-Made_Thing--P108B.was_produced_by ->
E12.Production:
{ E12.Production -- (P9B.forms_part_of) [0,n] -> **E5.Event**:
{E7.Activity -- P14F.carried_out_by-> **E39.Actor:**
{ E39.Actor --
(P107B.is_current_or_former_member_of) [0,n] -
> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}
}
OR
E70.Thing-- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] ->
E24.Physical_Man-Made_Thing:
{ E24.Physical_Man-Made_Thing--P108B.was_produced_by ->
E12.Production:
{ E12.Production -- (P9B.forms_part_of) [0,n] -> **E5.Event**:
{E7.Activity -- P14F.carried_out_by-> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of
)[0,n] -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}

c. refers to or is about

A Thing can have as a theme or refer to an Actor, or a Thing may be created having as source a Thing that refers to an Actor. In the latter case we suppose subject preserving output events that inherit the "digitized" link, so that in the procession chain the original theme is not lost. Also a Thing can be a copy of a Thing or bear similarity with a Thing that refers to an Actor and in this way they refer to the same Actor. These relationships can be modeled in the general relationship "is about".
In CIDOC-CRM properties that define the "is about" relationship are :

- P62F.depicts
- P67F.refers_to

The respective general fundamental relationship "*Thing refers to or is about Actor* " is:

E70.Thing --(F5F.consists_of_shows_features_of)[0,n]-> **E70.Thing:**
{E24.Physical_Man-Made_Thing -- P62F.depicts -> **E39.Actor:**
{E39.Actor -- (P107F.has_current_or_former_member) [0,n] -> **E39.Actor** [-- P2F.has_type -> **E55.Type**]
}
OR
E89.Propositional_Object --P67F.refers_to -> **E39.Actor:**
{E39.Actor -- (P107F.has_current_or_former_member) [0,n] -> **E39.Actor** [-- P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries -> **E73.Information_Object:**
{ **E73.Information_Object** --P67F.refers_to-> **E39.Actor:**
{E39.Actor -- (P107F.has_current_or_former_member) [0,n] -> **E39.Actor** [-- P2F.has_type -> **E55.Type**]
}
}
OR
D1.Digital_Object -- L11B.was_output_of -> **D7.Digital_Machine_Event:**
{D7.Digital_Machine_Event --(P9B.forms_part_of)[0,n] ->
D2.Digitization_Process :
{D2.Digitization_Process -- L1F.digitized ->**E70.Thing:**
{E24.Physical_Man-Made_Thing -- P62F.depicts -> **E39.Actor:**
{E39.Actor -- (P107F.has_current_or_former_member) [0,n] -> **E39.Actor** [-- P2F.has_type -> **E55.Type**]
}
OR
E89.Propositional_Object --P67F.refers_to-> **E39.Actor:**
{E39.Actor -- (P107F.has_current_or_former_member) [0,n] -> **E39.Actor** [-- P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{ **E73.Information_Object** --P67F.refers_to ->
E39.Actor:
{E39.Actor --
(P107F.has_current_or_former_member) [0,n] ->
E39.Actor [-- P2F.has_type -> **E55.Type**]
}
}
}
}
}
}

d. by

This relation connects a Thing with the Actor who holds the property of having an active participation in the creation, modification or any other action on the Thing. We accept that parts of a Thing and the Thing itself share the same actors.
In this case we can further assume that when an actor performs an action they most probably do so as a role of being a part in a group or institution, for instance if Thomas Miller digitally acquires a photograph of a statue and Thomas Miller is a member of FORTH foundation then this photograph may also be considered as a product of FORTH.
The respective general fundamental relationship "*Thing by Actor* " is:

E70.Thing--(F4F.is_composed_of)$^{[0,n]}$ -> **E70.Thing :**
{ E70.Thing -- {P92B.was_brought_into_existence_by OR P16B.was_used_for OR P39B.was_measured_by OR P31B.was_modified_by }-> **E7.Activity:**
{ E7.Activity -- (P9B.forms_part_of)$^{[0,n]}$ -> **E7.Activity:**
{ E7.Activity -- P14F.carried_out_by -> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)$^{[0,n]}$ -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
OR
E18.Physical_Thing -- P12B.was_present_at-> **C2.Finding_Event:**
{ C2.Finding_Event -- (P9B.forms_part_of)$^{[0,n]}$ -> **E5.Event:**
{ E5.Event --P11F.had_participant-> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)$^{[0,n]}$ -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
OR
E18.Physical_Thing-- P24B.changed_ownership_through -> **E8.Acquisition:**
{ E8.Acquisition -- P22F.transferred_title_to -> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)$^{[0,n]}$ -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
OR
E18.Physical_Thing--{P51F.has_former_or_current_owner OR P109F.has_current_or_former_curator OR P105F. right_held_by } -> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)$^{[0,n]}$ -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
OR
E18.Physical_Thing-- P104F.is_subject_to ->**E30.Right:**
{ E30.Right --P75B.is_possessed_by-> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)$^{[0,n]}$ -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}

Specializations:

In the fundamental relationship *by* we can also define some sub-fundamental relationships, which contain more restricted information. This information is however commonly asked for by the users and a more specialized query may return better results based to what they actually want to know about.

- Used by

It is common that one is interested in the Things that were used by an Actor.
The respective specialized fundamental relationship "*Thing used by Actor* " is:

E70.Thing--(F4F.is_composed_of)[0,n] -> **E70.Thing :**
{E70.Thing -- P16B.was_used_for-> **E7.Activity:**
{ E7.Activity -- (P9B.forms_part_of)[0,n] -> **E5.Event:**
{E7.Activity-- P14F.carried_out_by-> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)[0,n] ->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
}
}
}

- Created by

The most important of the specialized relationships of the *by* FR is the "created by". With it the results are restricted to the Things an Actor has created.
The respective specialized fundamental relationship "*Thing created by Actor "* is:

E70.Thing--(F4F.is_composed_of)[0,n] -> **E70.Thing :**
{ E70.Thing -- P92B.was_brought_into_existence_by-> **E63.Beginning_of_Existence:**
{ E5.Event-- (P9B.forms_part_of)[0,n] -> **E5.Event:**
{E7.Activity -- P14F.carried_out_by -> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)[0,n] ->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
}
}
}

- Digitized by

This is a very specialized relation for objects that have undergone some digitization process and it is very common among users in this discourse.
The respective specialized fundamental relationship "*Thing digitized by Actor "* is:

E70.Thing--(F4F.is_composed_of)[0,n] -> **E70.Thing :**
{ E70.Thing -- L1B.was_digitized_by-> **D2.Digitization_Process:**
{ E5.Event-- (P9B.forms_part_of)[0,n] -> **E5.Event:**
{ E7.Activity -- P14F.carried_out_by -> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)[0,n] ->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
}
}
}

- Modified by

This relationship enables the user to restrict the results to the Things that have been modified by an Actor.
The respective specialized fundamental relationship "*Thing modified by Actor "* is:

E18.Physical_Thing --(P46F.is_composed_of) [0,n]-> **E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing -- P31B.was_modified_by-> **E11.Modification:**
{ E5.Event-- (P9B.forms_part_of)[0,n] -> **E5.Event:**
{E7.Activity --P14F.carried_out_by-> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)[0,n] ->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
}

}
}

- Found or acquired by

With this relationship one can get all the Things that were found or acquired by a specific Actor.
The respective specialized fundamental relationship "*Thing found or acquired by Actor"* is:

E70.Thing -- (P46F.is_composed_of) [0,n]-> **E18.Physical_Thing:**
{**E18.Physical_Thing** -- P12B.was_present_at->**C2.Finding_Event:**
{ **C2.Finding_Event** -- (P9B.forms_part_of)[0,n] -> **E5.Event:**
{**E5.Event**--P11F.had_participant-> **E39.Actor:**
{**E39.Actor** -- (P107B.is_current_or_former_member_of)[0,n] ->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
}
}
OR
E18.Physical_Thing-- P24B.changed_ownership_through -> **E8.Acquisition:**
{**E5.Event**-- (P9B.forms_part_of)[0,n] -> **E5.Event:**
{**E7.Activity**-- P14F.carried_out_by-> **E39.Actor:**
{**E39.Actor** -- (P107B.is_current_or_former_member_of)[0,n] ->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
}
}
OR
E18.Physical_Thing--P51F.has_former_or_current_owner -> **E39.Actor:**
{**E39.Actor** -- (P107B.is_current_or_former_member_of)[0,n] -> **E39.Actor** [--
P2F.has_type -> **E55.Type**]
}
}

e. from

A thing can be or once have been under the possession or responsibility of one or more persons or organizations. With this relation we can track Things that have been under the possession or authority of an Actor, either individual or a group of people.
In CIDOC-CRM properties that define the "is about" relationship are :

- P49F.has_former_or_current_keeper
- P51F.has_former_or_current_owner

The respective general fundamental relationship "*Thing from Actor "* is:

E70.Thing --(P46B.forms_part_of) [0,n] -> **E18.Physical_Thing:**
{**E18.Physical_Thing**--{P49F.has_former_or_current_keeper OR
P51F.has_former_or_current_owner } -> **E39.Actor:**
{**E39.Actor** -- (P107B.is_current_or_former_member_of)[0,n] -> **E39.Actor** [--
P2F.has_type -> **E55.Type**]
}
}

Thing-Event

a. refers to or is about

A Thing can have as a theme or refer to an Event, or a Thing may have been created having as source a Thing that refers to an Event. In the latter case we suppose subject preserving output events that inherit the "digitized" link, so that in the procession chain the original theme is not lost. Also a Thing can be a

copy of a Thing or bear similarity with a Thing that refers to an Event and in this way they refer to the same Event. These relationships can be modeled in the general relationship "is about".
In CIDOC-CRM properties that define the "is about" relationship are :

- P62F.depicts
- P67F.refers_to

The respective general fundamental relationship "*Thing refers to or is about Event"* is:

E70.Thing -- (F5F.consists_of_shows_features_of)[0,n] -> **E70.Thing:**
{ E24.Physical_Man-Made_Thing -- P62F.depicts -> **E5.Event:**
{E5.Event --(P9B.forms_part_of)[0,n] -> **E5.Event** [-- P2F.has_type -> **E55.Type**]
}
OR
E89.Propositional_Object --P67F.refers_to -> **E5.Event:**
{E5.Event --(P9B.forms_part_of)[0,n] -> **E5.Event** [-- P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries -> **E73.Information_Object:**
{ **E73.Information_Object** --P67F.refers_to -> **E5.Event:**
{E5.Event --(P9B.forms_part_of)[0,n] -> **E5.Event** [-- P2F.has_type -> **E55.Type**]
}
}
OR
D1.Digital_Object -- L11B.was_output_of -> **D7.Digital_Machine_Event:**
{D7.Digital_Machine_Event --(P9B.forms_part_of)[0,n] ->
D2.Digitization_Process :
{D2.Digitization_Process -- L1F.digitized -> **E18.Physical_Thing:**
{ E24.Physical_Man-Made_Thing -- P62F.depicts -> **E5.Event:**
{E5.Event --(P9B.forms_part_of)[0,n] -> **E5.Event** [-- P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{ **E73.Information_Object** --P67F.refers_to ->
E5.Event:
{E5.Event --(P9B.forms_part_of)[0,n] ->
E5.Event [-- P2F.has_type -> **E55.Type**]
}
}
}
}
}
}

b. is referred to at

Things may be referred by Events during which Things that refer to them were created.

In CIDOC-CRM properties that define the "is referred to at" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Thing is referred to at Event"* is:

E70.Thing -- (F4B.is_component_of)[0,n] -> **E70.Thing:**
{E70.Thing--P67B.is_referred_to_by -> **E89.Propositional_Object:**
{E89.Propositional_Object -- P94B.was_created_by -> **E65.Creation:**
{ E65.Creation -- (P9B.forms_part_of) [0,n] -> **E5.Event** [--P2F.has_type -> **E55.Type**]
}

OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] ->
E24.Physical_Man-Made_Thing:
{ E24.Physical_Man-Made_Thing--P108B.was_produced_by ->
E12.Production:
{ E12.Production -- (P9B.forms_part_of) [0,n] -> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
}
}
}
OR
E70.Thing-- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing--P108B.was_produced_by -> **E12.Production:**
{ E12.Production -- (P9B.forms_part_of) [0,n] -> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
}
}

c. from

Things can be linked with certain events despite the fact that the kind of link may not be known to the user. So questions concerning the placement of Things in events may be as generic as a generic *from.* More specific constraints on this relationship can be applied at the specializations of the *from* relationship.
The respective general fundamental relationship "*Thing from Event"* is:

E70.Thing -- (F4B.is_component_of)[0,n] -> **E70.Thing:**
{ E70.Thing -- P12B.was_present_at-> **E5.Event:**
{E5.Event--(P9B.forms_part_of)[0,n]->**E5.Event** [--P2F.has_type -> **E55.Type**]
}
OR
E70.Thing -- P30B.custody_transferred_through-> **E5.Event:**
{E5.Event--(P9B.forms_part_of)[0,n]->**E5.Event** [--P2F.has_type -> **E55.Type**]
}
OR
E70.Thing -- P19B.was_made_for->**E7.Activity:**
{**E7.Activity**--(P9B.forms_part_of)[0,n]->**E5.Event** [--P2F.has_type -> **E55.Type**]
}
OR
E70.Thing -- P113B.was_removed_by->**E7.Activity:**
{**E7.Activity**--(P9B.forms_part_of)[0,n]->**E5.Event** [--P2F.has_type -> **E55.Type**]
}
OR
E70.Thing -- P147B.was_curated_by->**E7.Activity:**
{**E7.Activity**--(P9B.forms_part_of)[0,n]->**E5.Event** [--P2F.has_type -> **E55.Type**]
}

}

Specializations:
In the fundamental relationship *from* we can also define some sub-fundamental relationships, which contain more restricted information. This information is however commonly asked for by the users and a more specialized query may return better results based to what they actually want to know about.

- Destroyed in

In this specialization we are interested in the Things that were destroyed in an Event. We suppose that the destruction of a thing further means the destruction of its parts.
The respective specialized fundamental relationship "*Thing destroyed in Event"* is:

E70.Thing -- (F4B.is_component_of)$^{[0,n]}$-> **E70.Thing:**
{E70.Thing-- P93B.was_taken_out_of_existence_by-> **E64.End_of_Existence:**
{E64.End_of_Existence --(P9B.forms_part_of)$^{[0,n]}$->
E5.Event [--P2F.has_type -> **E55.Type**]
}
}

- Created in

Things created in Events are of special interest for researchers and scientists so this specialization is very useful. We take as granted that the parts of a Thing are created during the creation of the Thing.
The respective specialized fundamental relationship "*Thing created in Event"* is:

E70.Thing -- (F4B.is_component_of)$^{[0,n]}$-> **E70.Thing:**
{E70.Thing -- P92B.was_brought_into_existence_by-> **E63.Beginning_of_Existence:**
{ E63.Beginning_of_Existence --(P9B.forms_part_of)$^{[0,n]}$->**E5.Event**[--P2F.has_type->
E55.Type]
}
}

- Modified in

In this specialization we are interested in Things that have been modified during an event.
The respective specialized fundamental relationship "*Thing modified in Event"* is:

E70.Thing -- (P46B.forms_part_of) $^{[0,n]}$ -> **E70.Thing:**
{E24.Physical_Man-Made_Thing--P31B.was_modified_by-> **E11.Modification:**
{**E11.Modification**--(P9B.forms_part_of)$^{[0,n]}$->
E5.Event [--P2F.has_type -> **E55.Type**]
}
}

- Used in

Often one is interested in the Things that were used during an Event.
The respective specialized fundamental relationship "*Thing used in Event"* is:

E70.Thing -- (F4B.is_component_of)$^{[0,n]}$-> **E70.Thing:**
{**E70.Thing** -- P16B.was_used_for->**E7.Activity:**
{**E7.Activity**--(P9B.forms_part_of)$^{[0,n]}$->**E5.Event** [--P2F.has_type -> **E55.Type**]
}
OR
E70.Thing -- P19B.was_made_for->**E7.Activity:**
{**E7.Activity**--(P9B.forms_part_of)$^{[0,n]}$->**E5.Event** [--P2F.has_type -> **E55.Type**]
}

}

- Digitized in

This specialization is especially useful for users in the concept of digitization processes. With this can one retrieve the events that were responsible for the digitization of the specific Thing.
The respective specialized fundamental relationship "*Thing digitized in Event"* is:

E70.Thing --(P46B.forms_part_of)$^{[0,n]}$ -> **E70.Thing:**
{**E70.Thing** -- L1B.was_digitized_by-> **D2.Digitization_Process:**
{ **D2.Digitization_Process** --(P9B.forms_part_of)$^{[0,n]}$-> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
}

Thing-Time

The Thing-Event fundamental relationships can be switched to Thing-Time fundamental relationships, by further adding the CIDOC-CRM property P4F.has_time-span at the range category Event **(E5.Event** --P4F.has_time-span->**E52.Time-Span)**. This happens because Time refers to the chronological definition of Events.

a. refers to or is about

A Thing can have as a theme or refer to a Time span, or a Thing may have been created having as source a Thing that refers to a Time span. In the latter case we suppose subject preserving output events that inherit the "digitized" link, so that in the procession chain the original theme is not lost. Also a Thing can be a copy of a Thing or bear similarity with a Thing that refers to a Time span and in this way they refer to the same Time span. These relationships can be modeled in the general relationship "is about".

In CIDOC-CRM properties that define the "is about" relationship are :

- P62F.depicts
- P67F.refers_to

The respective general fundamental relationship "*Thing refers to or is about Time"* is:

E70.Thing -- (F5F.consists_of_shows_features_of)$^{[0,n]}$ -> **E70.Thing:**
{ E24.Physical_Man-Made_Thing -- P62F.depicts -> **E52.Time-Span:**
{E52.Time-Span -- (P86B.contains)$^{[0,n]}$ **->E52.Time-Span**[-- P2F.has_type -> **E55.Type**]
}
OR
E89.Propositional_Object --P67F.refers_to -> **E52.Time-Span:**
{E52.Time-Span --(P86B.contains)$^{[0,n]}$->**E52.Time-Span**[-- P2F.has_type -> **E55.Type**]
}

OR
E24.Physical_Man-Made_Thing -- P128F.carries -> **E73.Information_Object:**
{ **E73.Information_Object** --P67F.refers_to -> **E52.Time-Span:**
{E52.Time-Span --(P86B.contains)$^{[0,n]}$->**E52.Time-Span**[-- P2F.has_type -> **E55.Type**]
}
}
OR
D1.Digital_Object -- L11B.was_output_of -> **D7.Digital_Machine_Event:**
{D7.Digital_Machine_Event --(P9B.forms_part_of)$^{[0,n]}$ -> **D2.Digitization_Process** :
{D2.Digitization_Process -- L1F.digitized -> **E18.Physical_Thing:**
{ E24.Physical_Man-Made_Thing -- P62F.depicts -> **E52.Time-Span:**
{E52.Time-Span --(P86B.contains)$^{[0,n]}$->**E52.Time-Span**[-- P2F.has_type -> **E55.Type**]
}

OR
E24.Physical_Man-Made_Thing -- P128F.carries -> **E73.Information_Object:**
{ **E73.Information_Object** --P67F.refers_to -> **E52.Time-Span:**

{**E52.Time-Span** --(P86B.contains)[0,n]->
E52.Time-Span[-- P2F.has_type -> **E55.Type**]
}
}
}
}
}
}

b. has met

Things can be linked with certain events despite the fact that the kind of link may not be known to the user. So questions concerning the placement of Things in events may be as generic as a generic *has met*. More specific constraints on this relationship can be applied at the specializations of the *has met* relationship.
The respective general fundamental relationship "*Thing has met Time"* is:

E70.Thing -- (F4B.is_component_of)[0,n] -> **E70.Thing:**
{E70.Thing -- P12B.was_present_at-> **E5.Event :**
{E5.Event --(P9B.forms_part_of)[0,n]-> **E5.Event:**
{E5.Event-- P4F.has_time-span-> **E52.Time-Span:**
{E52.Time-Span --(P86F.falls_within)[0,n]-> **E52.Time-Span**[--
P2F.has_type -> **E55.Type**]
}
}
}
}

Specializations:
In the fundamental relationship *has met* we can also define some sub-fundamental relationships, which contain more restricted information. This information is however commonly asked for by the users and a more specialized query may return better results based to what they actually want to know about.

- Destroyed in

In this specialization we are interested in the Things that were destroyed in some Time. We suppose that the destruction of a thing further means the destruction of its parts.
The respective specialized fundamental relationship "*Thing destroyed in Time"* is:

E70.Thing -- (F4B.is_component_of)[0,n]-> **E70.Thing:**
{E70.Thing-- P93B.was_taken_out_of_existence_by-> **E64.End_of_Existence:**
{E64.End_of_Existence --(P9B.forms_part_of)[0,n]->
E5.Event:
{E5.Event-- P4F.has_time-span-> **E52.Time-Span:**
{E52.Time-Span --(P86F.falls_within)[0,n]-> **E52.Time-Span**[--
P2F.has_type -> **E55.Type**]
}
}
}
}

- Created in

Things created in a specific Time span are of special interest for researchers and scientists so this specialization is very useful. Because it happens that people are not aware of the name of creation events, this sub-relation is very useful as it provides them with the option of searching for creation within a time span. We take as granted that the parts of a Thing are created during the creation of the Thing.

The respective specialized fundamental relationship "*Thing created in Time"* is:

E70.Thing -- (F4B.is_component_of)[0,n]-> **E70.Thing:**
{E70.Thing -- P92B.was_brought_into_existence_by-> **E63.Beginning_of_Existence:**
{ E63.Beginning_of_Existence --(P9B.forms_part_of)[0,n]-> **E5.Event:**
{E5.Event-- P4F.has_time-span-> **E52.Time-Span:**
{E52.Time-Span --(P86F.falls_within)[0,n]-> **E52.Time-Span**[--
P2F.has_type -> **E55.Type**]
}

}
}
}

- Modified in

In this specialization we are interested in Things that have been modified during some Time.
The respective specialized fundamental relationship "*Thing modified in Time"* is:

E70.Thing -- (P46B.forms_part_of) [0,n] -> **E70.Thing:**
{E24.Physical_Man-Made_Thing--P31B.was_modified_by-> **E11.Modification:**
{**E11.Modification**--(P9B.forms_part_of)[0,n]-> **E5.Event:**
{E5.Event-- P4F.has_time-span-> **E52.Time-Span:**
{E52.Time-Span --(P86F.falls_within)[0,n]-> **E52.Time-Span**[--
P2F.has_type -> **E55.Type**]
}
}
}
}

- Used in

Often one is interested in the Things that were used during some Time.
The respective specialized fundamental relationship "*Thing used in Time"* is:

E70.Thing -- (F4B.is_component_of)[0,n]-> **E70.Thing:**
{**E70.Thing** -- P16B.was_used_for->**E7.Activity:**
{E7.Activity--(P9B.forms_part_of)[0,n]-> **E5.Event:**
{E5.Event-- P4F.has_time-span-> **E52.Time-Span:**
{E52.Time-Span --(P86F.falls_within)[0,n]-> **E52.Time-Span**[--
P2F.has_type -> **E55.Type**]
}
}
}
}

- Digitized in

This specialization is especially useful for users in the concept of digitization processes. With this can one retrieve the things that were digitized in a specific time.
The respective specialized fundamental relationship "*Thing digitized in Time"* is:

E70.Thing --(P46B.forms_part_of)[0,n] -> **E70.Thing:**
{**E70.Thing** -- L1B.was_digitized_by-> **E5.Event:**
{E5.Event-- P4F.has_time-span-> **E52.Time-Span:**
{E52.Time-Span --(P86F.falls_within)[0,n]-> **E52.Time-Span**[--
P2F.has_type -> **E55.Type**]
}
}
}

Thing-Concept

a. has type

This is a simple relationship that expresses the Concept a.c.a. Type of the Thing. This relationship may be of great importance when we want to gather all the Things that have the same type and then use them in conjunction with another FC regarding Things.
The respective general fundamental relationship "*Thing has type Concept*" is:

E70.Thing -- (F4F.is_composed_of)[0,n]-> **E70.Thing:**
{ E70.Thing -- P2F.has_type-> **E55.Type:**
{E55.Type --(P127F. has_broader_term)[0,n]-> **E55.Type**
}
OR
E70.Thing --P45F. consists_of-> **E57.Material:**
{ E57.Material --(P127F.has_broader_term)[0,n] -> **E55.Type**
}
OR
E70.Thing -- P92B.was_brought_into_existence_by-> **E7.Activity:**
{ E7.Activity --(P9B.forms_part_of)[0,n]->**E7.Activity:**
{E7.Activity--P33F.used_specific_technique->
E29.Design_or_Procedure:
{E29.Design_or_Procedure-- P68F.foresees_use_of->
E57.Material:
{ E57.Material --(P127F.has_broader_term)[0,n] ->
E55.Type
}
}
OR
E11.Modification -- P126F.employed -> E57.Material:
{ E57.Material --(P127F.has_broader_term)[0,n] -> **E55.Type**
}
}
}
OR
E70.Thing -- P44F.has_condition-> **E3.Condition_State:**
{ E3.Condition_State-- (P2F.has_type)[0,n] **->E55.Type }**
}

Specialization:

- Is made of

It is typical that a user may be interested in objects that are not generally of some type, but that are made of some specific material. For this case, they can use the specialization of the *has type* FR, **Thing is made of Concept.**

E70.Thing -- (F4F.is_composed_of)[0,n] -> **E70.Thing:**
{ E70.Thing --P45F. consists_of-> **E57.Material**
OR
E70.Thing -- P92B.was_brought_into_existence_by-> **E7.Activity:**
{E7.Activity --(P9B.forms_part_of)[0,n]->**E7.Activity:**
{E7.Activity--P33F.used_specific_technique->
E29.Design_or_Procedure:
{E29.Design_or_Procedure-- P68F.foresees_use_of->
E57.Material
}
OR

E11.Modification -- P126F.employed -> **E57.Material**
}
}
}

b. refers to

E70.Thing --(F5F.consists_of_shows_features_of)[0,n]-> **E70.Thing:**
{E24.Physical_Man-Made_Thing -- P62F.depicts -> **E55.Type:**
{ E55.Type --(P127F.has_broader_term)[0,n] -> **E55.Type**
}
OR
E89.Propositional_Object --P67F.refers_to -> **E55.Type:**
{ E55.Type --(P127F.has_broader_term)[0,n] -> **E55.Type**
E55.Type
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries -> **E73.Information_Object:**
{ **E73.Information_Object** --P67F.refers_to-> **E55.Type:**
{ E55.Type --(P127F.has_broader_term)[0,n] -> **E55.Type**
E55.Type
}
}
OR
D1.Digital_Object -- L11B.was_output_of -> **D7.Digital_Machine_Event:**
{D7.Digital_Machine_Event --(P9B.forms_part_of)[0,n] ->
D2.Digitization_Process :
{D2.Digitization_Process -- L1F.digitized ->**E70.Thing:**
{E24.Physical_Man-Made_Thing -- P62F.depicts -> **E55.Type:**
{ E55.Type --(P127F.has_broader_term)[0,n] -> **E55.Type**
}
OR
E89.Propositional_Object --P67F.refers_to-> **E55.Type:**
{ E55.Type --(P127F.has_broader_term)[0,n] -> **E55.Type**
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{ **E73.Information_Object** --P67F.refers_to ->
E55.Type:
{ E55.Type -- (P127F.has_broader_term)[0,n] ->
E55.Type
}
}
}
}
}
}

PLACE

The fundamental category place comprises geometric extents in space, on earth and on objects, which are independent of matter or temporal changes. Nevertheless intuitively Physical Features that are located in a Place are themselves considered to be Places. Such examples can be cities or settlements. Thus, by referring to Knossos we don't only speak of the ancient city but also the place where the city was located. In relation with the other FCs Place most commonly is used to identify where events take place or where material items, living or lifeless have been once or are permanently located.

Place-Place

a. has part

Given a Place one may be interested in broader or narrower Places it has some connection with. In this relationship, we are interested in the broader Places, in other words the Places that the Place forms part of.

In CIDOC-CRM properties that define the "has part" relationship are:

- P89B.contains

The respective general fundamental relationship "*Place has part Place"* is:

E53.Place--(P89B.contains)$^{[0,n]}$ -> **E53.Place** [--P2F.has_type -> **E55.Type**]

b. Is part of

This is the backward relationship of the "has part" relationship. Thus, here we are interested in Places that the given Place consists of as a part-whole relationship.
In CIDOC-CRM properties that define the "is part of" relationship are :

- P89F.falls_within

The respective general fundamental relationship "*Place is part of or limit of Place"* is:

E53.Place --(P89F.falls_within)$^{[0,n]}$ -> **E53.Place** [--P2F.has_type -> **E55.Type**]

c. Borders or overlaps with

To complete the relationships that a Place can have with another Place, there is also the proximity property among Places. So, with this relationship the user can find out which Places the given Place borders or overlaps with.
The following example shows how useful is for this relationship to also use the whole-part transitivity. Of course in the end we run the risk of concluding to that a Place borders or overlaps with itself or a broader Place, which actually is the fundamental Relationship Place *has part* Place.
The respective general fundamental relationship "*Place borders with Place"* is:
E53.Place --(P89B.contains)$^{[0,n]}$ -> **E53.Place *:***
{**E53.Place** -- {P122F.borders_with OR P121F.overlaps_with **}-> E53.Place:**
{E53.Place --(P89F.falls_within)$^{[0,n]}$ -> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
}

Place-Thing

a. refers to

A place can refer to a Thing by artefacts created or located in that place that have as subject or depict the thing. Also a Place refers to a Thing when the Actor that owns or keeps the Thing has or has ever had their residence in that Place. So with this relationship we can find out about the Places where a Thing is referred to at.
In CIDOC-CRM properties that define the "refer to" or "is about" relationship are:

- P62F.depicts
- P67F.refers_to

The respective general fundamental relationship "*Place refers to Thing"* is:

E53.Place--(P89B.contains) [0,n] -> **E53.Place:**
{E53.Place-- P53B.is_former_or_current_location_of -> **E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing -- P62F.depicts -> **E70.Thing:**
{E70.Thing --(F4F.is_composed_of)[0,n]-> **E70.Thing** [-- P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing --P128F.carries-> **E73.Information_Object:**
{E73.Information_Object -- P67F.refers_to -> **E70.Thing:**
{E70.Thing --(F4F.is_composed_of)[0,n]-> **E70.Thing** [-- P2F.has_type -> **E55.Type**]
}
}
}

OR
E53.Place-- P7B.witnessed-> **E5.Event:**
{ E5.Event --(P9F.consists_of)[0,n]-> **E5.Event:**
{ E65.Creation -- P94F.has_created-> **E18.Physical_Thing:**
{ E89.Propositional_Object --P67F.refers_to -> **E70.Thing:**
{E70.Thing --(F4F.is_composed_of)[0,n]-> **E70.Thing** [-- P2F.has_type -> **E55.Type**]
}
}
OR
E12.Production -- P108F.has_produced -> **E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing -- P62F.depicts -> **E70.Thing:**
{E70.Thing --(F4F.is_composed_of)[0,n]-> **E70.Thing** [-- P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing --P128F.carries->
E73.Information_Object:
{E73.Information_Object -- P67F.refers_to -> **E70.Thing:**
{E70.Thing --(F4F.is_composed_of)[0,n]-> **E70.Thing** [-- P2F.has_type -> **E55.Type**]
}
}
}
}
}

b. is referred to by

A Place may be referred to by Things that have as theme or subject the Place, or that refer to or are about the Place. We may even expand the reference to Places that contain the Place, both geographically and by context (so here we use the P88 CIDOC-CRM property).
In CIDOC-CRM properties that define the "is referred by" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Place is referred to by Thing"* is:

E53.Place -- (P89F.falls_within)[0,n]-> **E53.Place:**
{E53.Place --P67B.is_referred_to_by -> **E89.Propositional_Object:**
{ E89.Propositional_Object --(P148B.is_component_of) [0,n]-> **E89.Propositional_Object**[--P2F.has_type -> **E55.Type**]
OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**

{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] ->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing[--P2F.has_type -> **E55.Type**]
OR
E24.Physical_Man-Made_Thing -- L1B.was_digitized_by->
D2.Digitization_Process:
{D2.Digitization_Process--(P9F.has_part)[0,n]->
D7.Digital_Machine_Event:
{D7.Digital_Machine_Event-- L11F.had_output->
E70.Thing:
{ E70.Thing--
(F5B.forms_part_of_shows_features_of)[0,n] ->
E70.Thing [--P2F.has_type -> **E55.Type**]
}
}
}
}
}
OR
E53.Place -- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing** :
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] ->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing[--P2F.has_type -> **E55.Type**]
OR
E24.Physical_Man-Made_Thing -- L1B.was_digitized_by->
D2.Digitization_Process:
{D2.Digitization_Process--(P9F.has_part)[0,n]->
D7.Digital_Machine_Event:
{D7.Digital_Machine_Event-- L11F.had_output->
E70.Thing:
{ E70.Thing--
(F5B.forms_part_of_shows_features_of)[0,n] ->
E70.Thing [--P2F.has_type -> **E55.Type**]
}
}
}
}
}

c. has met

This relationship is used to connect a place with things that have once been located in a places or have been created there so consequently have been once located there. Movement of a Thing from one Place to another also implies that the Thing has once been at both Places. Moreover we can make the assumption that a Thing has been located in the same place as their keeper or owner.
The respective general fundamental relationship "*Place has met Thing"* is:

E53.Place--(P89B.contains) [0,n] -> **E53.Place:**
{E53.Place-- P53B.is_former_or_current_location_of -> **E24.Physical_Man-Made_Thing** :
{ E24.Physical_Man-Made_Thing -- (P46F.is_composed_of) [0,n] ->
E24.Physical_Man-Made_Thing[--P2F.has_type -> **E55.Type**]
}
OR
E53.Place --P7B.witnessed->**E5.Event:**
{E5.Event --(P9F.consists_of)[0,n]-> **E5.Event:**
{E5.Event-- P92F.brought_into_existence -> **E70.Thing:**
{ E70.Thing --(F4B.is_component_of)[0,n]-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
OR
E11.Modification-- P31F.has_modified -> **E70.Thing:**

{ **E70.Thing** --(F4B.is_component_of)[0,n]-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
OR
E9.Move -- P25F.moved -> **E19.Physical_Object:**
{ **E19.Physical_Object** --(P46B.forms_part_of)[0,n] -> **E19.Physical_Object**[--P2F.has_type -> **E55.Type]**
}
}
}
OR
E53.Place-- P74B.is_current_or_former_residence_of -> **E39.Actor:**
{E39.Actor-- {P49B.is_former_or_current_keeper_of OR P51B.is_former_or_current_owner_of**} -> E18.Physical_Thing:**
{**E18.Physical_Thing** --(P46B.forms_part_of)[0,n] -> **E18.Physical_Thing** [--P2F.has_type -> **E55.Type]**
}
}
}

Specializations:
The user frequently may only be interested in things that are products of a place so a specialization of the former relationship is:

- is_origin_of

This is a specialization, for queries asking for the place where an instance of the category Thing was created.
The respective specialized fundamental relationship "*Place is origin of Thing"* is:

E53.Place--(P89B.contains)[0,n] -> **E53.Place:**
{E53.Place --P7B.witnessed->**E5.Event:**
{E5.Event --(P9F.consists_of)[0,n]-> **E5.Event:**
{E5.Event--P92F.brought_into_existence-> **E70.Thing:**
{ **E70.Thing** --(F4B.is_component_of)[0,n]-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
}
}
}

- is location of

This is a specialization, for queries asking for the place where an instance of the category Thing is or was once located.
E53.Place--(P89B.contains)[0,n] -> **E53.Place:**
{E53.Place-- P53B.is_former_or_current_location_of -> **E24.Physical_Man-Made_Thing** :
{ **E24.Physical_Man-Made_Thing** -- (P46F.is_composed_of)[0,n] -> **E24.Physical_Man-Made_Thing**[--P2F.has_type -> **E55.Type**]
}
}

Place-Actor

a. refers to

A place can refer to an actor by artefacts created or located in that place that have as subject or depict the actor. So with this relationship we can find out about the Places where an Actor is referred to at.

In CIDOC-CRM properties that define the "refer to" or "is about" relationship are :

- P62F.depicts
- P67F.refers_to

The respective general fundamental relationship "*Place refers to Actor"* is:

E53.Place--(P89B.contains)[0,n] -> **E53.Place:**
{E53.Place-- P53B.is_former_or_current_location_of -> **E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing -- P62F.depicts ->**E39.Actor:**
{E39.Actor --(P107F.has_current_or_former_member)[0,n]->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing --P128F.carries-> **E73.Information_Object:**
{E73.Information_Object -- P67F.refers_to->**E39.Actor:**
{E39.Actor --(P107F.has_current_or_former_member)[0,n]->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
}
}
OR
E53.Place-- P7B.witnessed-> **E5.Event:**
{ E5.Event --(P9F.consists_of)[0,n]-> **E5.Event:**
{E65.Creation -- P94F.has_created-> **E89.Propositional_Object:**
{ E89.Propositional_Object --P67F.refers_to -> **E39.Actor:**
{E39.Actor --**(**P107F.has_current_or_former_member)[0,n]->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
}

OR
E12.Production -- P108F.has_produced -> **E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing -- P62F.depicts ->**E39.Actor:**
{E39.Actor --**(**P107F.has_current_or_former_member)[0,n]->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing --P128F.carries->
E73.Information_Object:
{E73.Information_Object -- P67F.refers_to ->**E39.Actor:**
{E39.Actor --
(P107F.has_current_or_former_member)[0,n]->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
}
}
}
}

b. is referred to by

A Place may be referred to by Actors though artifacts that they create and have as a theme or subject or refer to the Place.
In CIDOC-CRM properties that define the "is referred to by" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Place is referred to by Actor"* is:

E53.Place-- (P89F.falls_within)[0,n] -> **E53.Place:**
{E53.Place --P67B.is_referred_to_by-> **E89.Propositional_Object:**
{E89.Propositional_Object -- P94B.was_created_by ->**E65.Creation**:
{E65.Creation -- (P9B.forms_part_of)[0,n]-> **E7.Activity:**

{E7.Activity -- P14F.carried_out_by->**E39.Actor:**
{E39.Actor --
(P107F.has_current_or_former_member)[0,n]->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
}
}
OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] ->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing -- P108B.was_produced_by ->
E12.Production:
{ E12.Production -- (P9B.forms_part_of)[0,n]-> **E5.Event:**
{E7.Activity -- P14F.carried_out_by->
E39.Actor:
{E39.Actor --
(P107F.has_current_or_former_member
)[0,n]->
E39.Actor [--P2F.has_type ->
E55.Type]
}
}
}
}
}
}
OR
E53.Place -- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing**:
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of)[0,n] ->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing -- P108B.was_produced _by ->
E12.Production:
{E12.Production -- (P9B.forms_part_of)[0,n]-> **E5.Event:**
{E7.Activity -- P14F.carried_out_by->**E39.Actor:**
{E39.Actor --
(P107F.has_current_or_former_member)[0,n]-
>**E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}
}

c. has met

This relationship is used to connect a place with persons or groups of people that are born (formed) in a place or have had the place as residence. Also when an event that is witnessed by an Actor takes place at a certain Place, then we deduct that also the Actor has met the Place.
In CIDOC-CRM properties that define the "has met" relationship are :

- P74B.is_current_or_former_residence_of
- P7B.witnessed

The respective general fundamental relationship "*Place has met Actor"* is:

E53.Place--(P89B.contains)[0,n] -> **E53.Place:**
{E53.Place -- P74B.is_current_or_former_residence_of-> **E39.Actor:**
{E39.Actor --(P107F.has_current_or_former_member)[0,n]->**E39.Actor** [--
P2F.has_type -> **E55.Type**]
}

OR
E53.Place --P7B.witnessed->**E5.Event:**
{E5.Event --(P9F.consists_of)[0,n]-> **E5.Event:**
{E5.Event-- P12F.occurred_in_the_presence_of ->**E39.Actor:**
{E39.Actor --(P107B.is_current_or_former_member)[0,n]-> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
}

Specializations:
The user frequently may only be interested in actors (individuals or groups) that originate from a place so a specialization of the former relationship is:

- is_origin_of

This specialization is specified for places that are the birth or formation places of instances of the category Actor.
The respective specialized fundamental relationship "*Place is origin of Actor"* is:

E53.Place--(P89B.contains)[0,n] -> **E53.Place:**
{E53.Place --P7B.witnessed->**E5.Event:**
{E5.Event --(P9F.consists_of)[0,n] -> **E5.Event:**
{E63.Beginning_of_Existence-- P92F.brought_into_existence ->**E39.Actor:**
{E39.Actor --(P107F.has_current_or_former_member)[0,n]-> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}

Place-Event

The Place-Event fundamental relationships can be switched to Thing-Time fundamental relationships, by further adding the CIDOC-CRM property P4F.has_time-span at the range category Event **(E5.Event** -- P4F.has_time-span->**E52.Time-Span)**. This happens because Time refers to the chronological definition of Events.

a. has met

This relationship is used to show in what Places an Event took Place. An event may take place in various places, if we consider that the sub-events possibly take place in other places. We consider that the main event occurs in all the places where its sub-events take place.
The respective specialized fundamental relationship "*Place has met Event"* is:

E53.Place-- (P89B.contains)[0,n] -> **E53.Place:**
{E53.Place--P7B.witnessed-> **E5.Event:**
{E5.Event -- (P9B.forms_part_of)[0,n]->**E5.Event** [--P2F.has_type -> ***E55.Type***]
}
}

b. refers to

A Place can refer to an Event through objects that have as theme or depict an Event and are or were located in the Place or through people that are at a Place and that mention or refer to an Event.
In CIDOC-CRM properties that define the "refer to" or "is about" relationship are :

- P62F.depicts
- P67F.refers_to

The respective general fundamental relationship "*Place refers to Event"* is:

E53.Place--(P89B.contains)[0,n]-> **E53.Place:**
{E53.Place-- P53B.is_former_or_current_location_of -> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- (P46F.is_composed_of) [0,n] ->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing -- P62F.depicts-> **E5.Event**:
{ E5.Event --(P9F.consists_of)[0,n]-> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing--P128F.carries-> **E73.Information_Object:**
{E73.Information_Object --P67F.refers_to -> **E5.Event**:
{E5.Event --(P9F.consists_of)[0,n]-> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}

c. is referred to at

A Place may be referred by Events during which Things that refer to the Place are created.
In CIDOC-CRM properties that define the "is referred to at" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Place is referred to at Event"* is:

E53.Place--(P89F.falls_within)[0,n]-> **E53.Place:**
{ E53.Place --P67B.is_referred_to_by -> **E89.Propositional_Object:**
{E89.Propositional_Object -- (F5B.forms_part_of_shows_features_of)[0,n]->
E89.Propositional_Object:
{E89.Propositional_Object -- P94B.was_created_by -> **E65.Creation:**
{ E65.Creation -- (P9B.forms_part_of)[0,n] -> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing -- (F5B.forms_part_of_shows_features_of)[0,n]-> **E24.Physical_Man-Made_Thing**:
{ E24.Physical_Man-Made_Thing--P108B.was_produced_by -> **E12.Production:**
{ E12.Production -- (P9B.forms_part_of)[0,n] -> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}
OR
E53.Place -- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing --(F5B.forms_part_of_shows_features_of)[0,n]->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing--P108B.was_produced_by -> **E12.Production:**
{ E12.Production -- (P9B.forms_part_of) [0,n] -> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
}
}

}

Place-Time

The Place-Event fundamental relationships can be switched to Thing-Time fundamental relationships, by further adding the CIDOC-CRM property P4F.has_time-span at the range category Event **(E5.Event** -- P4F.has_time-span->**E52.Time-Span)**. This happens because Time refers to the chronological definition of Events.

a. has met

This relationship is used to show in what Places Events took place during a certain period of time. An event may take place in various places, if we consider that the sub-events possibly take place in other places. We consider that the main event occurs in all the places where its sub-events take place.
The respective specialized fundamental relationship "*Place has met Time"* is:

E53.Place-- (P89B.contains)$^{[0,n]}$ -> **E53.Place:**
{E53.Place--P7B.witnessed-> **E5.Event:**
{E5.Event -- (P9B.forms_part_of)$^{[0,n]}$-> **E5.Event:**
{E5.Event-- P4F.has_time-span-> **E52.Time-Span:**
{E52.Time-Span --(P86F.falls_within)$^{[0,n]}$-> **E52.Time-Span**[--
P2F.has_type -> **E55.Type**]
}
}
}
}

Place-Concept

a. has type

This relationship connects a place with its type. A place may belong to more than one type categories.
The respective fundamental relationship "*Place has type Concept"* is:
E53.Place -- P2F.has_type ->**E55.Type**:
{ **E55.Type** -- (P127F.has_broader_term)$^{[0,n]}$ -> **E55.Type**
}

ACTOR

The Actor category comprises people, either individually or in groups, who have the potential to perform intentional actions for which they can be held responsible. Since people may be members of groups and groups can be members of other groups, the following relations can be applied at the beginning of any of the fundamental relationships. So it is displayed here as root of the relationships path-chain.

E39.Actor--(P107B.is_current_or_former_member_of)[0,n]->
E74.Group
OR
E74.Group--(P107F.has_current_or_former_member)[0,n]->
E39.Actor

Actor-Place

a. refers to

An actor may refer to a Place by Things they create and refer to or have as subject a Place.
The respective fundamental relationship "*Actor refers to Place"* is:

E39.Actor --(P107F.has_current_or_former_member)[0,n]->**E39.Actor:**
{E39.Actor -- P14B.performed -> **E7.Activity:**
{E7.Activity -- (P9F.consists_of)[0,n]-> **E5.Event:**
{E7.Activity--P92F.brought_into_existence -> **E70.Thing:**
{**E70.Thing**-- (F5F.consists_of_shows_features_of)[0,n]->
E70.Thing:
{E24.Physical_Man-Made_Thing -- P62F.depicts ->
E53.Place:
{E53.Place --(P89B.contains)[0,n]-> **E53.Place**
[--P2F.has_type -> **E55.Type**] }
OR
E89.Propositional_Object --P67F.refers_to-> **E53.Place:**
{E53.Place --(P89B.contains)**[0,n]-> E53.Place**
[--P2F.has_type -> **E55.Type**] }
OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{E73.Information_Object -- P67F.refers_to ->
E53.Place:
{E53.Place --(P89B.contains)**[0,n]->**
E53.Place [--P2F.has_type ->
E55.Type]
}
}
}
}
}
}
}

b. is referred to at

There are cases that one is interested in Actors that are referred to at a certain Place. Reference implies the presence or creation of a Thing that refers to the Actor in interest, as even speech is a human product, thus a Thing. So, when we talk about something, write about something or in any other way mention something the means we do it is mapped to a Thing that we create.
In this manner we connect the Thing and Place FCs not only directly but also through other Things and through Events.
In CIDOC-CRM properties that define the "is referred to at" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Actor is referred to at Place"* is:

E39.Actor--(P107B.is_current_or_former_member_of)[0,n]-> **E39.Actor:**
{E39.Actor -- P67B.is_referred_to_by -> **E89.Propositional_Object:**
{E73.Information_Object -- P94B.was_created_by -> **E65.Creation**:
{E65.Creation --(P9B.forms_part_of)[0,n] ->**E5.Event:**
{E5.Event -- P7F.took_place_at ->**E53.Place:**
{E53.Place --(P89F.falls_within)**[0,n]**-> **E53.Place**
[--P2F.has_type -> **E55.Type**]
}
}
}
OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] ->
E18.Physical_Thing:
{**E18.Physical_Thing --** P53F.has_former_or_current_location ->
E53.Place :
{E53.Place --(P89F.falls_within)**[0,n]**-> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
}
}
OR
E39.Actor -- P62B.is_depicted_by -> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] -> **E24.Physical_Man-Made_Thing**:
{**E24.Physical_Man-Made_Thing** -- P53F.has_former_or_current_location ->
E53.Place:
{E53.Place --(P89F.falls_within)**[0,n]**-> **E53.Place**
[--P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P108B.was_produced_by ->
E12.Production:
{ E12.Production --(P9B.forms_part_of)[0,n]-> **E5.Event:**
{ E5.Event -- P7F.took_place_at ->**E53.Place**:
{E53.Place --(P89F.falls_within)**[0,n]**-> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}
}

c. has met

This is a general relationship that connects an Actor with a place they have been at. This can also happen through an event.
The respective fundamental relationship "*Actor has met Place"* is:

E39.Actor --(P107F.has_current_or_former_member)[0,n]-> **E39.Actor:**
{E39.Actor--P12B.was_present_at-> **E5.Event:**
{E5.Event--(P9B.forms_part_of)[0,n]-> **E5.Event:**
{E5.Event --P7F.took_place_at-> **E53.Place**:
{E53.Place --(P89F.falls_within)**[0,n]**-> **E53.Place** [--P2F.has_type -> **E55.Type**]
}

}
}
OR
E39.Actor -- P74F.has_current_or_former_residence **-> E53.Place**:
{E53.Place --(P89F.falls_within**)[0,n]-> E53.Place** [--P2F.has_type -> **E55.Type**]
}
}

d. from

An actor is from a place usually when their residence is located there, or they have been born there. The respective fundamental relationship "*Actor from Place"* is:

E39.Actor--(P107B.is_current_or_former_member_of)[0,n]-> **E39.Actor:**

{E39.Actor-- P74F.has_current_or_former_residence-> **E53.Place**:
{E53.Place --(P89F.falls_within**)[0,n]-> E53.Place** [--P2F.has_type -> **E55.Type**]
}
OR
E39.Actor -- P92B.was_brought_into_existence_by -> **E63.Beginning_of_Existence**:
{E63.Beginning_of_Existence --(P9B.forms_part_of)[0,n]-> **E5.Event:**
{E5.Event --P7F.took_place_at->**E53.Place**:
{E53.Place --(P89F.falls_within**)[0,n]-> E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
}
}

Actor-Thing

a. refers to

An actor may refer to a Thing by Things they create and refer to or have as subject a Thing. The respective fundamental relationship "*Actor refers to Thing"* is:

E39.Actor --(P107F.has_current_or_former_member**)[0,n]->E39.Actor:**
{E39.Actor -- P14B.performed -> **E7.Activity:**
{E7.Activity -- (P9F.consists_of)[0,n]-> **E7.Activity:**
{E7.Activity--P92F.brought_into_existence -> **E70.Thing:**
{**E24.Physical_Man-Made_Thing** --
(F5F.consists_of_shows_features_of)[0,n]-> **E24.Physical_Man-Made_Thing**:
{E24.Physical_Man-Made_Thing -- P62F.depicts -> **E70.Thing:**
{E70.Thing -- (F4F.is_composed_of)[0,n] -> **E70.Thing**
[-- P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries -> **E73.Information_Object:**
{E73.Information_Object -- P67F.refers_to -> **E70.Thing:**
{E70.Thing -- (F4F.is_composed_of)[0,n] -> **E70.Thing**
[-- P2F.has_type -> **E55.Type**]
}
}
}
}
}

}

b. is referred to by

An Actor may be referred to by Things that have as theme or subject the Actor, or that refer to or are about the Actor.
In CIDOC-CRM properties that define the "is referred to by" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Actor is referred to by Thing"* is:

E39.Actor -- (P107B.is_current_or_former_member_of)[0,n] -> **E39.Actor:**
{E39.Actor --P67B.is_referred_to_by -> **E89.Propositional_Object:**
{E89.Propositional_Object -- (P148B.is_component_of)[0,n] ->
E89.Propositional_Object:
{ E89.Propositional_Object[--P2F.has_type -> **E55.Type**]
OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) 0,n) ->
E24.Physical_Man-Made_Thing[--P2F.has_type -> **E55.Type**]
}
}
}
OR
E39.Actor -- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing**:
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) 0,n) ->
E24.Physical_Man-Made_Thing[--P2F.has_type -> **E55.Type**]
}
}

c. is origin of

Things are created by actors and are kept or owned by them. So, actors can serve as creators or owners of Things. These properties can be unified into one fundamental relationship that shows the origin of a Thing. This is the reverse relation of *Thing from Actor*.
The respective fundamental relationship "*Actor is origin of Thing"* is:

E39.Actor --(P107F.has_current_or_former_member)[0,n]->**E39.Actor:**
{E39.Actor -- P14B.performed -> **E7.Activity:**
{E7.Activity -- (P9F.consists_of)[0,n]-> **E5.Event:**
{E63.Beginning_of_Existence --P92F.brought_into_existence ->
E70.Thing:
{E70.Thing -- (F4F.is_composed_of)[0,n]-> **E70.Thing** [--
P2F.has_type -> **E55.Type**]
}
}
}
OR
E39.Actor --{ P49B.is_former_or_current_keeper_of OR
P51B.is_former_or_current_owner_of }-> **E18.Physical_Thing:**
{E18.Physical_Thing -- (P46F.is_composed_of) [0,n] ->
E18.Physical_Thing [--P2F.has_type -> **E55.Type**]
}
}

Specializations:

In the "*is origin of"* FR we can define specializations to distinguish between the creator of one Thing and the owner /keeper of it. So, we have:

- is generator of

This specialization is specified to return actors that have created the given by the query parameter instances of the category Thing.
The respective fundamental relationship "*Actor is generator of Thing"* is:

E39.Actor --(P107F.has_current_or_former_member)[0,n]->**E39.Actor:**
{E39.Actor -- P14B.performed -> **E7.Activity:**
{E7.Activity -- (P9F.consists_of)[0,n]-> **E5.Event:**
{E63.Beginning_of_Existence --P92F.brought_into_existence **->**
E70.Thing:
{E70.Thing -- (F4F.is_composed_of)[0,n]-> **E70.Thing** [--
P2F.has_type -> **E55.Type**]
}
}
}
}

- has

An actor can have a Thing under their possession or keep.
The respective fundamental relationship "*Actor has Thing"* is:

E39.Actor --(P107F.has_current_or_former_member)[0,n]->**E39.Actor:**
{E39.Actor --{ P49B.is_former_or_current_keeper_of OR
P51B.is_former_or_current_owner_of }-> **E18.Physical_Thing:**
{E18.Physical_Thing -- (P46F.is_composed_of) [0,n] ->
E18.Physical_Thing [--P2F.has_type -> **E55.Type**]
}
}

d. has met

This is a general relationship connecting an Actor with Things that they have met (been at the same place the same time) at least once. This is possible when the actor and the Thing were present at the same Event. Nevertheless a Thing may be documented to be present to a certain Event and so may be an Actor, but this may not necessarily mark that the two entities have met each other. This may happen when we are talking about a big event, such as the French Revolution, that takes place in various areas in the same or different time sub-periods. However, since here we are interested in high recall rates we prefer to include such uncertainties than to exclude valid cases.
The respective fundamental relationship "*Actor has met Thing"* is:

E39.Actor--P12B.was_present_at-> **E5.Event:**
{ E5.Event -- (P9F.consists_of)[0,n]-> **E5.Event:**
{E5.Event-- P12F.occurred_in_the_presence_of ->
E70.Thing[--P2F.has_type -> **E55.Type**]
}
}

Actor-Actor

a. has met

This is a general relationship connecting an Actor with Actors that they have met (been at the same place the same time) at least once. This is possible when the actors were present at the same Event. Nevertheless two actors may be documented to be present to a certain Event, but this may not necessarily mark that the two entities have met each other. This may happen when we are talking about a big event, such as the French Revolution, that takes place in various areas in the same or different time sub-periods. However, since here we are interested in high recall rates we prefer to include such uncertainties than to exclude valid cases.

The respective fundamental relationship "*Actor has met Actor"* is:

E39.Actor --(P107F.has_current_or_former_member)[0,n]-> **E39.Actor:**
{E39.Actor--P12B.was_present_at-> **E5.Event:**
{E5.Event-- (P9F.consists_of)[0,n] -> **E5.Event:**
{E5.Event --P12F.occurred_in_the_presence_of-> **E39.Actor:**
{ E39.Actor--(P107B.is_current_or_former_member_of)[0,n]->
E39.Actor[--P2F.has_type -> **E55.Type**]
}
}
}
}

b. refers to

An actor can refer to another actor by Things that they create and refer to another Actor.
The respective fundamental relationship "*Actor refers to Actor"* is:

E39.Actor --(P107F.has_current_or_former_member)[0,n]-> **E39.Actor:**
{E39.Actor -- P14B.performed -> **E7.Activity:**
{E7.Activity -- (P9F.consists_of)[0,n]-> **E5.Event:**
{E63.Beginning_of_Existence --{P92F.brought_into_existence**} ->**
E70.Thing:
{**E70.Thing**-- (F5F.consists_of_shows_features_of)[0,n]->
E70.Thing:
{E24.Physical_Man-Made_Thing -- P62F.depicts **->**
E39.Actor:
{E39.Actor--
(P107F.has_current_or_former_member)[0,n] ->
E39.Actor[--P2F.has_type -> **E55.Type**]
}
OR
E89.Propositional_Object --P67F.refers_to-> **E39.Actor:**
{E39.Actor--
(P107F.has_current_or_former_member)[0,n]->
E39.Actor[--P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{E73.Information_Object --P67F.refers_to ->
E39.Actor:
{E39.Actor--
(P107F.has_current_or_former_member
)[0,n]-> **E39.Actor**[--P2F.has_type ->
E55.Type]
}
}
}
}
}
}
}

c. is referred to by

An Actor may be referred to by Actors though artifacts that they create and have as a theme or subject or refer to the Actor.
In CIDOC-CRM properties that define the "is referred to by" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Actor is referred to by Actor"* is:

E39.Actor -- (P107B.is_current_or_former_member_of)[0,n] -> **E39.Actor:**
{E39.Actor --P67B.is_referred_to_by-> **E89.Propositional_Object:**
{E89.Propositional_Object -- P94B.was_created_by ->**E65.Creation**:
{E65.Creation -- (P9B.forms_part_of)[0,n]-> **E7.Activity:**
{E7.Activity -- P14F.carried_out_by-> **E39.Actor:**
{E39.Actor --(P107B.is_current_or_former_member_of)[0,n] ->**E39.Actor** [--P2F.has_type ->**E55.Type**]
}
}
}
OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) 0,n) ->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing -- P108B.was_produced _by ->
E12.Production:
{ E12.Production -- (P9B.forms_part_of)[0,n]->
E7.Activity:
{E7.Activity -- P14F.carried_out_by->
E39.Actor:
{E39.Actor --
(P107B.is_current_or_former_member_of) [0,n] ->**E39.Actor** [--P2F.has_type ->**E55.Type**]
}
}
}
}
}
OR
E39.Actor -- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing**:
{E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] ->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing -- P108B.was_produced _by ->
E12.Production:
{ E12.Production -- (P9B.forms_part_of)[0,n]-> **E7.Activity:**
{E7.Activity -- P14F.carried_out_by-> **E39.Actor:**
{E39.Actor --
(P107B.is_current_or_former_member_of) [0,n] ->**E39.Actor** [--P2F.has_type ->**E55.Type**]
}
}
}
}
}

d. from

This relationship connects an Actor with their generators. Generator may mean parent if we are talking about individuals or founder if we are talking about groups of people.
The respective fundamental relationship "*Actor from Actor"* is:

E39.Actor -- (P107B.is_current_or_former_member_of)[0,n] -> **E39.Actor:**
{E21.Person--P98B.was_born -> **E67.Birth:**
{E67.Birth--{P97F.from_father OR P96F.by_mother**}->E21.Person**[--P2F.has_type -> **E55.Type**]
}
OR

E74.Group -- P95B.was_formed_by -> **E66.Formation :**
{E66.Formation-- (P9F.consists_of)[0,n]-> **E7.Activity:**
{ E7.Activity--P14F.carried_out_by **E39.Actor:**
{E39.Actor --(P107B.is_current_or_former_member_of) [0,n] -> **E39.Actor** [--P2F.has_type ->**E55.Type**]
}
}
}
}

e. is origin of

This is the reverse relationship of from and as such it returns the children or the groups one Actor has created.
The respective fundamental relationship "*Actor is oringin of Actor"* is:

E39.Actor-- (P107F.has_current_or_former_member)[0,n]-> **E39.Actor:**
{**E21.Person** -- {P97B.was_father_for OR P96B.gave_birth }-> **E67.Birth:**
{E67.Birth--P98F.brought_into_life->**E21.Person**[--P2F.has_type -> **E55.Type**]
}
OR
E39.Actor --P14B.performed -> **E7.Activity**:
{**E7.Activity** -- (P9B.forms_part_of)[0,n]-> **E5.Event:**
{**E66.Formation** -- P95F.has_formed -> **E39.Actor:**
{E39.Actor-- (P107F.has_current_or_former_member)[0,n]-> **E39.Actor**[--P2F.has_type -> **E55.Type**]
}
}
}
}

f. has member

With this FR the groups to which an actor belongs to are returned. This relationship between groups and actors can be documented either explicitly or through a Joining Event,
The respective fundamental relationship "*Actor has member Actor"* is:

E74.Group--(P107F.has_current_or_former_member)[0,n]-> **E39.Actor :**
{E39.Actor[--P2F.has_type -> **E55.Type**]
OR
E74.Group --P144B.gained_member_by -> **E85.Joining:**
{E85.Joining --P143F.joined -> **E39.Actor:**
{E39.Actor-- (P107F.has_current_or_former_member)[0,n]-> **E39.Actor**[--P2F.has_type -> **E55.Type**]
}
}
}

g. is member of

This is the reverse link of the *has member* relationship. It returns the members of a certain group.
The respective fundamental relationship "*Actor is member of Actor"* is:

E39.Actor-- **(**P107B.is_current_or_former_member_of)[0,n]-> **E39.Actor :**
{**E74.Group** [--P2F.has_type -> **E55.Type**]
OR
E39.Actor -- P143B.was_joined_by -> **E85.Joining:**
{ E85.Joining -- P144F.joined_with -> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)[0,n]-> **E39.Actor**[--P2F.has_type -> **E55.Type**]
}
}
}

Actor-Event

a. refers to

An actor may refer to an Event through Things they produce or create and which Things have as subject or refer to the Event.
The respective fundamental relationship "*Actor refers to Event"* is:

E39.Actor --(P107F.has_current_or_former_member)[0,n]-> **E39.Actor:**
{E39.Actor -- P14B.performed -> **E7.Activity:**
{E7.Activity -- (P9F.consists_of)[0,n]-> **E5.Event:**
{E63.Beginning_of_Existence --P92F.brought_into_existence **->**
E70.Thing:
{E70.Thing -- (F5F.consists_of_shows_features_of)[0,n]->
E70.Thing:
{E24.Physical_Man-Made_Thing -- P62F.depicts **->**
E5.Event:
{E5.Event-- (P9F.consists_of)[0,n]->
E5.Event [-- P2F.has_type -> **E55.Type**]
}
OR
E89.Propositional_Object --P67F.refers_to -> **E5.Event:**
{E5.Event-- (P9F.consists_of)[0,n]->
E5.Event [-- P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{E73.Information_Object -- P67F.refers_to ->
E5.Event:
{E5.Event-- (P9F.consists_of)[0,n]->
E5.Event [-- P2F.has_type ->
E55.Type]
}

}
}
}
}
}
}

b. is referred to at

An Actor may be referred by Events during which Things that refer to the Actor are created.
In CIDOC-CRM properties that define the "is referred to at" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Actor is referred to at Event"* is:

E39.Actor -- (P107B.is_current_or_former_member_of)[0,n] -> **E39.Actor:**
{E39.Actor -- P67B.is_referred_to_by -> **E89.Propositional_Object:**
{E89.Propositional_Object -- P94B.was_created_by -> **E65.Creation:**
{E65.Creation -- (P9B.forms_part_of) [0,n] -> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] ->
E24.Physical_Man-Made_Thing:

{E24.Physical_Man-Made_Thing--P108B.was_produced_by ->
E12.Production:
{E12.Production -- (P9B.forms_part_of) [0,n] -> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
}
}
}
OR
E39.Actor -- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] ->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing--P108B.was_produced_by ->
E12.Production:
{E12.Production -- (P9B.forms_part_of) [0,n] -> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
}
}
}

c. from

This relationship returns all the Actors that are brought into existence, so have origin in a specific Event. We assume a group and the member-groups are formed concurrently. Here we have to distinguish the cases between groups and persons as it would be a mistake that would diminish precision if we would say that individual members of groups were brought into existence at the same event as the group's formation. Eg John that is a member of FORTH institute was not born the same day as FORTH's establishment.
The respective specialized fundamental relationship "*Actor from Event"* is:

E39.Actor--(P107B.is_current_or_former_member)[0,n]-> **E39.Actor:**
{E39.Actor--{P95.was_formed_by OR P98B.was_born **} -> E63.Beginning_of_Existence:**
{ E63.Beginning_of_Existence --(P9B.forms_part_of)[0,n] ->**E5.Event** [--P2F.has_type -> **E55.Type**]
}
}

d. has met

This is a general relationship that connects an Actor with an event they have been present at. It does not specify the role of the Actor in the Event. For more specification specializations on this FR are provided afterwards.
The respective fundamental relationship "*Actor has met Event"* is:

E39.Actor --(P107F.has_current_or_former_member)[0,n]-> **E39.Actor:**
{E39.Actor--P12B.was_present_at-> **E5.Event:**
{E5.Event--(P9B.forms_part_of)[0,n] ->**E5.Event** [--P2F.has_type -> **E55.Type**]
}
}

Specializations:

In the fundamental relationship *has met* we can also define some sub-fundamental relationships, which contain more restricted information. This information is however commonly asked for by the users and a more specialized query may return better results precision-wisely based to what they actually want to know.

- Performed action at

This relationship returns all the Actors that performed some kind of action in an Event.

The respective specialized fundamental relationship "*Actor performed action at Event"* is:

E39.Actor --(P107F.has_current_or_former_member)[0,n]-> **E39.Actor:**
{E39.Actor-- P14B.performed->**E7.Activity:**
{ **E7.Activity--(**P9B.forms_part_of)[0,n] ->**E5.Event** [--P2F.has_type -> **E55.Type**]
}
}

Actor-Time

The Actor-Event fundamental relationships can be switched to Actor-Time fundamental relationships, by further adding the CIDOC-CRM property P4F.has_time-span at the range category Event **(E5.Event** --P4F.has_time-span->**E52.Time-Span)**. This happens because Time refers to the chronological definition of Events.

a. refers to

An actor may refer to a Time span through Things they produce or create and which Things have as subject or refer to that Time.
The respective fundamental relationship "*Actor refers to Time"* is:

E39.Actor --(P107F.has_current_or_former_member)[0,n]-> **E39.Actor:**
{E39.Actor -- P14B.performed -> **E7.Activity:**
{E7.Activity -- (P9F.consists_of)[0,n]-> **E5.Event:**
{E63.Beginning_of_Existence --P92F.brought_into_existence **->**
E70.Thing:
{E70.Thing -- (F5F.consists_of_shows_features_of)[0,n]->
E70.Thing:
{E24.Physical_Man-Made_Thing -- P62F.depicts **->**
E52.Time-Span:
{E52.Time-Span --(P86B.contains)[0,n]->
E52.Time-Span[-- P2F.has_type -> **E55.Type**]
}

OR
E89.Propositional_Object --P67F.refers_to ->
E52.Time-Span:
{E52.Time-Span --(P86B.contains)[0,n]->
E52.Time-Span[-- P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{E73.Information_Object -- P67F.refers_to **->**
E52.Time-Span:
{E52.Time-Span --(P86B.contains)[0,n]-> **E52.Time-Span**[-- P2F.has_type -> **E55.Type**]
}
}
}
}
}
}
}

b. from

This relationship returns all the Actors that are brought into existence in a specific Time. We assume a group and the member-groups are formed concurrently. Here we have to distinguish the cases between groups and persons as it would be a mistake that would diminish precision if we would say that

individual members of groups were brought into existence at the same event as the group's formation. Eg John that is a member of FORTH institute was not born the same day as FORTH's establishment. The respective specialized fundamental relationship "*Actor from Time"* is:

E39.Actor--(P107B.is_current_or_former_member_of)[0,n]-> **E39.Actor:**
{E39.Actor--{P95.was_formed_by OR P98B.was_born } **-> E63.Beginning_of_Existence:**
{ E63.Beginning_of_Existence --(P9B.forms_part_of)[0,n] -> **E5.Event:**
{E5.Event-- P4F.has_time-span-> **E52.Time-Span:**
{E52.Time-Span --(P86F.falls_within)[0,n]-> **E52.Time-Span**[--
P2F.has_type -> **E55.Type**]
}
}
}
}

c. has met

This is a general relationship that connects an Actor with some Time they have been present in. It does not specify the role of the Actor in that time. For more specification specializations on this FR are provided afterwards.
The respective fundamental relationship "*Actor has met Event"* is:

E39.Actor --(P107F.has_current_or_former_member)[0,n]-> **E39.Actor:**
{E39.Actor--P12B.was_present_at-> **E5.Event:**
{E5.Event--(P9B.forms_part_of)[0,n] -> **E5.Event:**
{E5.Event-- P4F.has_time-span-> **E52.Time-Span:**
{E52.Time-Span --(P86F.falls_within)[0,n]-> **E52.Time-Span**[--
P2F.has_type -> **E55.Type**]
}
}
}
}

Specializations:

In the fundamental relationship *has met* we can also define some sub-fundamental relationships, which contain more restricted information. This information is however commonly asked for by the users and a more specialized query may return better results precision-wisely based to what they actually want to know.

- Performed action at

This relationship returns all the Actors that performed some kind of action in an specific time span.

The respective specialized fundamental relationship "*Actor performed action at Time"* is:

E39.Actor --(P107F.has_current_or_former_member)[0,n]-> **E39.Actor:**
{E39.Actor-- P14B.performed->**E7.Activity:**
{ **E7.Activity--(**P9B.forms_part_of)[0,n] ->**E5.Event:**
{E5.Event-- P4F.has_time-span-> **E52.Time-Span:**
{E52.Time-Span --(P86F.falls_within)[0,n]-> **E52.Time-Span**[--
P2F.has_type -> **E55.Type**]
}
}
}
}

Actor-Concept

a. has type

This relationship connects an Actor with its type, which describes its concept. An Actor may belong to more than one type categories. Also by including the types of the members, we actually mean that the wider Actor "includes" the members' types.
The respective fundamental relationship "*Actor has type Concept"* is:

E39.Actor --(P107F.has_current_or_former_member)$^{[0,n]}$-> **E39.Actor:**
{E39.Actor-- P2F.has_type-> **E55.Type**:
{ **E55.Type** -- {(P127F.has_broader_term)$^{[0,n]}$ OR (P2F.has_type)$^{[0,n]}$}-> **E55.Type**
[P2F.has_type -> **E55.Type**]
}
}

EVENT

The Event fundamental Category is indeed an important one. In the context of an event, actors, things, places and other events or time periods are included answering the 4 main questions WHO, WHEN, WHERE, WHAT. So, as the metadata of Events are potentially rich in information also regarding the other FCs, it becomes a considerable "link" between different FCs. But not only is it used for intermediate linking of Categories, but as a domain or range Category of one FR as well. In this set of FRs the Event category is used as Domain.
Time can be interpreted as the placement of the Event in Time and as such it can be used in the same way as the Event FC, only by adding the *is time-span of* property before the Event domain Category. That is:
E52.Time-Span-- P4B.is_time-span_of ->**E5.Event**
It is also decided that E5.Event is going to be defined as a subclass of the class E52.Time-Span. Using the CIDOC-CRM and the CIDOC-CRMdig, it is a common practice that times are defined as rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime" properties of the E5.Event class that they define, instead of using the property P4F.has_time-span to connect instances of the two classes. For this reason, when querying about the Time FC, it would be safe to also include to the query, instances of the FC Event. Then, before displaying the results to the user on the interface, we connect the instances of the Event results to their time-span using the rdf property: rdf:data-time, which time is declared in the metadata as *literal*. This is also why we cannot include such results to the result set, when asking for instances of the E52.Time-Span.

Event-Place

a. refers to

An event may refer to a Place by Things that are created or produced during this Event and refer to or depict the Place. A simple existence in an Event of a Thing that refers to a Place can not deduct that the Event also refers to the Place. Moreover we could say that an Event refers to the Place where this event took place, such as with the French Revolution that refers to France and also took place in France.
The respective general fundamental relationship "*Event refers to Place"* is:

E5.Event -- (P9F.consists_of)$^{[0,n]}$-> **E5.Event:**
{E63.Beginning_of_Existence --P92F.brought_into_existence -> **E70.Thing:**
{**E70.Thing**-- (F5F.consists_of_shows_features_of)$^{[0,n]}$ -> **E70.Thing**:
{E24.Physical_Man-Made_Thing -- P62F.depicts -> **E53.Place:**
{E53.Place --(P89B.contains)$^{[0,n]}$ -> **E53.Place** [--
P2F.has_type -> **E55.Type**]
}
OR
E89.Propositional_Object --P67F.refers_to-> **E53.Place:**
{E53.Place --(P89B.contains)$^{[0,n]}$ -> **E53.Place** [--
P2F.has_type -> **E55.Type**]
}
OR

E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{E73.Information_Object -- P67F.refers_to -> **E53.Place:**
{E53.Place --(P89B.contains) $^{[0,n]}$ -> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
}
}
OR
D2.Digitization_Process -- L1F.digitized ->**E70.Thing:**
{ E70.Thing--(F5F.consists_of_shows_features_of)$^{[0,n]}$-> **E70.Thing:**
{ E24.Physical_Man-Made_Thing -- P62F.depicts -> **E53.Place:**
{E53.Place --(P89B.contains) $^{[0,n]}$ -> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
OR
E89.Propositional_Object --P67F.refers_to-> **E53.Place:**
{E53.Place --(P89B.contains) $^{[0,n]}$ -> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{ **E73.Information_Object** --P67F.refers_to -> **E53.Place:**
{E53.Place --(P89B.contains) $^{[0,n]}$ -> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}

b. is referred to at

There are cases that one is interested in Events that are referred to at a certain Place. Reference implies the presence or creation of a Thing that refers to the Event in interest, as even speech is a human product, thus a Thing. So, when we talk about something, write about something or in any other way mention something the means we do it is mapped to a Thing that we create.
In this manner we connect the Event and Place FCs not only directly but also through other Things and through Events.
In CIDOC-CRM properties that define the "is referred to" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Actor is referred to at Place"* is:

E5.Event -- (P9B.forms_part_of)$^{[0,n]}$-> **E5.Event:**
{**E5.Event** -- P67B.is_referred_to_by -> **E89.Propositional_Object:**
{E89.Propositional_Object -- (P148B.is_component_of)$^{[0,n]}$->
E89.Propositional_Object:
{E73.Information_Object -- P94B.was_created_by -> **E65.Creation**:
{E65.Creation --(P9B.forms_part_of)$^{[0,n]}$-> **E5.Event**:
{E5.Event-- P7F.took_place_at ->**E53.Place:**
{E53.Place --(P89F.falls_within)$^{[0,n]}$-> **E53.Place**
[--P2F.has_type -> **E55.Type**]
}
}
}
OR

E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] -> **E24.Physical_Man-Made_Thing**:
{**E24.Physical_Man-Made_Thing** --
P53F.has_former_or_current_location -> **E53.Place :**
{E53.Place --(P89F.falls_within)[0,n]-> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing --
P108B.was_produced_by -> **E12.Production**:
{ E12.Production --P9B.forms_part_of-> **E5.Event:**
{ E5.Event -- P7F.took_place_at ->
E53.Place:
{E53.Place --(P89F.falls_within)[0,n] -> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}
}
}
OR
E5.Event -- P62B.is_depicted_by -> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] -> **E24.Physical_Man-Made_Thing**:
{ **E24.Physical_Man-Made_Thing** -- P53F.has_former_or_current_location -> **E53.Place:**
{E53.Place --(P89F.falls_within)[0,n]-> **E53.Place**
[--P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P108B.was_produced_by ->
E12.Production:
{ E12.Production --P9B.forms_part_of-> **E5.Event:**
{ E5.Event -- P7F.took_place_at ->**E53.Place**:
{E53.Place --(P89F.falls_within)[0,n]-> **E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}}

c. from

Events usually take place at one or more Places, which property is either directly connected to the Event, or may be inherited from super-events. If both cases are valid, we are interested in all the Places where an event and its super-events have taken place at.
Also here we can include events that have taken place with regard to a Thing, which is located in the place.
The respective general fundamental relationship "*Event from Place"* is:

E5.Event-- (P9B.forms_part_of)[0,n] -> **E5.Event:**
{E5.Event--P7F.took_place_at-> **E53.Place**:
{E53.Place--(P89F.falls_within) [0,n] ->
E53.Place [--P2F.has_type -> **E55.Type**]

}
OR
E5.Event--P8F.took_place_on_or_within **->E19.Physical_Object:**
{ E19.Physical_Object -- P53F.has_former_or_current_location->E53.Place:
{E53.Place--(P89F.falls_within) $^{[0,n]}$ ->**E53.Place** [--P2F.has_type -> **E55.Type**]
}
}
}

Event-Thing

a. refers to or is about

An event may refer to a Thing by Things that are created or produced in the Event and refer to or are about the Thing. In another aspect, an Event also refers to Things that are created, produced or destroyed in the Event, such as the Destruction of Parthenon that refers to Parthenon.
The respective general fundamental relationship "*Event refers to or is about Thing"* is:

E5.Event -- (P9F.consists_of) $^{[0,n]}$-> **E5.Event:**
{E63.Beginning_of_Existence --P92F.brought_into_existence ->**E70.Thing:**
{E70.Thing --(F5F.consists_of_shows_features_of) $^{[0,n]->}$**E70.Thing:**
{E24.Physical_Man-Made_Thing -- P62F.depicts **-> E70.Thing:**
{ E70.Thing --(F4F.is_composed_of) $^{[0,n]}$-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
OR
E89.Propositional_Object --P67F.refers_to-> **E70.Thing:**
{ E70.Thing --(F4F.is_composed_of) $^{[0,n]}$-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{E73.Information_Object -- P67F.refers_to -> **E70.Thing:**
{ E70.Thing --(F4F.is_composed_of) $^{[0,n]}$-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
}
}
}
OR
D2.Digitization_Process -- L1F.digitized ->**E70.Thing:**
{ E70.Thing--(F5F.consists_of_shows_features_of) $^{[0,n]}$-> **E70.Thing:**
{ E70.Thing [-- P2F.has_type -> **E55.Type**]
OR
E24.Physical_Man-Made_Thing -- P62F.depicts -> **E70.Thing:**
{E70.Thing -- (F4F.is_composed_of) $^{[0,n]}$-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
OR
E89.Propositional_Object --P67F.refers_to-> **E70.Thing:**
{E70.Thing -- (F4F.is_composed_of) $^{[0,n]}$-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}

OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{ **E73.Information_Object** --P67F.refers_to -> **E70.Thing:**

{E70.Thing -- (F4F.is_composed_of)[0,n]-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}

b. is referred to by

An Event may be referred to by Things that have as theme or subject the Event, or that refer to or are about the Event. We may even expand the reference to Events that contain the Event in reference.
In CIDOC-CRM properties that define the "is referred to by" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Event is referred to by Thing"* is:

E5.Event --(P9B.forms_part_of)[0,n] -> **E5.Event:**
{E5.Event--P67B.is_referred_to_by -> **E89.Propositional_Object:**
{**E89.Propositional_Object** -- (F5B.forms_part_of_shows_features_of)[0,n]->
E89.Propositional_Object:
{E89.Propositional_Object [--P2F.has_type -> **E55.Type**]
OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- (P46F.is_composed_of)[0,n]->
E24.Physical_Man-Made_Thing[--P2F.has_type -> **E55.Type**]
}
}
}
OR
E5.Event -- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing**:
{ **E24.Physical_Man-Made_Thing** --(F5B.forms_part_of_shows_features_of)[0,n]->
E24.Physical_Man-Made_Thing[--P2F.has_type -> **E55.Type**]
}
}

c. has met

This is a generic relationship used to return all the Events that a certain Thing has met. In other words, this means that a Thing has been present at the Event.
The respective general fundamental relationship "*Event has met Thing"* is:

E5.Event --(P9F.consists_of)[0,n] -> **E5.Event:**
{E5.Event-- P12F.occurred_in_the_presence_of -> **E70.Thing:**
{E70.Thing --(F4F.is_composed_of)[0,n]-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
}

Specializations:
For the "has met" relationship we can define three sub-relationships, in order to be able to be more specific for actions performed on things during an event. This is useful because users are likely to be interested in certain things included in an Event such as the "created", "destroyed", "modified" and "used" Things. So we define:

- created

This sub-FR is used to relate the Thing with the Events of its creation. One may think that a Thing is created during one event and not many, but also the super events of the creation event may be considered as creation events.
The respective general fundamental relationship "*Event created Thing"* is:

E5.Event --(P9F.consists_of)[0,n] -> **E5.Event:**
{ E63.Beginning_of_Existence --P92F.brought_into_existence -> **E70.Thing:**
{E70.Thing --(F4F.is_composed_of)[0,n]-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
}

- destroyed

This sub-FR is used to relate the Thing with the Events of its destruction. One may think that a Thing is destroyed during one event and not many, but also the super events of the destruction event may be considered as destruction events.
The respective general fundamental relationship "*Event destroyed Thing"* is:

E5.Event --(P9F.consists_of)[0,n] -> **E5.Event:**
{ E64.End_of_Existence -- P93F.took_out_of_existence ->**E70.Thing:**
{E70.Thing --(F4F.is_composed_of)[0,n]-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
}

- modified

This sub-FR is used to relate the Thing with the Events of its modification
The respective fundamental relationship "*Event modified Thing"* is:

E5.Event --(P9F.consists_of)[0,n] ->**E11.Modification**:
{**E11.Modification**-- P31F.has_modified->**E24.Physical_Man-Made_Thing**:
{**E24.Physical_Man-Made_Thing** --(F4F.is_composed_of)[0,n]->
E18.Physical_Thing[--P2F.has_type -> **E55.Type**]
}
}

- used

This sub-FR is used to relate the Thing with Events in which it has been used.
The respective fundamental relationship "*Event used Thing"* is:

E5.Event --(P9F.consists_of)[0,n] ->**E7.Activity**:
{**E7.Activity** -- P16F.used_specific_object -> **E70.Thing:**
{E70.Thing --(F4F.is_composed_of)[0,n]-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
OR
E7.Activity -- P125F.used_object_of_type -> **E55.Type**
}

Event-Actor

a. refers to or is about

An event may refer to an Actor by Things that are created or produced in the Event and refer to or are about the Actor.
The respective general fundamental relationship "*Event refers to or is about Actor"* is:

E5.Event -- (P9F.consists_of)[0,n]-> **E7.Activity:**
{E63.Beginning_of_Existence --P92F.brought_into_existence ->**E70.Thing:**
{E70.Thing --(F5F.consists_of_shows_features_of)[0,n]->
E70.Thing:
{E24.Physical_Man-Made_Thing -- P62F.depicts **-> E39.Actor:**

{E39.Actor --
(P107F.has_current_or_former_member_of)[0,n] ->
E39.Actor [--P2F.has_type -> **E55.Type**]
}

OR

E89.Propositional_Object --P67F.refers_to-> **E39.Actor:**
{E39.Actor --
(P107F.has_current_or_former_member_of)[0,n] ->
E39.Actor [--P2F.has_type -> **E55.Type**]
}

OR

E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{E73.Information_Object --P67F.refers_to ->**E39.Actor:**
{E39.Actor -- (P107F.has_current_or_former_member_of
)[0,n] -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
}

OR
D2.Digitization_Process -- L1F.digitized ->**E70.Thing:**
{ E70.Thing--(F5F.consists_of_shows_features_of)[0,n]-> **E70.Thing:**
{ E24.Physical_Man-Made_Thing -- P62F.depicts -> **E39.Actor:**
{E39.Actor -- (P107F.has_current_or_former_member_of
)[0,n] -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}

OR

E89.Propositional_Object --P67F.refers_to-> **E39.Actor:**
{E39.Actor -- (P107F.has_current_or_former_member_of
)[0,n] -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}

OR

E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{ **E73.Information_Object** --P67F.refers_to -> **E39.Actor:**
{E39.Actor -- (P107F.has_current_or_former_member_of
)[0,n] -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}

}
}
}
}

b. is referred to by

An actor may refer to an Event by material or immaterial Things they create and which refer to some Event.
In CIDOC-CRM properties that define the "is referred to by" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Event is referred to by Actor"* is:

E5.Event --(P9B.forms_part_of)[0,n] -> **E5.Event:**
{E5.Event --P67B.is_referred_to_by -> **E89.Propositional_Object:**

{**E89.Propositional_Object --** (F5B.forms_part_of_shows_features_of)[0,n] ->
E89.Propositional_Object:
{E89.Propositional_Object -- P94B.was_created_by -> **E65.Creation:**
{ E65.Creation -- (P9B.forms_part_of)[0,n] -> **E65.Creation**:
{ E65.Creation-- P14F.carried_out_by -> **E39.Actor:**
{E39.Actor --
(P107B.is_current_or_former_member_of) [0,n] -
> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) [0,n] ->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing--P108B.was_produced_by ->
E12.Production:
{ E12.Production -- (P9B.forms_part_of)[0,n] ->
E12.Production:
{E12.Production -- P14F.carried_out_by->
E39.Actor:
{E39.Actor --
(P107B.is_current_or_former_member_of) [0,n] -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}
}
OR
E5.Event -- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing:**
{**E24.Physical_Man-Made_Thing --** (F5B.forms_part_of_shows_features_of)[0,n]-
>**E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing--P108B.was_produced_by ->
E12.Production:
{E12.Production -- (P9B.forms_part_of) [0,n] ->
E12.Production:
{E12.Production -- P14F.carried_out_by-> **E39.Actor:**
{E39.Actor --
(P107B.is_current_or_former_member_of) [0,n] -
> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}

c. by

Events are usually carried out by Actors, and by this FR one can query for all the Events carried out by certain Actors. Also the *by* FR includes the influenced by relation between the two Categories.
The respective general fundamental relationship "*Event by Actor"* is:

E5.Event-- (P9B.forms_part_of)[0,n]-> **E7.Activity:**
{E7.Activity--P14F.carried_out_by->**E39.Actor:**

{E39.Actor--(P107B.is_current_or_former_member_of)[0,n]-> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
OR
E7.Activity -- P15F.was_influenced_by-> **E39.Actor:**
{E39.Actor--(P107B.is_current_or_former_member_of)[0,n]->**E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}

d. has met

This is a general relationship that connects an Event with actors that were present at the Event.
The respective general fundamental relationship "*Event has met Actor"* is:

E5.Event--(P9F.consists_of)[0,n] -> **E5.Event:**
{**E5.Event** -- P12F.occurred_in_the_presence_of-> **E39.Actor:**
{E39.Actor--(P107B.is_current_or_former_member_of)[0,n]->**E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}

Specializations:

In the fundamental relationship *has met* we can also define some sub-fundamental relationships, which contain more restricted information. This information is however commonly asked for by the users and a more specialized query may return better results based to what they actually want to know about.

- Brought into existence

With this specification we query only for events that brought into existence the specific Actor.
The respective general fundamental relationship "*Event brought into existence Actor"* is:

E5.Event--(P9F.consists_of)[0,n] -> **E5.Event:**
{E63.Beginning_of_Existence --P92F.brought_into_existence-> **E39.Actor:**
{ E39.Actor --(P107F.has_current_or_former_member_of)[0,n] -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}

- Took out of existence

With this specification we query only for events that took out of existence the specific Actor.
The respective fundamental relationship "*Event took out of existence Actor"* is:

E5.Event--(P9F.consists_of)[0,n] -> **E5.Event:**
{E64.End_of_Existence -- P93F.took_out_of_existence -> **E39.Actor:**
{ E39.Actor --(P107F.has_current_or_former_member_of)[0,n] -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}

Event-Time

The Event-Event fundamental relationships can be switched to Event-Time fundamental relationships, by further adding the CIDOC-CRM property P4F.has_time-span at the range category Event **(E5.Event** --P4F.has_time-span->**E52.Time-Span)**. This happens because Time refers to the chronological definition of Events.

a. refers to or is about

An event may refer to a specific Time by Things that refer to or are about the Thing.
The respective general fundamental relationship "*Event refers to or is about Time"* is:

E5.Event -- (P9F.consists_of)[0,n]-> **E5.Event:**

{E63.Beginning_of_Existence --P92F.brought_into_existence ->**E70.Thing:**
{E70.Thing --(F5F.consists_of_shows_features_of)[0,n]-> **E70.Thing:**
{E24.Physical_Man-Made_Thing -- P62F.depicts -> **E52.Time-Span:**
{E52.Time-Span --(P86B.contains)[0,n]->**E52.Time-Span**[-- P2F.has_type -> **E55.Type**]
}

OR
E89.Propositional_Object -- P67F.refers_to->**E52.Time-Span:**
{E52.Time-Span --(P86B.contains)[0,n]->**E52.Time-Span**[-- P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{E73.Information_Object -- P67F.refers_to ->**E52.Time-Span:**
{E52.Time-Span --(P86B.contains)[0,n]->**E52.Time-Span**[-- P2F.has_type -> **E55.Type**]
}
}
}
}
OR
D2.Digitization_Process -- L1F.digitized ->**E70.Thing:**
{ E70.Thing--(F5F.consists_of_shows_features_of)[0,n]-> **E70.Thing:**
{ E24.Physical_Man-Made_Thing -- P62F.depicts -> **E52.Time-Span:**
{E52.Time-Span --(P86B.contains)[0,n]->**E52.Time-Span**[-- P2F.has_type -> **E55.Type**]
}

OR
E89.Propositional_Object --P67F.refers_to-> **E52.Time-Span:**
{E52.Time-Span --(P86B.contains)[0,n]->**E52.Time-Span**[-- P2F.has_type -> **E55.Type**]
}
OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{ **E73.Information_Object** --P67F.refers_to ->
E52.Time-Span:
{E52.Time-Span --(P86B.contains)[0,n]->
E52.Time-Span [-- P2F.has_type -> **E55.Type**]
}
}
}
}
}

b. from/is part of

Events are usually placed in time spans, and this relationship is designed to return the Events that happened in a specific Time.
The respective general fundamental relationship "*Event from Time"* is:

E5.Event--{ (P9B.forms_part_of)[0,n] OR (P119F.meets_in_time_with) [0,n] OR (P119B.is_met_in_time_by) [0,n] OR (P118F.overlaps_in_time_with) [0,n] OR (P118B.is_overlapped_in_time_by) [0,n] OR (P117F.occurs_during) [0,n] OR (P114F.is_equal_in_time_to) [0,n] OR (P10F.falls_within) [0,n] }-> **E5.Event:**
{E5.Event-- P4F.has_time-span-> **E52.Time-Span:**
{E52.Time-Span --(P86F.falls_within)[0,n]-> **E52.Time-Span**[-- P2F.has_type -> **E55.Type**]

}
}

c. has part

Events may contain other events or may co-occur with other events. Nevertheless, only the time span of the events may be known. So, it is useful to be able to ask using this information. It is the reverse of the *Event from Time* FR. Both the relations include relations that are about the co-occurrence of the two Events.
The respective general fundamental relationship "*Event has part Time"* is:

E5.Event--{(P9F.consists_of)[0,n]OR(P119F.meets_in_time_with) [0,n] OR (P119B.is_met_in_time_by) [0,n] OR(P118F.overlaps_in_time_with) [0,n] OR (P118B.is_overlapped_in_time_by) [0,n] OR(P117B.includes)[0,n] OR (P114F.is_equal_in_time_to) (0,n)OR(P10B.contains) [0,n]} -> **E5.Event:**
{E5.Event-- P4F.has_time-span-> **E52.Time-Span:**
{E52.Time-Span --(P86F.falls_within)[0,n]-> **E52.Time-Span**[-- P2F.has_type -> **E55.Type**]
}
}

Event-Event

The Event-Event fundamental relationships can be switched to Event-Time fundamental relationships, by further adding the CIDOC-CRM property P4F.has_time-span at the range category Event **(E5.Event** --P4F.has_time-span->**E52.Time-Span)**. This happens because Time refers to the chronological definition of Events.

a. refers to or is about

An event may refer to an Event by Things that are created or produced in the Event and refer to or are about the Event.
The respective general fundamental relationship "*Event refers to or is about Event"* is:

E5.Event -- (P9F.consists_of)[0,n]-> **E5.Event:**
{E63.Beginning_of_Existence --P92F.brought_into_existence ->**E70.Thing:**
{E70.Thing --(F5F.consists_of_shows_features_of)[0,n]-> **E70.Thing:**
{E24.Physical_Man-Made_Thing -- P62F.depicts **-> E5.Event:**
{**E5.Event** -- (P9F.consists_of)[0,n]->**E5.Event** [-- P2F.has_type-> **E55.Type**]
}
OR
E89.Propositional_Object -- P67F.refers_to-> **E5.Event:**
{**E5.Event** -- (P9F.consists_of)[0,n]->**E5.Event** [-- P2F.has_type-> **E55.Type**]
}

OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{E73.Information_Object -- P67F.refers_to -> **E5.Event:**
{**E5.Event** -- (P9F.consists_of)[0,n]->**E5.Event** [-- P2F.has_type-> **E55.Type**]
}
}
}
}
OR
D2.Digitization_Process -- L1F.digitized ->**E70.Thing:**
{ E70.Thing--(F5F.consists_of_shows_features_of)[0,n]-> **E70.Thing:**
{ E24.Physical_Man-Made_Thing -- P62F.depicts -> **E5.Event:**
{**E5.Event** -- (P9F.consists_of)[0,n]->**E5.Event** [--P2F.has_type-> **E55.Type**]

}
OR
E89.Propositional_Object --P67F.refers_to-> **E5.Event:**
{**E5.Event** -- (P9F.consists_of)[0,n]->**E5.Event** [--P2F.has_type-> **E55.Type**]
}

OR
E24.Physical_Man-Made_Thing -- P128F.carries ->
E73.Information_Object:
{ **E73.Information_Object** --P67F.refers_to -> **E5.Event:**
{ **E5.Event** -- (P9F.consists_of)[0,n]-> **E5.Event** [--P2F.has_type-> **E55.Type**]
}
}
}
}
}

b. is referred to at

An Event may be referred by Events during which Things that refer to the Event are created.
In CIDOC-CRM properties that define the "is referred to at" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Event is referred to at Event"* is:

E5.Event -- (P9F.consists_of)[0,n]->**E5.Event:**
{E5.Event--P67B.is_referred_to_by -> **E89.Propositional_Object:**
{**E89.Propositional_Object** --(F5B.forms_part_of_shows_features_of)[0,n]
-> **E89.Propositional_Object:**
{E89.Propositional_Object -- P94B.was_created_by ->
E65.Creation:
{ E65.Creation -- (P9B.forms_part_of)[0,n] -> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
OR
E73.Information_Object -- P128B.is_carried_by ->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing --(P46B.forms_part_of)[0,n] -> **E24.Physical_Man-Made_Thing**:
{E24.Physical_Man-Made_Thing--P108B.was_produced_by -> **E12.Production:**
{E12.Production -- (P9B.forms_part_of)[0,n] -> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
}
}
}
OR
E5.Event-- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing:**
{ **E24.Physical_Man-Made_Thing** -- (F5B.forms_part_of_shows_features_of)[0,n]->**E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing--P108B.was_produced_by ->
E12.Production:
{ E12.Production -- (P9B.forms_part_of)[0,n] -> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
}

}
}

c. from/is part of

Events may occur in the context of a bigger event or may occur concurrently with another event.
The respective general fundamental relationship "*Event from Event"* is:

E5.Event--{ (P9B.forms_part_of)[0,n] OR (P119F.meets_in_time_with) [0,n] OR (P119B.is_met_in_time_by) [0,n] OR P118F.overlaps_in_time_with OR P132F.overlaps_with OR (P118B.is_overlapped_in_time_by) [0,n] OR (P117F.occurs_during) [0,n] OR (P114F.is_equal_in_time_to) [0,n] OR (P10F.falls_within) [0,n] }-> **E5.Event** [--P2F.has_type -> **E55.Type**]

d. has part

Events may contain other events or may co-occur with other events. It is the reverse of the *Event from Event* FR, although the relations that are about the co-occurrence of the two Events.
The respective general fundamental relationship "*Event has part Event"* is:

E5.Event--{(P9F.consists_of)[0,n]OR(P119F.meets_in_time_with) [0,n] OR (P119B.is_met_in_time_by) [0,n] OR(P118F.overlaps_in_time_with) [0,n] OR P118B.is_overlapped_in_time_by OR P118B.is_overlapped_in_time_by OR(P117B.includes)[0,n] OR (P114F.is_equal_in_time_to) (0,n)OR(P10B.contains) [0,n]} -> **E5.Event**[--P2F.has_type -> **E55.Type**]

Event-Concept

a. has type

This relationship connects an Event with its type, which describes its concept. An Event may belong to more than one type categories.
The respective fundamental relationship "*Event has type Concept"* is:

E5.Event--(P9F.consists_of)[0,n]-> **E5.Event:**
{E5.Event-- P2F.has_type-> **E55.Type**:
{ **E55.Type** -- (P127F.has_broader_term)[0,n] -> **E55.Type** [P2F.has_type -> **E55.Type**]
}
}

TIME

Time can be considered as an extension to the fundamental category Event. To be more precise, Event can be placed in Time, by specifying the time barriers within which it extends. Many times it is likely that we do not know the name of the Event, but we identify it by the time when it occurred. Thus, Time can be used in the same way as the Event FC, only by adding the *is time-span of* property before the Event domain Category. That is:
E52.Time-Span-- P4B.is_time-span_of ->**E5.Event**
It is also decided that E5.Event is going to be defined as a subclass of the class E52.Time-Span. Using the CIDOC-CRM and the CIDOC-CRMdig, it is a common practice that times are defined as rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime" properties of the E5.Event class that they define, instead of using the property P4F.has_time-span to connect instances of the two classes. In this way, even though properties like the crmdig:L31F.has_starting_date-time has as range literals (rdfs:literal) we can also interpret them as Times. For this reason, when querying about the Time FC, it would be safe to also include to the query, instances of the FC Event. Then, before displaying the results to the user, we connect the instances of the Event results to their time-span, using the properties of the schema that are defined as rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime",and which provide the time as *literal*. This is also why we cannot include such results to the result set, when asking for instances of the E52.Time-Span.
Take for example the case:
<uuid:DigitizationEvent1> crmdig:L31F.has_starting_date-time < 2010-10-10T00:00:00Z >.
Here the time is added as literal in the repository, but we can include it in the results, by asking for events and include the respective linking with crmdig:L31F.has_starting_date-time. Similar links, are: crmdig:L32F.has_ending_date-time and L61F.was_ongoing_at.
The following paths, define the connection of the FC Time with the other FCs, directly and not through the rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime", property on the Events. So, the two solutions must be combined at query time to include all possible results.
Nevertheless relationships like the "Time refers to FC" may appeal as too wide and general, and are not likely to be used, so I exclude them from this Fundamental Category.

Time-Place

a. has met

Time and Place do not seem to have any special relations, except from the "has met" relationship. A time has met a Place actually means that within this time an event has happened at this place.
The respective general fundamental relationship "*Time has met Place"* is:

E52.Time-Span --(P86F.falls_within)[0,n] -> **E52.Time-Span:**
{ E52.Time-Span --P4B.is_time-span_of **->E5.Event:**
{E5.Event --(P9B.forms_part_of)[0,n] -> **E5.Event:**
{E5.Event--P7F.took_place_at-> **E53.Place**:
{E53.Place--(P89F.falls_within) [0,n] ->
E53.Place [--P2F.has_type -> **E55.Type**]
}
}
}
}

Time-Thing

a. is referred to by

A Time may be referred to directly by a Thing or by an event that happens during this Time. An Event may be referred to by Things that have as theme or subject the Event, or that refer to or are about the Event.
In CIDOC-CRM properties that define the "is referred to by" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Time is referred to by Thing"* is:

E52.Time-Span --(P86F.falls_within)[0,n] -> **E52.Time-Span:**

{ E52.Time-Span --P67B.is_referred_to_by -> **E89.Propositional_Object:**
{**E89.Propositional_Object** -- (F5B.forms_part_of_shows_features_of)[0,n]->
E89.Propositional_Object:
{E89.Propositional_Object [--P2F.has_type -> **E55.Type**]
OR
E73.Information_Object -- P128B.is_carried_by ->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing -- (P46F.is_composed_of)[0,n]-
> **E24.Physical_Man-Made_Thing**[--P2F.has_type ->
E55.Type]
}
}
}
OR
E52.Time-Span -- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing**:
{**E24.Physical_Man-Made_Thing** --
(F5B.forms_part_of_shows_features_of)[0,n]-> **E24.Physical_Man-Made_Thing**[--P2F.has_type -> **E55.Type**]
}
}

b. has met

This is a generic relationship used to return all the Times that a certain Thing has met. In other words, this means that a Thing has been present at some Event that happened during the Time.
The respective general fundamental relationship "*Event has met Thing"* is:

E52.Time-Span --(P86F.falls_within)[0,n] -> **E52.Time-Span:**
{ E52.Time-Span --P4B.is_time-span_of ->**E5.Event:**
{E5.Event --(P9F.consists_of)[0,n] -> **E5.Event:**
{E5.Event-- P12F.occurred_in_the_presence_of -> **E70.Thing:**
{E70.Thing --(F4F.is_composed_of)[0,n]-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
}
}
}

Specializations:
For the "has met" relationship we can define three sub-relationships, in order to be able to be more specific for actions performed on things during an event. This is useful because users are likely to be interested in certain things included in an Event such as the "created", "destroyed", "modified" and "used" Things. So we define:

- signals the beginning of

This sub-FR is used to relate the Thing with the Time of its creation. One may think that a Thing is created during one event and not many, but also the super events of the creation event may be considered as creation events.
The respective general fundamental relationship "*Time signals the beginning of Thing"* is:

E52.Time-Span --(P86F.falls_within)[0,n] -> **E52.Time-Span:**
{ E52.Time-Span --P4B.is_time-span_of ->**E5.Event:**
{E5.Event --(P9F.consists_of)[0,n] -> **E5.Event:**
{ E63.Beginning_of_Existence --P92F.brought_into_existence ->
E70.Thing:
{E70.Thing --(F4F.is_composed_of)[0,n]-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
}
}

}

- signals the end of

This sub-FR is used to relate the Thing with the Time of its destruction. One may think that a Thing is destroyed during one event and not many, but also the super events of the destruction event may be considered as destruction events and so may the Time when they occurred.
The respective general fundamental relationship "*Time signals the end of Thing"* is:

E52.Time-Span --(P86F.falls_within)[0,n] -> **E52.Time-Span:**
{ E52.Time-Span --P4B.is_time-span_of ->**E5.Event:**
{E5.Event --(P9F.consists_of)[0,n] -> **E5.Event:**
{ E64.End_of_Existence -- P93F.took_out_of_existence ->**E70.Thing:**
{E70.Thing --(F4F.is_composed_of)[0,n]-> **E70.Thing** [--P2F.has_type -> **E55.Type**]
}
}
}
}

Time-Actor

a. is referred to by

An actor may refer to some Time by material or immaterial Things refer to the Time.
In CIDOC-CRM properties that define the "is referred to by" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Time is referred to by Actor"* is:

E52.Time-Span --(P86F.falls_within)[0,n] -> **E52.Time-Span:**
{ E52.Time-Span --P67B.is_referred_to_by -> **E89.Propositional_Object:**
{**E89.Propositional_Object** -- (F5B.forms_part_of_shows_features_of)[0,n] -> **E89.Propositional_Object:**
{E89.Propositional_Object -- P94B.was_created_by -> **E65.Creation:**
{ E65.Creation -- (P9B.forms_part_of)[0,n] -> **E65.Creation**:
{ E65.Creation-- P14F.carried_out_by -> **E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of) [0,n] -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
OR
E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of)[0,n] -> **E24.Physical_Man-Made_Thing**:
{E24.Physical_Man-Made_Thing-- P108B.was_produced_by -> **E12.Production:**
{E12.Production -- (P9B.forms_part_of)[0,n] -> **E7.Activity**:
{ E7.Activity -- P14F.carried_out_by-> **E39.Actor:**
{ E39.Actor -- (P107B.is_current_or_former_member_of) [0,n] -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}

}
}
}
}
}
}
OR
E52.Time-Span -- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing:**
{**E24.Physical_Man-Made_Thing** --
(F5B.forms_part_of_shows_features_of)[0,n]->**E24.Physical_Man-Made_Thing:**
{**E24.Physical_Man-Made_Thing**--
P108B.was_produced_by -> **E12.Production:**
{**E12.Production** -- (P9B.forms_part_of) [0,n] ->
E7.Activity:
{**E7.Activity** -- P14F.carried_out_by->
E39.Actor:
{**E39.Actor** --
(P107B.is_current_or_former_member_of) [0,n] -> **E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}
}
}

b. has met

This is a general relationship that connects a Time with actors that were present at Events that occurred during the Time.
The respective general fundamental relationship "T*ime has met Actor"* is:

E52.Time-Span --(P86F.falls_within)[0,n] -> **E52.Time-Span:**
{**E52.Time-Span** --P4B.is_time-span_of ->**E5.Event:**
{**E5.Event**--(P9F.consists_of)[0,n] -> **E5.Event:**
{**E5.Event** -- P12F.occurred_in_the_presence_of-> **E39.Actor:**
{**E39.Actor**--(P107B.is_current_or_former_member_of)[0,n]->**E39.Actor** [--P2F.has_type -> **E55.Type**]
}
}
}
}

Specializations:

In the fundamental relationship *has met* we can also define some sub-fundamental relationships, which contain more restricted information. This information is however commonly asked for by the users and a more specialized query may return better results based to what they actually want to know about.

- signals the beginning of

With this specification we query only for times within which events that brought into existence the specific Actor took place.
The respective general fundamental relationship "*Time signals the beginning of Actor"* is:

E52.Time-Span --(P86F.falls_within)[0,n] -> **E52.Time-Span:**
{**E52.Time-Span** --P4B.is_time-span_of ->**E5.Event:**
{**E5.Event**--(P9F.consists_of)[0,n] -> **E5.Event:**

{E63.Beginning_of_Existence --P92F.brought_into_existence->
E39.Actor:
{ E39.Actor --(P107F.has_current_or_former_member)[0,n] ->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
}
}
}

- signals the end of

With this specification we query only for times that took out of existence the specific Actor.
The respective fundamental relationship "*Time signals the end of Actor"* is:

E52.Time-Span --(P86F.falls_within)[0,n] -> **E52.Time-Span:**
{E52.Time-Span --P4B.is_time-span_of ->**E5.Event:**
{E5.Event--(P9F.consists_of)[0,n] -> **E5.Event:**
{E64.End_of_Existence -- P93F.took_out_of_existence -> **E39.Actor:**
{ E39.Actor --(P107F.has_current_or_former_member)[0,n] ->
E39.Actor [--P2F.has_type -> **E55.Type**]
}
}
}
}

Time-Event

a. is referred to at

Time may be referred to at Events during which Things that refer to the Time are created.
In CIDOC-CRM properties that define the "is referred to at" relationship are:

- P67B.is_referred_to_by
- P62B.is_depicted_by

The respective general fundamental relationship "*Time is referred to at Event"* is:

E52.Time-Span --(P86F.falls_within)[0,n] -> **E52.Time-Span:**
{ E52.Time-Span --P67B.is_referred_to_by -> **E89.Propositional_Object:**
{**E89.Propositional_Object** --(F5B.forms_part_of_shows_features_of)[0,n]
-> **E89.Propositional_Object:**
{E89.Propositional_Object -- P94B.was_created_by ->
E65.Creation:
{ E65.Creation -- (P9B.forms_part_of)[0,n] -> **E5.Event** [-
-P2F.has_type -> **E55.Type**]
}
OR
E73.Information_Object -- P128B.is_carried_by ->
E24.Physical_Man-Made_Thing:
{E24.Physical_Man-Made_Thing --
(P46B.forms_part_of)[0,n] -> **E24.Physical_Man-Made_Thing**:
{E24.Physical_Man-Made_Thing--
P108B.was_produced_by -> **E12.Production:**
{E12.Production -- (P9B.forms_part_of)[0,n]
-> **E5.Event** [--P2F.has_type -> **E55.Type**]
}
}
}
}
}

OR
E52.Time-Span -- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing:**
{ **E24.Physical_Man-Made_Thing** -- (F5B.forms_part_of_shows_features_of)[0,n]-
>**E24.Physical_Man-Made_Thing:**
{E24.Physical_Man-Made_Thing--P108B.was_produced_by ->
E12.Production:
{ E12.Production -- (P9B.forms_part_of)[0,n] -> **E5.Event**
[--P2F.has_type -> **E55.Type**]
}
}
}
}

b. is part of

Time is part of an Event means that the period within with the Event took place, contains this Time.
The respective general fundamental relationship "T*ime from Event"* is:

E52.Time-Span --(P86F.falls_within)[0,n] -> **E52.Time-Span:**
{E52.Time-Span --P4B.is_time-span_of ->**E5.Event:**
{E5.Event--{ (P9B.forms_part_of)[0,n] OR (P119F.meets_in_time_with) [0,n] OR
(P119B.is_met_in_time_by) [0,n] OR (P118F.overlaps_in_time_with) [0,n] OR
(P118B.is_overlapped_in_time_by) [0,n] OR (P117F.occurs_during) [0,n] OR
(P114F.is_equal_in_time_to) [0,n] OR (P10F.falls_within) [0,n] OR
(P115F.finishes)[0,n] OR (P116F.starts)[0,n]}-> **E5.Event** [--P2F.has_type ->
E55.Type]
}
}

c. has part

Events can be defined in the context of wider Time spans. With this relationship we can find which Times are considered to contain the specified Event.
The respective general fundamental relationship "*Time has part Event"* is:

E52.Time-Span --(P86F.falls_within)[0,n] -> **E52.Time-Span:**
{E52.Time-Span --P4B.is_time-span_of ->**E5.Event:**
{E5.Event--{(P9F.consists_of)[0,n] OR (P119F.meets_in_time_with) [0,n] OR (
P119B.is_met_in_time_by) [0,n] OR (P118F.overlaps_in_time_with) [0,n] OR (
P118B.is_overlapped_in_time_by) [0,n] OR (P117B.includes)[0,n] OR
(P114F.is_equal_in_time_to) (0,n) OR (P10B.contains) [0,n] OR
(P115B.is_finished_by) [0,n] OR (P116B.is_started_by) [0,n]} -> **E5.Event**[--
P2F.has_type -> **E55.Type**]
}
}

Time-Time

a. is part of

This relationship correlates the specified Time in range with Times that are parts of it. This can be either done directly or indirectly through events.
The respective general fundamental relationship "T*ime is part of Time"* is:

E52.Time-Span --(P86F.falls_within)[0,n] -> **E52.Time-Span:**
{E52.Time-Span --P4B.is_time-span_of ->**E5.Event:**
{E5.Event--{ (P9B.forms_part_of)[0,n] OR (P119F.meets_in_time_with) [0,n] OR
(P119B.is_met_in_time_by) [0,n] OR (P118F.overlaps_in_time_with) [0,n] OR
(P118B.is_overlapped_in_time_by) [0,n] OR (P117F.occurs_during) [0,n] OR
(P114F.is_equal_in_time_to) [0,n] OR (P10F.falls_within) [0,n] OR
(P115F.finishes)[0,n] OR (P116F.starts)[0,n]}-> **E5.Event**:

{ E5.Event-- P4F.has_time-span **-> E52.Time-Span**[--P2F.has_type -> **E55.Type**]
}

}
}

b. has part

Events can be defined in the context of wider Time spans. With this relationship we can find which Times are considered to contain the specified Event.
The respective general fundamental relationship "*Time has part Time"* is:

E52.Time-Span --(P86F.falls_within)[0,n] **-> E52.Time-Span:**
{E52.Time-Span --P4B.is_time-span_of **->E5.Event:**
{E5.Event--{(P9F.consists_of)[0,n] OR (P119F.meets_in_time_with)[0,n] OR (P119B.is_met_in_time_by)[0,n] OR (P118F.overlaps_in_time_with)[0,n] OR (P118B.is_overlapped_in_time_by)[0,n] OR (P117B.includes)[0,n] OR (P114F.is_equal_in_time_to)[0,n] OR (P10B.contains)[0,n] OR(P115B.is_finished_by)[0,n] OR (P116B.is_started_by)[0,n]} -> **E5.Event**:
{ E5.Event-- P4F.has_time-span **-> E52.Time-Span**[--P2F.has_type -> **E55.Type**]
}
}
}

Time-Concept

a. has type

This relationship connects a Time with its type, for example decade, century etc.
The respective fundamental relationship "*Event has type Concept"* is:

E52.Time-Span -- P2F.has_type-> **E55.Type**:
{**E55.Type** -- {(P127F.has_broader_term)[0,n] OR (P2B.is_type_of)[0,n]-> **E55.Type**
}

CONCEPT

Concept is the Fundamental Category comprising the Type(s) that instances of the other Fundamental Categories have. This is a separate category from Thing concerning the discourse of cultural heritage and more specifically the digitization of cultural heritage objects. Instances of Concept are universal as they categorize particular instances whereas instances of Things are particular since they have specific identification and properties. Here the declaration of Concept does is not specified as it is not concerned to be a Thing. In other information systems that describe Types, such as in Biology, Types could be concerned as Things.
To link instances of Concept to instances of other FCs we use the *has type* property.

Concept-Place

a. is type of

Concepts that comprise the Types of a certain Place, including the types of its sub-parts.
The respective fundamental relationship "*Concept is type of Place"* is:

E55.Type -- {(P127B.has_narrower_term)[0,n] OR (P2B.is_type_of)[0,n]}-> **E55.Type:**
{E55.Type --P2B.is_type_of->**E53.Place:**
{E53.Place -- (P89F.falls_within)[0,n]-> **E53.Place**
}
}

Concept-Thing

a. is type of

Concepts that comprise the Types of a certain Thing.

The respective fundamental relationship "*Concept is type of Thing"* is:

E55.Type –(P127B.has_narrower_term)[0,n] -> **E55.Type:**
{E55.Type --P2B.is_type_of->**E70.Thing:**
{E70.Thing -- (F4B.is_component_of) [0,n] -> **E70.Thing**
}
OR
E57.Material – F45B.is_incorporated_in-> **E70.Thing:**
{E70.Thing -- (F4B.is_component_of) [0,n] -> **E70.Thing**
}
OR
E57.Material –P68B.use_foreseen_by->**E29.Design_or_Procedure:**
{ E29.Design_or_Procedure – P33B.was_used_by->**E7.Activity:**
{ E7.Activity -- P92F.brought_into_existence-> **E70.Thing:**
{E70.Thing -- (F4B.is_component_of) [0,n] -> **E70.Thing**
}
}
}
OR
E57.Material --P126B.was_employed_in-> **E11.Modification:**
{ E7.Activity -- P92F.brought_into_existence-> **E70.Thing:**
{E70.Thing -- (F4B.is_component_of) [0,n] -> **E70.Thing**
}
}
OR
E57.Material – P2B.is_type_of -> **E3.Condition_State:**
{ E3.Condition_State -- (P5B.forms_part_of)[0,n] **-> E3.Condition_State:**
{ E3.Condition_State --P44B.is_condition_of **-> E70.Thing:**
{E70.Thing -- (F4B.is_component_of) [0,n] -> **E70.Thing**
}
}
}

}

b. is referred to by

E55.Type – (P127B.has_narrower_term)[0,n] -> **E55.Type:**
{ E55.Type --P67B.is_referred_to_by -> **E89.Propositional_Object:**
{E89.Propositional_Object -- (P148B.is_component_of)[0,n] ->
E89.Propositional_Object[--P2F.has_type -> **E55.Type**]
OR
E89.Propositional_Object -- (P148B.is_component_of)[0,n] ->
E89.Propositional_Object:
{E73.Information_Object -- P128B.is_carried_by -> **E24.Physical_Man-Made_Thing:**
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) 0,n) ->
E24.Physical_Man-Made_Thing[--P2F.has_type -> **E55.Type**]
}
}
}
OR
E55.Type -- P62B.is_depicted_by-> **E24.Physical_Man-Made_Thing**:
{ E24.Physical_Man-Made_Thing -- (P46B.forms_part_of) 0,n) ->
E24.Physical_Man-Made_Thing[--P2F.has_type -> **E55.Type**]
}
}

Concept-Actor

a. is type of

Concepts that comprise the Types of a certain Thing.
The respective fundamental relationship "*Concept is type of Actor"* is:

E55.Type -- (P127B.has_narrower_term)[0,n]-> **E55.Type:**
{E55.Type --P2B.is_type_of->**E39.Actor:**
{E39.Actor -- (P107B.is_current_or_former_member_of)[0,n]-> **E39.Actor**
}
}

Concept-Event

a. is type of

Concepts that comprise the Types of a certain Event.
Excavation-- P2B.is_type_of -> **Excavation Event at Knossos**

The respective fundamental relationship "*Concept is type of Event"* is:

E55.Type -- (P127B.has_narrower_term)[0,n]-> **E55.Type:**
{E55.Type --P2B.is_type_of->**E5.Event:**
{ E5.Event -- (P9B.forms_part_of)[0,n]-> **E5.Event**
}
}

Concept-Time

a. is type of

Concepts that comprise the Types of a certain Time. This relationship is of minor interest, I keep it though for matter of symmetry.
The respective fundamental relationship "*Concept is type of Time"* is:

E55.Type -- (P127B.has_narrower_term)[0,n]-> **E55.Type:**
{E55.Type --P2B.is_type_of-> **E52.Time-Span**
}

Concept-Concept

a. has type

A Concept may have broader terms with which it is expressed, and with this FR we can get all the broader terms of this type.
The respective fundamental relationship "*Concept has type Concept"* is:

E55.Type--(P127F.has_broader_term)[0,n] -> **E55.Type**

b. is type of

A Concept may have narrower terms with which it is expressed, and with this FR we can get all the narrower terms of this type.
The respective fundamental relationship "*Concept is type of Concept"* is:

E55.Type-- (P127B.has_narrower_term)[0,n] -> **E55.Type**