Implementing CIDOC CRM Search Based on Fundamental Relations and OWLIM Rules*

Vladimir Alexiev, Ontotext Corp

vladimir.alexiev@ontotext.com

Abstract. The CIDOC CRM provides an ontology for describing entities, properties and relationships appearing in cultural heritage (CH) documentation, history and archeology. CRM promotes shared understanding by providing an extensible semantic framework that any CH information can be mapped to. CRM data is usually represented in semantic web format (RDF) and comprises complex graphs of nodes and properties.
An important question is how a user can search through such complex graphs, since the number of possible combinations is staggering. One approach "compresses" the semantic network by mapping many CRM entity classes to a few "Fundamental Concepts" (FC), and mapping whole networks of CRM properties to fewer "Fundamental Relations" (FR). These FC and FRs serve as a "search index" over the CRM semantic web and allow the user to use a simpler query vocabulary.
We describe an implementation of CRM FR Search based on OWLIM Rules, done as part of the ResearchSpace (RS) project. We describe the technical details, problems and difficulties encountered, benefits and disadvantages of using OWLIM rules, and preliminary performance results. We provide implementation experience that can be valuable for further implementation, definition and maintenance of CRM FRs.

Keywords: CIDOC CRM, cultural heritage, semantic search, Fundamental Concepts, Fundamental Relations

1 Introduction

The CIDOC Conceptual Reference Model (CRM) [1], ISO Standard 21127:2006, provides an ontology for describing entities, properties and relationships appearing in cultural heritage (CH) documentation, history and archeology. CRM promotes shared understanding by providing an extensible semantic framework that any CH information can be mapped to. CRM data is usually represented in semantic web format (RDF) and comprises complex graphs of nodes and properties.

* This work is partially supported by the Andrew W. Mellon Foundation under the ResearchSpace project of the British Museum. The author thanks the anonymous referees for their feedback

An important question is how a user can search through such complex graphs, since the number of possible combinations is staggering. [2] presents one approach that "compresses" the semantic network by mapping many CRM entity classes to a few "Fundamental Concepts" (FC), and mapping whole networks of CRM properties to fewer "Fundamental Relations" (FR). These FC and FRs serve as a "search index" over the CRM semantic web and allow the user to use a simpler query vocabulary.

We describe an implementation of CRM FR Search based on OWLIM Rules [6], done as part of the ResearchSpace project [6] funded by the Andrew W. Mellon foundation and run by the British Museum. We describe the technical details of our approach, problems and difficulties encountered, benefits and disadvantages of using OWLIM rules, and preliminary performance results. We provide implementation experience that can be a valuable guide for the further implementation, definition and maintenance of CRM FRs.

The FP7 project 3D COFORM [7] is also implementing FR search, and we have established a collaboration.

2 Example: Thing from Place

As an example, let's consider the FR "Thing from Place". It is intended to capture all alternatives through which a Thing's **origin** can be related to Place, and is defined in [8] as:

```
FC70_Thing --(P46i_forms_part_of* | P106i_forms_part_of* | P148i_is_component_of*)-> FC70_Thing:
 {FC70_Thing --(P53_has_former_or_current_location | P54_has_current_permanent_location)-> E53_Place:
  {E53_Place --P89_falls_within*-> E53_Place}
 OR FC70_Thing --P92i_was_brought_into_existence_by-> E63_Beginning_of_Existence:
  {E63_Beginning_of_Existence --P9i_forms_part_of*-> E5_Event:
   {E5_Event --P7_took_place_at-> E53_Place:
    {E53_Place --P89_falls_within*-> E53_Place}
   OR E7_Activity --P14_carried_out_by-> E39_Actor:
    {E39_Actor --P107i_is_current_or_former_member_of* -> E39_Actor:
     {E39_Actor --P74_has_current_or_former_residence  -> E53_Place:
      {E53_Place --P89_falls_within*-> E53_Place}
     OR E39_Actor --P92i_was_brought_into_existence_by-> E63_Beginning_of_Existence:
      {E63_Beginning_of_Existence --P9i_forms_part_of*-> E5_Event:
       {E5_Event --P7_took_place_at-> E53_Place:
        {E53_Place --P89_falls_within* -> E53_Place}}}}}}}
 OR E19_Physical_Thing  --P25i_moved_by-> E9_Move:
  {E9_Move --(P26_moved_to | P27_moved_from)-> E53_Place:
   {E53_Place  --P89_falls_within*-> E53_Place}}
 OR E19_Physical_Object --P24i_changed_ownership_through-> E8_Acquisition:
  {E8_Acquisition --P9i_forms_part_of*-> E5_Event:
   {E5_Event --P7_took_place_at-> E53_Place:
    {E53_Place --P89_falls_within*-> E53_Place}}}}
```

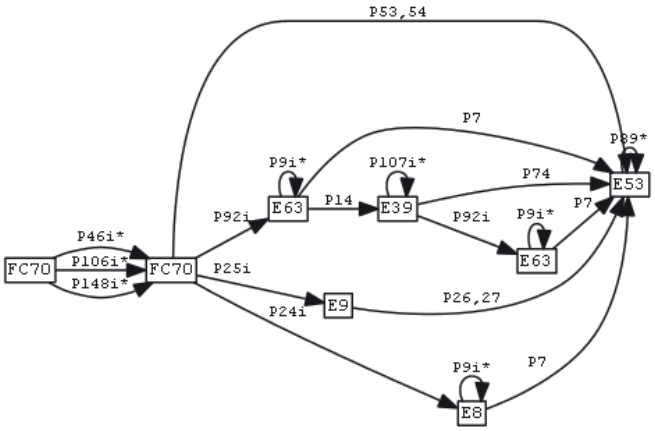
Note: we've made the following (mostly notational) simplifications:

- removed the construct "--P2F.has_type-> E55.Type" (allowing to search by event type) from a number of places
- removed "C2.Finding" which is a Find event of interest to archeology, defined in 3D COFORM but not part of CRM proper
- renamed "C1.Object" to "FC70_Thing" (which stands for Fundamental Concept "Thing")
- used Erlangen CRM [9] notation for entities (e.g. E5_Event) and properties (e.g. P89_falls_within, P24i_changed_ownership_through)
- used "property*" instead of "(property)(0,n)" to denote reflexive-transitive closure, and later use "property+" to indicate transitive closure
- used SPARQL Property Paths notation [10]: "(prop1 | prop2)" instead of "{prop1 OR prop2}" to indicate alternative (disjunction)

2.1 Interpretation and Graphical Represenatation

This FR can be interpreted as follows, where "(...)*" means "optionally and recursively" i.e. reflexive-transitive closure:

- a Thing (part of another Thing)* is considered to be "from" Place if it:
- is formerly or currently located at Place (that falls within another)*
- or was brought into existence (produced/created) by an Event (part of another)*
  - that happened at Place (that falls within another)*
  - or was carried out by an Actor (who is member of a Group)*
    - who formerly or currently has residence at Place (that falls within another)*
    - or was brought into existence (born/formed) by an Event (part of another)* that happened at Place (that falls within another)*
- or was Moved to/from a Place (that falls within another)*
- or changed ownership through an Acquisition (part of another)*
  - that happened at Place (that falls within another)*



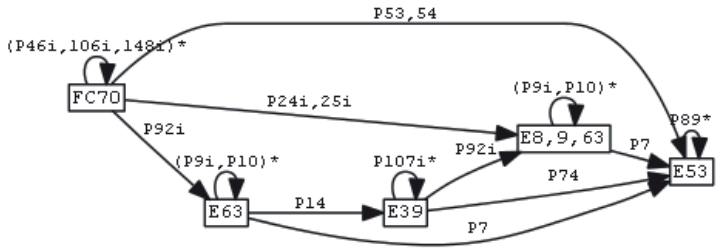
Although defined as a tree of property paths, the FR is better depicted as a network through a simple merge of leaf-level nodes

2.2 Corrections and Rationalization

We reviewed each FR, made some corrections, and rationalized the network. This FR:

- Allowed paths of mixed properties (e.g. P46i,P106i) at the beginning
- Allowed a loop P9i* at E9 (Move forms part of a bigger event) by merging the nodes E8, E9, and the second E63
  - We could even merge the first E63, but then we'd have a back-link, so before traversing P14 must check that the event is E12, E65, or E81 (i.e. the production/creation of a Thing), so that won't lead to simplification
- Allowed P10_falls_within in addition to P9i_forms_part_of (after consultation with the original authors)
- Skipped P26,P27 since these are subproperties of (infer) P7, so it's enough to check for P7

The result is this network:



3 Inverses, Transitive, Parallel-Serial Networks

The example above suggests several implementation considerations:

- Most CRM properties have an inverse and [9] declares them as owl:inverseOf (symmetric properties are their own inverse). FRs use CRM properties in both directions: forward (e.g. P53_has_former_or_current_location) and inverse (P24i_changed_ownership_through), so it's useful to rely on owl:inverseOf inferencing
- FRs use transitive closure (denoted +) to traverse the various "part" hierarchies of CRM (physical object parts, conceptual object parts, sub-places, sub-events), so it's useful to rely on owl:TransitiveProperty inferencing. CRM scope notes suggest that 14 properties (and their inverses) should be transitive: P9 P10 P46 P86 P88 P89 P106 P114 P115 P116 P117 P120 P127 P148. [9] declares them as owl:TransitiveProperty (except P9 P46 that were forgotten, so we declared them). In addition to these "atomic" properties, disjunctions of properties often also need to be declared as transitive.

- FRs often use reflexive-transitive closure (denoted *). However, we have opted not to use reflexive closure in the implementation, since it would generate a lot of trivial facts (self-loops). We use disjunction instead: the iterated property is applied 0 times in the first disjunct, and *n* times in the second
- FRs are defined *mostly* as parallel-serial networks of properties, using SPARQL Property Paths constructs [10]

3.1 Decomposing Thing from Place Into sub-FRs

The example network in section 2.2 can be decomposed into "sub-FRs" as follows. We use prefix FRT for a transitive sub-FR, FRX for a non-transitive sub-FR, and FR for the final result: FR7 "thing from place". A major challenge has been coming up with names for these sub-FRs, so we used numbering from CRM properties

```
    # self-loops and simple disjunctions
FRT_46i_106i_148i := (P46i|P106i|P148i)+
FRT_9i_10 := (P9|P10)+
FRT_107i := P107i+
FRT_89 := P89+
FRX_53_54 := (P53|P54)
FRX_24i_25i := (P24i|P25i)
    # growing fragments
FRX_92i := P92i | P92i/FRT_9i_10
FRX_92i_14 := FRX_92i/P14 | FRX_92i/P14/FRT_107i
FRX_FC70_E8_9_63 := FRX_92i_14/P92i | FRX_24i_25i
FRX_FC70_E8_9_63_P7 := FRX_FC70_E8_9_63/P7 | FRX_FC70_E8_9_63/FRT_9i_10/P7
FRX7 := FRX_53_54 | FRX_FC70_E8_9_63_P7 | FRX_92i_14/P74 | FRX_92i/P7
FRX7_P89 := FRX7 | FRX7/FRT_89
FR7 := FRX7_P89 | FRT_46i_106i_148i/FRX7_P89
```

3.2 Implementing Networks with RDFS and OWL

Parallel-serial networks can be implemented wholly within the RDFS and OWL vocabularies (we express the implementation fragments in RDF Turtle):

Pattern	Construct	Implementation
inverse	prop := ^prop1	prop1 owl:inverseOf prop2.
parallel	prop := prop1|prop2	prop1 rdfs:subPropertyOf prop. prop2 rdfs:subPropertyOf prop.
serial	prop := prop1/prop2	prop owl:PropertyChainAxiom (prop1 prop2).
transitive	prop := prop1+	prop1 rdfs:subPropertyOf prop. prop owl:TransitiveProperty
reflexive-transitive	prop := prop1 prop2*	Converted to the following: prop := prop1 | (prop1/prop2+)

3.3 Type Checking and Conjunctive Properties

The original definition [8] supposes type checks for every node (FC70, E63, etc). So for example the final definition of the target FR should be:

```
x FR7_from_place y := x a FC70_Thing;  x FR7 y; y a E53_Place.
```

Here x,y are variables, "a" stands for rdf:type as usual, and ";" separates triple patterns and stands for conjunction.

In many cases the type checks can be skipped since they are implied by the appropriate property ranges. E.g. all of P53 P54 P7 P47 P89 have range E53, so there is no need to check the type of the final node.

But in some cases type checks are required, e.g. for the "about" FR family that applies to various FCs and is segmented into several FRs: Thing about Thing, Thing about Place, Thing about Actor, etc. If the type check is at the first or last node, it can be added in SPARQL. But if the type check is needed in the middle of a network, we need a conjunctive property.

Unfortunately properties cannot be defined by conjunction in OWL 2 [11]. While the same answer suggests that adding role conjunctions in DLs increases computational complexity significantly, [12] shows conditions under which role conjunction can be added without increase in complexity. In particular, OWL RL can be extended with role conjunctions without any restrictions or increase in complexity, and [13] proposes extending OWL with such capabilities. Such extensions may become available in a future OWL version (OWL 3)

4 OWLIM Rules

Because of the difficulty described in 3.3, we chose to implement FRs using OWLIM Rules [6]. OWLIM [4,5] is a semantic repository by Ontotext Corp that provides high-performance and scalability, comprehensive OWL RL and QL reasoning, custom rules, incremental assert/retract, clustering and other enterprise features.

OWLIM Rules use simple unification: a set of premise triple patterns is checked against the repository, and if it matches, a set of consequence triples is inferred and stored in the repository. The rules are translated to Java bytecode for speed.

The OWLIM Rules syntax is verbose (one line per premise/conclusion). Since we had to define a lot of rules, we defined a simpler syntax (one line per rule, see examples below) that we translate using a simple script. The syntax is similar to N3 Rules, but simpler.

RDFS and OWL2 are implemented in OWLIM using OWLIM Rules. The user loads different rule sets (PIE files) depending on the required reasoning capabilities. We started from RDFS that implements sub-class and sub-property reasoning, and added a bit of OWL that implements inverse and transitive reasoning:

```
p <rdf:type> <owl:TransitiveProperty>; x p y; y p z => x p z
p1 <owl:inverseOf> p2; x p1 y => y p2 x
p1 <owl:inverseOf> p2; x p2 y => y p1 x
```

The implementation of owl:propertyChainAxiom is more complex (using the full rules syntax), mostly because it deals with RDF list iteration. We don't use it for the current implementation:

```
Id: prp_spo2_1
   p <owl:propertyChainAxiom> pc
   start pc last              [Context <onto:_checkChain>]
   ---------------------------
   start p last
Id: prp_spo2_2
   pc <rdf:first> p
   pc <rdf:rest> t            [Constraint t != <rdf:nil>]
   start p next
   next t last                [Context <onto:_checkChain>]
   ---------------------------
   start pc last              [Context <onto:_checkChain>]
Id: prp_spo2_3
   pc <rdf:first> p
   pc <rdf:rest> <rdf:nil>
   start p last
   ---------------------------
   start pc last              [Context <onto:_checkChain>]
```

Then we added specific rules for the FRs. We used a Literate Programming style to intersperse FR definitions and discussions with FR implementation in our wiki, then weaved the final FR rules using a simple script.

4.1 Benefits of OWLIM Rules

The important benefits of OWLIM Rules used in our implementation are:

- Speed: OWLIM uses forward-chaining materializing inference, so consequences are stored in the repository and query answering is very fast. Custom rules are treated just like system rules.
- Rules are "reversible": when a triple is retracted, all relevant rules are checked. If an inferred triple matches the consequences and there are no other triples that support it, the triple is retracted as well. This suppors incremental retract and is extremely important for high-update use cases such as BBC World Cup, BBC Olympics, and ResearchSpace.
- Support conjunctive checks, i.e. overcome the problem described in section 3.3

4.2 Disadvantages of OWLIM Rules

The main disadvantages of OWLIM rules are:

- They are not flexible: every time the rule set is changed, the repository needs to be reloaded from scratch. In contrast, once the RDFS/OWL vocabularies are implemented as rules (see section 4), adding a meta-property (e.g. owl:TransitiveProperty or owl:inverseOf) recomputes all relevant consequences dynamically.
- They are proprietary to OWLIM. Ontotext is considering the implementation of proposed standard rule languages in future OWLIM versions.
- They don't support negation in a real sense (e.g. one can check that a rule variable is not bound to a specific class, but cannot check that a variable does not have a specific type or one of its sub-classes). Implications of this are discussed in sections 5.1 and 6.

5 Results and Performance

Once each FR is depicted as a diagram similar to the one in 2.2, the implementation as OWLIM rules is straightforward if tedious. E.g. the first line in the decomposition shown in 3.1 is implemented as these 3 rules ("rso" stands for "ResearchSpace Ontology"):

```
x <crm:P46i_forms_part_of> y => x <rso:FRT_46i_106i_148i> y
x <crm:P106i_forms_part_of> y => x <rso:FRT_46i_106i_148i> y
x <crm:P148i_is_component_of> y => x <rso:FRT_46i_106i_148i> y
```

We have implemented 11 FRs of Thing:

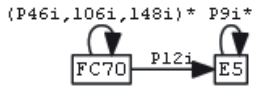
- refers to or is about Place: FR67_refers_to_or_is_about
- from Place: FR7_from_place
- is/was located in Place: FR53_is_was_located_in
- has met Actor: FR12_has_met
- by Actor: FR14_by
- refers to or is about Event: FR67_about_event
- has met Event: FR12_was_present_at
- is made of Material: FR45_is_made_of
- is/has Type: FR2_has_type
- used technique: FR32_used_technique
- identified by Identifier: FR1_identified_by

This took 86 OWLIM rules and 10 axioms. They use 44 source properties (from CRM0 and define and use 26 intermediate properties (sub-FRs, see 3.1).

5.1 Bug in Thing has met Event

We found a "bug" in the definition of Thing has met Event (FR12_was_present_at) that causes quadratic growth and exponential slowdown of data loading. The rule is defined benignly enough:

```
FC70_Thing --FR12_was_present_at-> E5_Event :=
  FC70_Thing --(P46i_forms_part_of | P106i_forms_part_of | P148i_is_component_of)* ->
    FC70_Thing --P12i_was_present_at-> E5_Event:
      E5_Event --P9i_forms_part_of*-> E5_Event
```



ResearchSpace currently deals with RKD and British Museum data, and we model an acquisition as an event having several of these types:

- E8_Acquisition: changes the current owner
- E10_Transfer_of_Custody: changes the current keeper
- E80_Part_Removal: removes the object from the old collection
- E79_Part_Addition: adds the object to the new collection

The acquisition is an event at which meet the object, buyer, seller, old collection and new collection. The object is part of the old collection (before the acquisition) and part of the new collection (after the acquisition). Because P46i_forms_part_of is included in the definition, this causes all objects in a collection to have met (witnessed) the acquisition of all other objects in the collection. This is logically undesirable:

- If Thing2 was added to Collection after Thing1, it's causally impossible for Thing2 to be present at the acquisition of Thing1
- If Thing2 was added to Collection before Thing1, one **could** say Thing2 quietly observed the addition of Thing1, but that is not really useful

More importantly for us, this is computationally very expensive for a large collection such as the British Museum that has over 1.5M objects.

We considered fixing the problem by adding a clause that the target of P46i_forms_part_of is not E78_Collection. However, OWLIM rules don't support true negation, so for the time being we've simply removed P46i from the definition, since our data does not deal with object parts.

5.2 Performance

Concerns were expressed that materializing sub-FR triples may increase the repository size too much and slow it down. We have preliminary performance results that are very promising and dispel these fears:

- A small repository of 11 Rembrandt paintings had 1.5M triples, including about 0.5M object triples (complex data about each painting, researches, documents, etc) and 1M thesaurus triples (people, places, etc). The FRs added only 25.8k triples, which is 1.7% of the total data or 5.1% of the object data.
- A large repository of over 1.5M British Museum objects and about 200M triples performs FR searches with no noticeable slow-down.

5.3 Benefits Compared to Straight SPARQL

To appreciate the query simplification that FRs afford, compare this simple query using the FR "Thing from Place" defined in sec. 3:

```
select * {?t FR7_from_place ?y}
```

To a "straight SPARQL" query:

```
select ?t ?p2 {
?t a FC70_Thing. ?t (P46i_forms_part_of* | P106i_forms_part_of* | P148i_is_component_of*) ?t1.
 {?t1 (P53_has_former_or_current_location | P54_has_current_permanent_location) ?p1}
 UNION
 {?t1 P92i_was_brought_into_existence_by ?e1. ?e1 P9i_forms_part_of* ?e2.
    {?e2 P7_took_place_at ?p1}
    UNION
    {?e2 P14_carried_out_by ?a1.
     ?a1 P107i_is_current_or_former_member_of* ?a2.
      {?a2 P74_has_current_or_former_residence ?p1}
      UNION
      {?a2 P92i_was_brought_into_existence_by ?e3. ?e3 P9i_forms_part_of* ?e4.
       ?e4 P7_took_place_at ?p1}}}
 UNION
 {?t2 P25i_moved_by ?e5. ?e5 (P26_moved_to | P27_moved_from) ?p1}
 UNION
 {?t2 P24i_changed_ownership_through ?e6.
  ?e6 P9i_forms_part_of ?e7. ?e7 P7_took_place_at ?p1}.
?p1 P89_falls_within* ?p2}
```

Even though it uses SPARQL 1.1 shortcut notation (Property Paths), the query is complex. It is also expensive, since it considers many alternative paths. When you consider that FRs are usually used in combination, the resulting queries become too complex. K.Tzompanaki reports that an FR implementation approach using straight SPARQL queries quickly becomes hard to manage (personal communication).

6 Summary and Future Work

We presented an implementation of CRM Search based on the "Fundamental Concepts" and "Fundamental Relations" approach [2]. FC and FRs serve as a "search index" over complex CRM semantic networks and allow the user to use a simpler query vocabulary.

Our implementation uses OWLIM Rules and was done over large repositories of Cultural Heritage objects. We describe the technical details, problems and difficulties encountered, benefits and disadvantages of using OWLIM rules, and preliminary performance results. We provided implementation experience that can be valuable for further implementation, definition and maintenance of CRM FRs.

Future work in this direction can include:

- Implement more FRs in collaboration with the 3D COFORM project. This includes more FRs of Thing, as well as FRs of other Fundamental Concepts (Person, Event, etc) that are not yet defined.
- Automate the discovery of shared sub-FRs to facilitate the implementation
- Take care of complications related to negation

7 References

1. CIDOC CRM website, http://www.cidoc-crm.org
2. Katerina Tzompanaki, Martin Doerr: A New Framework for Querying Semantic Networks. FORTH technical report TR419, May 2011
3. ResearchSpace project, http://www.researchspace.org
4. OWLIM website, http://www.ontotext.com/owlim
5. Barry Bishop, Atanas Kiryakov, Damyan Ognyanoff, Ivan Peikov, Zdravko Tashev, Ruslan Velkov, OWLIM: A family of scalable semantic repositories, Semantic Web Journal, Volume 2, Number 1, 2011.
6. Barry Bishop, Spas Bojanov. Implementing OWL 2 RL and OWL 2 QL rule-sets for OWLIM. Proc. of 8th International Workshop on OWL: Experiences and Directions (OWLED 2011), San Francisco, USA, June 5-6, 2011, CEUR-WS.org, ISSN 1613-0073
7. FP7 project 3D COFORM, http://www.3d-coform.org
8. Katerina Tzompanaki, Martin Doerr: Fundamental Categories and Relationships for intuitive querying CIDOC-CRM based repositories, Technical Report ICS-FORTH/TR-429, April 2012, http://www.cidoc-crm.org/docs/TechnicalReport429_April2012.pdf
9. Erlangen CRM mapping of CRM to OWL, http://erlangen-crm.org
10. SPARQL Property Paths, http://www.w3.org/TR/sparql11-property-paths
11. Answer on SemanticWeb.com, http://answers.semanticweb.com/questions/11602/-property-intersection-impossible-in-owl-2-full
12. Sebastian Rudolph, Markus Krötzsch, Pascal Hitzler: Cheap Boolean Role Constructors for Description Logics. Proceedings of 11th European Conference on Logics in Artificial Intelligence (JELIA 2008), p362-374, LNAI 5293, Sep 2008
13. Pascal Hitzler, Suggestions for OWL 3, Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2009), Oct 2009. CEUR 529, http://ceur-ws.org/Vol-529/owled2009_submission_6.pdf