Types and Annotations for CIDOC CRM Properties

Vladimir Alexiev

Ontotext Corp, 135 Tsarigradsko Shosse Blvd, Sofia, Bulgaria
vladimir.alexiev@ontotext.com

Abstract. The CIDOC CRM provides an extensive ontology for describing entities and properties appearing in cultural heritage (CH) documentation, history and archeology. CRM provides some means for describing information about properties (property types, attribute assignment, and "long-cuts") and guidelines for extending the vocabulary.
However, these means are far from complete, and in some cases there is little guidance how to "implement" them in RDF. In this article we outline the problems, relate them to established RDF patterns and mechanisms, and describe several implementation alternatives.

Keywords: cultural heritage, CIDOC CRM, properties, attribute assignment, attribution, RDF reification, property reification

1 Introduction

The CIDOC Conceptual Reference Model (CRM)[1] provides an ontology for describing the implicit and explicit entities and properties appearing in cultural heritage (CH) documentation, history and archeology (such as that published by galleries, libraries, archives, museums). CRM is the culmination of over 10 years of work and is an official standard ISO 21127:2006. CRM is intended to promote a shared understanding of cultural heritage information by providing a common semantic framework that any cultural heritage information can be mapped to. It is intended as a common language for domain experts and implementers and to provide the "semantic glue" needed to mediate between different sources of CH information.

In history, culturology and art research it is very important to capture not just statements (facts or suppositions), but also additional information about them, such as:

- Who said what when
- Roles and qualifications of relations, e.g. "Michelangelo (E21 Person) performed (P14B) the painting of the Sistine Chapel (E7 Activity) in the role of master craftsman (E55 Type)"

[1] http://www.cidoc-crm.org

- Other data about relations. E.g. consider the situation "The painting Bathing Susanna (E18 Physical Thing) changed ownership through (P24B) an auction (E8 Acquisition) as lot number 15". The lot should be modeled as an attribute (E42 Identifier) of the relation P24 between the painting and acquisition. It cannot be attached to the painting directly, since it may have been offered at several auctions. Nor can it be attached to the acquisition directly, since often several paintings are sold through one auction
- The status of a statement (fact, proposed, disputed, etc)
- Comments or discussions about a statement
- Relations to other data that justifies or disproves a statement
- Indication of probability or uncertainty

CRM data is usually represented in semantic web format (RDF), comprising graphs made of triples (statements). The triples connect nodes (URIs, literals or blank nodes) using properties identified by URIs. We should be careful to distinguish between a property and a property instance (statement). The problem of providing additional information about statements is not new and there are some established RDF patterns and mechanisms that we can use.

1.1 ResearchSpace Annotation Needs

The ResearchSpace project (RS)[2] is funded by the Andrew W. Mellon foundation, designed and administered by the British Museum (BM), and developed by a consortium led by Ontotext Corp. The project aims to support collaborative internet research, information sharing and web applications for the cultural heritage scholarly community (initially art researchers in the domain of classic paintings). The RS hosted environment intends to provide: Data, Digital analysis and Annotation tools, Collaboration tools, Semantic RDF data sources, Image annotation and collaboration tools, etc. Since RS wants to address a wide variety of data related to cultural heritage research, CRM is the most appropriate conceptual model and data schema for the project.

A core RS need is to allow an art researcher to annotate pretty much any value of any cultural object, e.g. the creator (Person who carried out the Production, also called "attribution"), the creation year, object type, material, dimensions, etc. Annotations are intended to capture Research Discourse and include the following abilities:

- provide comments about any field
- reply to someone else's comments, forming a discussion
- link another semantic object by embedding it in a comment
- link a field of another semantic object to use as justification. E.g. the dating of Rembrandt's "Bathing Susanna" is established as 1636 because a drawing reproduction by Willem de Poorter is signed and dated 1636.
- dispute old value

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

- propose new value, with justification in the form of comment or link to another object

In the process of designing RS, mapping existing museum data to CRM, and designing annotation schemas we faced several issues with CRM's ability to represent additional data about statements that led to this article.

2 CRM Means and Problems

CRM provides some means for describing information about properties (property types, attribute assignment and long-cuts). They are far from complete, and in some cases there is little guidance how to "implement" them in RDF. In this section we outline these means and the related problems.

2.1 Property Types

CRM includes several "properties of properties" that can distinguish between different "types" for a property. E.g. P3.1 is shown on the figure above and can distinguish between various notes (name, title, description, etc).
The full list of property types is: P3.1 has type, P14.1 in the role of, P16.1 mode of use, P19.1 mode of use, P62.1 mode of depiction, P67.1 has type, P69.1 has type, P102.1 has type, P130.1 kind of similarity, P136.1 in the taxonomic role, P137.1 in the taxonomic role, P138.1 mode of representation, P139.1 has type.

All these have another property as their range. Since "properties of properties" cannot be implemented in RDF directly, CRM recommends to implement them as sub-properties (e.g. P3a_name, P3b_description, etc).

Problem: This approach is not convenient if the specific relations are numerous and come from a thesaurus, e.g.:

- The Getty Union List of Artist Names (ULAN)[3] includes numerous subtypes for artist relations (associatedWith), such as: teacherOf, patronWas, etc
- The BM collection database includes 14 vocabularies for association codes (e.g. Acquisition Person, Production Person, Production Place) with over 230 codes.

If these 230 codes are implemented as 230 sub-properties, then an application will need to deal with all of them, which is significant complexity (in comparison, all of CRM has 143 properties)! Every time a code is added to a database, the corresponding ontology and data conversions would need to be modified.

RDF schemas are flexible, since data and metadata is all expressed as triples, and SPARQL allows you to query for all relations between objects even without knowing the relation URIs. But a thesaurus of types is more flexible still. E.g. in a search use case, it's better to let the user select (or multi-select) values coming from a thesaurus list, rather than property URIs. The CRM recommendation to use sub-properties con-

[3] http://www.getty.edu/research/tools/vocabularies/

verts data (flexibility) to schema (fixedness), that's why we consider it as problematic. Furthermore, it doesn't help if you need to attach other data (like the "lot number" example from sec.1)

2.2 Attribute Assignment

The CRM entity E13 Attribute Assignment goes a long way to provide statement annotation capabilities.

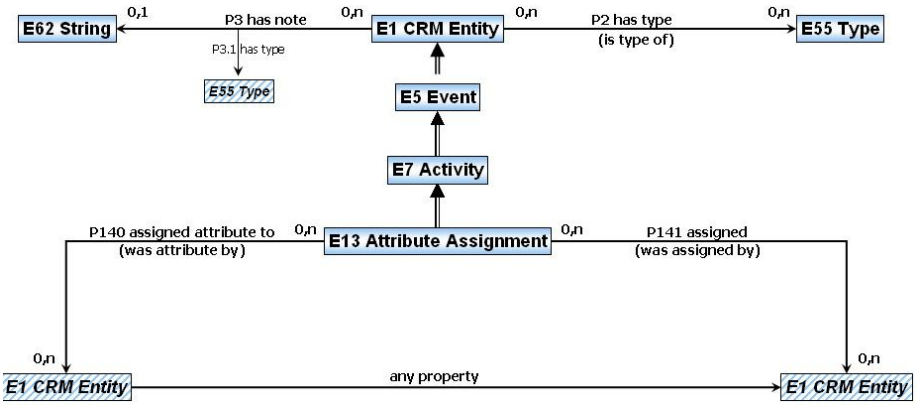

Fig. 1. E13_Attribute_Assignment

Fig.1 is taken from the CRM Graphical Representation[4]. Double arrows link sub-classes, single arrows are properties, and the thin arrow "P3.1" is a property type (described in the next section). E13 has fields (some of them inherited) for recording the following:

- who: P14_carried_out_by from E7_Activity
- when: P4_has_time-span from E5_Event
- said what: P3_has_note
- about what (subject): P140_assigned_attribute_to
- what value (object): P141_assigned
- "did" what, e.g. Dispute, Propose; Agree, Disagree, etc: a P2_has_type sub-property, from E1_CRM_Entity),
- what was the outcome, i.e. "dispositions" such as Proposed, Approved, Rejected, Published: another P2_has_type sub-property

[4] http://cidoc-crm.org/cidoc_graphical_representation_v_5_1/graphical_representaion_5_0_1.html

Problems:

1. Attribute Assignment doesn't mention the property being annotated (called "any property" in the figure). This means for example that one cannot annotate a specific authorship statement in the case of a multi-author work
2. It cannot annotate primitive values (numbers, strings). The range of P141 excludes E59_Primitive_Value, which is outside the E1_CRM_Entity class hierarchy

For these reasons we proposed to the CRM SIG that the range of P141 should be "property", just like the domain of Pn.1 is "property".

Regarding 1, M.Doerr proposed[5] in March 2012 to use P2_has_type for this purpose, but this would make CRM properties be of type E55_Type. This proposal has not been explored further and not established as practice.

2.3 Short-cuts and Long-cuts

CRM considers some properties as shortcuts of longer, more comprehensively articulated paths (we call them "long-cuts") that connect the same nodes through intermediate nodes.

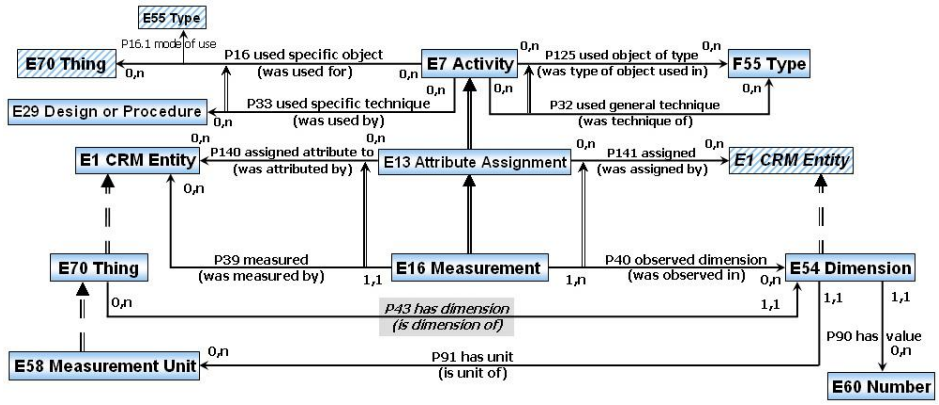

Fig. 2. E16_Measurement

Fig.2 (also from the CRM Graphical Representation) gives a good example:

- **Short-cut**: E70_Thing --P43_has_dimension-> E54_Dimension
- **Long-cut**: E1_CRM_Entity --P39B_was_measured_by-> E16_Measurement --P40_observed_dimension-> E54_Dimension. It allows us to record additional information about the Measurement, e.g. when it was made, by whom, etc

E13 Attribute Assignment is the "paradigmatic" long-cut, and indeed 4 of the long-cut classes are derived from it (below we show "long-cut class: short-cut property"):

[5] http://lists.ics.forth.gr/pipermail/crm-sig/2012-March/001762.html

- E14 Condition Assessment: P44 has condition
- E15 Identifier Assignment: P1 is identified by / P48 has preferred identifier
- E16 Measurement: P43 has dimension
- E17 Type Assignment: P2 has type

But many classes involved in long-cuts are not derived from Attribute Assignment, since they describe a more complex business situation than assigning an attribute to an object:

- E8 Acquisition: P51 has former or current owner, P52 has current owner
- E9 Move: P53 has former or current location, P55 has current location
- E10 Transfer of Custody: P49 has former or current keeper, P50 has current keeper
- E36 Visual Item: P62 depicts
- E53 Place: P56 bears feature
- E53 Place, E46 Section Definition: P8 took place on or within
- E46 Section Definition: P59 has section
- E12 Production / E65 Creation: P130 shows features of

The most involved of these situations can have two short-cuts, one of which is 2-step:

- **Short-cut**: E4 Period --P8 took place on or within-> E19 Physical Object. Here we just state that a period/event happened on an object, e.g. "Mutiny took place on Starship Enterprise"
- **Long-cut**: E4 Period --P7 took place at-> E53 Place -P59i is located on or within-> E18 Physical Thing. Here we consider a specific section of the object as a Place, e.g. "Mutiny took place at Upper Deck located on Starship Enterprise"
- **Longer-cut**: E53 Place --P87 is identified by-> E44 Place Appellation < E46 Section Definition --P58i defines section-> E18 Physical Thing. Here we consider P59 itself as a shortcut, and use E46 to define or describe the section of the object as a Place. E.g. "Mutiny took place at a place that is identified by a Section Definition that defines the location Upper Deck as a section located on Starship Enterprise"

Problems:

1. CRM states: "*An instance of the fully-articulated path always implies an instance of the shortcut property*". We disagree, since the long-cut may have a status of Tentative, Proposed, Suggested or even Formerly Thought To Be (i.e. not currently considered true), while the short-cut (without the ability to attach status information to it) should be considered true.
2. Documenting and qualifying properties is important in CH research. E.g. documenting "P14 carried out by" when it concerns the authorship of a work of art is called "attribution" and is a crucial activity in art research. CRM states: "*E13 Attribute Assignment allows for the documentation of how the assignment of any property came about, and whose opinion it was, even in cases of properties not explicitly characterized as shortcuts*". Unfortunately this is not true, because E13 doesn't mention the property being annotated, as explained in sec.2.2.

3. The domains and ranges of short-cuts and long-cuts do not always agree. As can be seen in the figure above, you can Measure any E1 Entity, but you can say "P43 has dimension" only about E70 Thing (which are persistent things, different from Actors). For example:

   – You cannot say "the car moved at 70 km/h" (E4 Period is not persistent), you'd have to say something like "a Measurement consisting of taking a look at the odometer P39 measured the car's movement and P40 observed dimension of 70km/h"
   – You cannot say "this Group has 70 members" (Group is not a Thing), you'd have to say "a Measurement consisting of counting P39 measured the Group's size and P40 observed dimension of 70 persons". Worse yet would be to create 70 anonymous entities E21 Person and make them P107i current or former members of the Group.

We have taken the last issue to the CRM SIG, but the full study of short-cuts vs long-cuts is still forthcoming.

2.4 Extending CRM

CRM defines guidelines for extending CRM in a compatible way:

1. All extension classes should be sub-classes of CRM classes.
2. All extension properties
   (a) Should be sub-properties of CRM properties, OR
   (b) Are part of a long-cut for which a CRM property is the short-cut.

The purpose of these guidelines is to allow applications that "understand" CRM but not the extension to still make queries and get useful results. Under the above conditions, the relevant CRM statements can be inferred automatically:

- Sub-class and sub-property is within RDFS,
- Short-cuts under 2(b) can be inferred with rules or property paths, e.g.

```
P43_has_dimension owl:propertyChainAxiom
 (P39B_was_measured_by P40_observed_dimension).
```

The rest of the article deals with various ways of constructing long-cuts (approach 2(b)) in an organized and explicit way.

CRM recommends to implement Property Types using approach 2(a), but we have criticized this in sec.2.1.

3 Solution Alternatives

The problem of adding more data to statements is not unique to CRM. It has been studied to some extent by the RDF community, and some patterns and mechanisms

have emerged. We consider these implementation alternatives below and provide some analysis.

3.1 Long-cuts or Split Properties

A simple way to make statements "addressable" is to split properties by introducing an intermediate node, i.e. make long-cuts.

For example, to annotate these two statements (in Turtle notation) about the production of the Sistine Chapel:

<obj/prod> P14F_carried_out_by <person/Michelangelo>.
<obj/prod> P14F_carried_out_by <person/GiovanniUnderstudy>.

We could split P14 into P14F1 and P14F2. Types, data and annotations can be attached easily to the intermediate node:

<obj/prod> P14F1_carried_out_role <obj/prod/role/1>, <obj/prod/role/2>.
<obj/prod/role/1> a E200_Production_Role;
 P14F2_carried_out_actor <person/Michelangelo>;
 P2F_has_type <production/role/master-craftsman>;
 P200F_has_probability <probability/certain>.
<obj/prod/role/2> a E200_Production_Role;
 P14F2_carried_out_actor <person/GiovanniUnderstudy>;
 P2F_has_type <production/role/understudy>;
 P200F_has_probability <probability/proposed>.

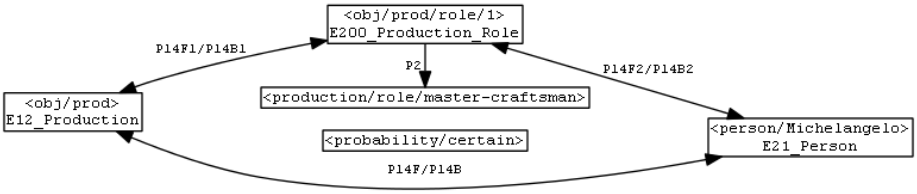

Fig. 3. Split Property P14 to Create a Long-Cut

Fig.3 shows this approach, assuming appropriate inverse properties are defined (P14B1, P14B2), showing only the first actor (<person/Michelangelo), and omitting the P200 arrow for simplicity. Considerations:

- Pro: it's a simple domain-specific way that can keep programming intuitive
- Pro: we can define simple rules to infer the short-cut, so this can be a CRM-compatible extension
- Pro: the intermediate node can be inserted "on demand" only when needed, i.e. if a researcher wants to annotate the property, and/or the property needs a type or attribute
- Cons: multiplies both the number of statements and property types by 3 (for those properties that need annotations/types)

- Cons: property-specific, i.e. not a universal solution

It is a worthy task to consider adding to CRM a limited set of such long-cut properties and intermediate classes. Careful consideration should be given to the proper level at which they should be defined. E.g. P14_carried_out_by is too low in the property hierarchy. It is better to make long-cut for its parent P11_had_participant or even grandparent P12_occurred_in_the_presence_of, since the role "carried out by" can be added to the intermediate node using P2_has_type.

3.2 Statement Reification

RDF Reification is described in the RDF Specification[6] and Primer[7]. It is a standard vocabulary to represent statements as explicit nodes:

- rdf:Statement: the intermediate node, holding the following properties
- rdf:subject: points to the triple's subject (we also define inverse ext:subjectOf)
- rdf:predicate: URI of the property
- rdf:object: points to the triple's object (we also define inverse ext:objectOf)

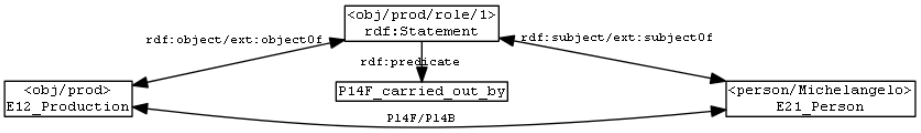

Fig. 4. RDF Statement Reification

Fig.4 shows this approach, omitting P2 and P200 for simplicity. rdf:Statement is similar to CRM's E13_Attribute_Assignment, but adds explicitly the property URI. The graph is also similar to the Split Property on Fig.3. Considerations:

- Pro: the intermediate node can be inserted "on demand" only when needed
- Pro: generic, i.e. can be a universal solution
- Pro: provides a more flexible approach than named graphs (see sec.3.4), since rdf:object can be omitted, e.g. in order to propose a new value without considering the old value.

For example, RS uses this approach (together with the OAC ontology[8]) to represent annotations about statements.

[6] http://www.w3.org/TR/rdf-mt/#Reif
[7] http://www.w3.org/TR/rdf-primer/#reification
[8] http://www.openannotation.org

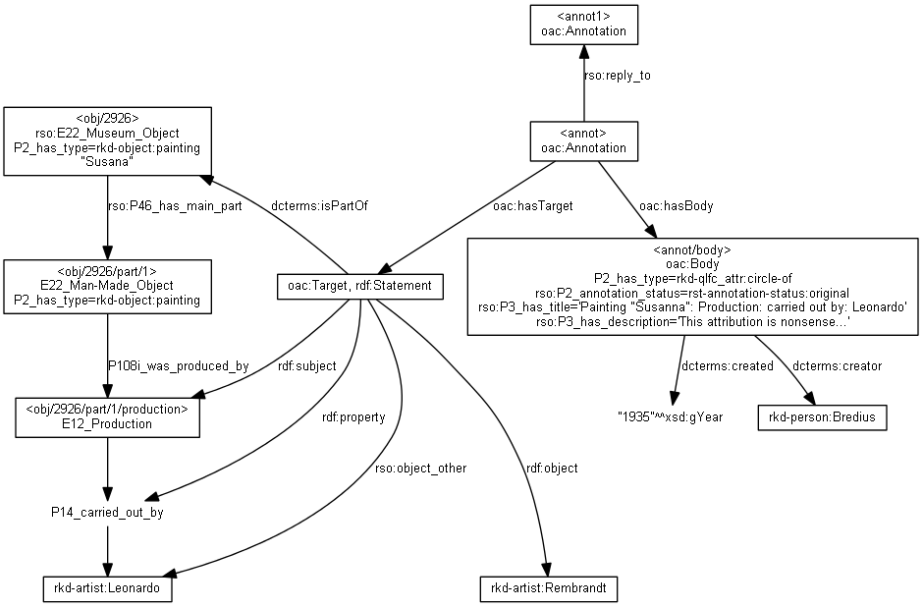

Fig. 5. Using RDF Reification and OAC in ResearchSpace

Fig.5 depicts a situation where Bredius in 1935 stated that Rembrandt (and not Leonardo) created the painting Susana:

- Cons: programming is less intuitive, since generic properties and intermediate node type are used
- Cons: reification is deprecated by the semantic web community because it works only on the syntactic level and fails to make important semantic distinctions
  - Tim Berners-Lee on the W3C semantic web mailing list (Jan 2007): "Reification is not just incomplete but broken… I think reification should be dropped from a future RDF spec."
  - Draft charter (2010) for the RDF update working group: Considers deprecating reification officially

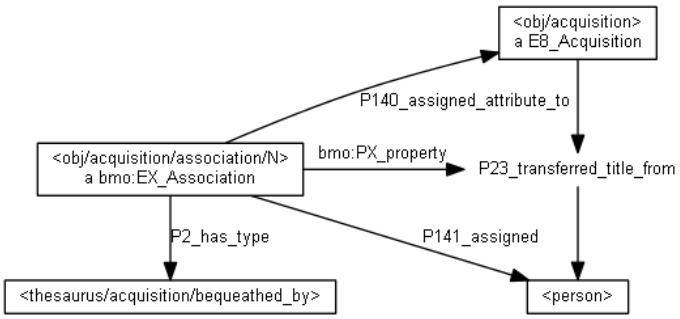

Fig. 6. Representing Bequeathal Using EX_Association and PX_property

Fig.6 shows an extension EX_Associaton (subclass of E13) that adds PX_property. It is used by BM to extend CRM properties (e.g. P23_transferred_title_from) to represent more specific situations such as Bequeathal (transferring title without any remuneration)

3.3 Property Reification

The Property Reification Vocabulary (PRV)[9] allows one to describe reification (short-cut vs long-cut) patterns explicitly. They are described using up to 5 attributes of class prv:PropertyReification. It's important that the description is independent of the data.

Table 1. Examples of prv:PropertyReification Records

reification_class	shortcut | shortcut_property	subject_property	object_property
E13_Attribute_Assignment		P140_assigned_attribute_to	P141_assigned
E16_Measurement	P43_has_dimension	P39_measured	P40_observed_dimension
E36_Visual_Item	P62_depicts	P65i_is_shown_by	P138_represents
E36_Visual_Item	P62i_is_depicted_by	P138_represents	P65i_is_shown_by
E53_Place	P8_took_place_on_or_within	P7i_witnessed	P59i_is_located_on_or_within
E46_Section_Definition	P59i_is_located_on_or_within	P87i_identifies	P58i_defines_section
ext:E200_Production_Role	ext:P14F_carried_out_by	ext:P14B1	ext:P14F2_carried_out_actor
rdf:Statement	rdf:predicate	rdf:subject	rdf:object
bmo:EX_Association	bmo:PX_property	P140_assigned_attribute_to	P141_assigned

Rather than giving complex explanations, Table 1 illustrates the use of PRV by defining one prv:PropertyReification record for each of the examples given in previous sections. The first row are prv: properties, the rest are crm: unless specified otherwise:

- A record with prv:shortcut is specific, in that it applies only to that shortcut
- A record with prv:shortcut_property is generic, in that it can point to different properties
- The trouble with E13 is that it has neither prv:shortcut, nor prv:shortcut_property

We have to be careful about the direction of properties (forward/inverse) and their role (subject_property/object_property). It is best to define reifications for both directions, which we have shown for E36_Visual_Item and P62/P62i only. Please note that when the short-cut (P62) changes direction, the long-cut properties (P65, P138) change role but not direction (they always point away from the reification node).

We can infer the short-cuts using the following 2 rules (in N3 Rules[10] notation)

[9] http://smiy.sourceforge.net/prv/spec/propertyreification.html
[10] http://www.w3.org/2000/10/swap/doc/Rules

```
{ ?pr a prv:PropertyReification ;
    prv:reification_class ?rc ;
    prv:shortcut ?sc ;
    prv:subject_property ?sp ;
    prv:object_property ?op .
  ?r a ?rc ;
    ?sp ?s ;
    ?op ?o .
}  => { ?s ?sc ?o } .
```

```
{ ?pr a prv:PropertyReification ;
    prv:reification_class ?rc ;
    prv:shortcut_property ?scp ;
    prv:subject_property ?sp ;
    prv:object_property ?op .
  ?r a ?rc ;
    ?scp ?sc ;
    ?sp ?s ;
    ?op ?o .
}  => { ?s ?sc ?o } .
```

These are slightly modified variants of the PRV *shortcut relation rule*: they require one of prv:shortcut or prv:shortcut_property to be present, not both. Please note that the presence of both is needed only by PRV's *property reification rule*, i.e. inferring the long-cut intermediate node; but we don't consider such inference to be useful (the very purpose of the long-cut is to hold more data than the short-cut).

We verify these rules by applying them to two different reifications of P43_has_dimension:

```
{ <PR1> a prv:PropertyReification ;
    prv:reification_class E16_Measurement ;
    prv:shortcut P43_has_dimension ;
    prv:subject_property P39_measured ;
    prv:object_property P40_observed_dimension.
  ?measurement a E16_Measurement ;
    P39_measured ?thing ;
    P40_observed_dimension ?dim .
}  => { ?thing P43_has_dimension ?dim } .
```

```
{ <PR2> a prv:PropertyReification ;
    prv:reification_class rdf:Statement ;
    prv:shortcut_property rdf:predicate ;
    prv:subject_property rdf:subject ;
    prv:object_property rdf:object .
  ?statement a rdf:Statement ;
    rdf:property P43_has_dimension ;
    rdf:subject ?thing ;
    rdf:object ?dim .
}  => { ?thing P43_has_dimension ?dim }.
```

Indeed, we see the appropriate short-cut will be inferred in both cases.

A worthy task is to create PRV records for all CRM short-cut properties (and all extension properties if an extension is used), so that semantic applications can discover them automatically.

3.4 Named Graphs

Named Graphs [1,2] extend the notion of RDF triples to quads. The additional element (called Context or Graph) can be used for any application purpose (e.g. attach provenance information, provide access control, etc). The Linked Data Patterns book[11] states: "Named graphs provide a more manageable alternative to reification for handling versioning and provenance in datasets".

[11] http://patterns.dataincubator.org/book/named-graphs.html

Usually the Context is used to group many triples into a graph, but in our case we'll use it for single triples. We can redo the example from sec.3.1 using TriG[12] (a variant of Turtle extended with named graphs) like this:

```
<obj/prod/role/1> {
  <obj/prod> P14F_carried_out_by <person/Michelangelo>
}

<obj/prod/role/2> {
  <obj/prod> P14F_carried_out_by <person/GiovanniUnderstudy>
}

{ # default named graph
  <obj/prod/role/1> a E200_Production_Role;
    P2F_has_type <production/role/master-craftsman>;
    P200F_has_probability <probability/certain>.
  <obj/prod/role/2> a E200_Production_Role;
    P2F_has_type <production/role/understudy>;
    P200F_has_probability <probability/proposed>.
}
```

SPARQL supports named graphs,[13] and queries can access the data and graph layers freely. For example the following query will return the two persons, with their role and probability:

```
SELECT ?person ?type ?probability WHERE {
  GRAPH ?role {<obj/prod> P14F_carried_out_by ?person}.
  ?role P2F_has_type ?type; P200F_has_probability ?probability.
}
```

Considerations about this approach:

- Pro: doesn't require the introduction of additional properties. The use of additional classes (e.g. E200_Production_Role) is optional.
- Pro: better recommended by semantic web practitioners, compared to reification.
- Pro: can apply the same annotation to a group of statements (although this is not very common for CH research discourse)
- Pro: modern semantic repositories support named graphs, so the approach is viable.
- Cons: in many repositories Context is not a first-class citizen like Subject, Predicate and Object. What we mean is that repositories maintain indexes such as SPOC, POSC, OPSC that allow one to find quickly subjects, properties and objects when some of the others are fixed. But fewer repositories maintain indexes where the Context leads (e.g. CSPO). This is not a problem when the number of unique

[12] http://www4.wiwiss.fu-berlin.de/bizer/trig/
[13] http://www.w3.org/TR/rdf-sparql-query/#rdfDataset

Contexts is small, but could become a problem when there is one Context per data triple

- Cons: if Context is used for individual statements, then it is not directly usable for access control and provenance. Additional statements should be used to group the Contexts into bigger graphs, which complicates the model and increases the number of statements.

4 Conclusions and Future Work

We have summarized CRM's means for representing additional data about properties, such as types, probability, status, disposition, qualification, role. We presented various problems associated with these means.

We then presented several RDF implementation alternatives, and considerations pro and cons for each alternative. We illustrated the alternatives with examples, and with real data representation cases from the BM and the RS project.

Much future work remains in order to select the best approach(es) for particular modeling needs, and to standardize the use of these approaches. Such standardization can take the form of adding to CRM, defining CRM extensions, or defining best practices.

Without reaching such agreements, CRM may fall short of its promise to provide a universal language for CH data integration. When more complex research discourse needs to be modeled, the questions we described in this article become important.

References

1. Carroll, J.J., Stickler, P.: TriX: RDF Triples in XML. International Journal on Semantic Web and Information Systems, Vol 1, No 1 (2005)
2. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named Graphs. Journal of Web Semantics, Vol 3, No 4 (2005)