$\times$-*Protégé*
An Ontology Editor for Defining Cartesian Types to Represent n-ary Relations

Christian Willms, Hans-Ulrich Krieger, Bernd Kiefer
German Research Center for Artificial Intelligence (DFKI)
Campus D3 2, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
christian.willms@dfki.de, krieger@dfki.de, kiefer@dfki.de

Abstract

Arbitrary n-ary relations ($n \geq 1$) can, in principle, be realized through binary relations obtained by a reification process which introduces new individuals to which the additional arguments are linked via "accessor" properties. Modern ontologies which employ standards such as RDF and OWL have mostly obeyed this restriction, but have struggled with it nevertheless. In (Krieger and Willms, 2015), we have laid the foundations for a *theory-agnostic* extension of RDFS and OWL and have implemented in the last year an extension of *Protégé*, called $\times$-*Protégé*, which supports the definition of Cartesian types to represent n-ary relations and relation instances. Not only do we keep the distinction between the domain and the range of an n-ary relation, but also introduce so-called *extra* arguments which can be seen as position-oriented unnamed *annotation* properties and which are accessible to entailment rules. As the direct representation of n-ary relations abolishes RDF triples, we have backed up $\times$-*Protégé* by the semantic repository and entailment engine *HFC* which supports tuples of arbitrary length. $\times$-*Protégé* is programmed in Java and is made available under the Mozilla Public License.

Keywords: ontology editor, $\times$-*Protégé*, Cartesian types, n-ary relations, RDF, RDFS, OWL, n-ary Description Logics.

1. Decription Logics, OWL, and RDF

Relations in description logics (DLs) are either unary (so-called *concepts* or *classes*) or binary (*roles* or *properties*) predicates (Baader et al., 2003). As the designers of OWL (Smith et al., 2004; Hitzler et al., 2012) decided to be compatible with already existing standards, such as RDF (Cyganiak et al., 2014) and RDFS (Brickley and Guha, 2014), as well as with the universal RDF data object, the *triple*,

subject predicate object

a unary relation such as $C(a)$ (class membership) becomes a binary relation via the RDF type predicate:

a rdf:type C

For very good reasons (mostly for decidability), DLs usually restrict themselves to decidable function-free two-variable subsets of first-order predicate logic. Nevertheless, people have argued ver early for relations of more than two arguments (Schmolze, 1989), some of them still retaining decidability and coming up with a better memory footprint and a better complexity for the various inference tasks (including querying) than their triple-based relatives (Krieger, 2012; Krieger, 2014). This idea conservatively extends the standard *triple-based* model towards a more general *tuple-based* approach ($n + 1$ being the arity of the *predicate*):

subject predicate object$_1$ $\ldots$ *object*$_n$

Using a standard relation-oriented notation, we often interchangeably write

$p(s, o_1, \ldots, o_n)$

Here is an example, dealing with *diachronic* relations (Sider, 2001), relation instances whose object values might change over time, but whose subject values coincide with each other. For example (quintuple representation),

peter marriedTo liz 1997 1999
peter marriedTo lisa 2000 2010

or (relation notation)

marriedTo(*peter, liz, 1997, 1999*)
marriedTo(*peter, lisa, 2000, 2010*)

which we interpret as the (time-dependent) statement that *Peter* was married to *Liz* from 1997 until 1999 and to *Lisa* from 2000–2010.

In a triple-based setting, semantically representing the same information requires a lot more effort. There already exist several approaches to achieve this (Welty and Fikes, 2006; Gangemi and Presutti, 2013; Krieger and Declerck, 2015), all coming up with at least one brand-new individual (introduced by a hidden existential quantification), acting as an *anchor* to which the object information (the range information of the relation) is bound through additional properties (a kind of *reification*). For instance, the so-called *N-ary relation encoding* (Hayes and Welty, 2006), a W3C best-practice recommendation, sticks to binary relations/triples and uses *container* objects to encode the range information (ppt1 and ppt2 being the *new* individuals):

peter marriedTo ppt1
ppt1 rdf:type nary:PersonPlusTime
ppt1 nary:value liz
ppt1 nary:starts "1997"^^xsd:gYear
ppt1 nary:ends "1999"^^xsd:gYear
peter marriedTo ppt2
ppt2 rdf:type nary:PersonPlusTime
ppt2 nary:value lisa
ppt2 nary:starts "2000"^^xsd:gYear
ppt2 nary:ends "2010"^^xsd:gYear

As we see from this small example, a quintuple is represented by five triples. The relation name is retained, however, the range of the relation changes from, say, Person to the type of the container object which we call here PersonPlusTime.

Rewriting ontologies to the *latter* representation is an unpleasant enterprise, as it requires further classes, redefines property signatures, and rewrites relation instances,

as shown by the marriedTo example above. In addition, reasoning and querying with such representations is extremely complex, expensive, and error-prone.

Unfortunately, the *former* tuple-based representation which argues for additional (temporal) arguments is **not** supported by *ontology editors* today, as it would require to deal with general n-ary relations ($n \geq 2$). *×-Protégé* fills exactly this gap.

2. Further Motivation

×-Protégé supports the definition of Cartesian types, composed from standard OWL classes and XSD datatypes. Given Cartesian types and by keeping the distinction between the *domain* $\mathbb{D}$ and the *range* $\mathbb{R}$ of a binary property p, it is now possible to define $m + n$-ary relations $p \subseteq \mathbb{D}_1 \times \cdots \times \mathbb{D}_m \times \mathbb{R}_1 \times \cdots \times \mathbb{R}_n$.

The deeper reason why it is still useful to separate domain and range arguments from one another is related to the so-called *property characteristics* built into OWL, e.g., symmetry or transitivity. This ultimately allows us to generalize the corresponding entailment rules, by replacing atomic classes with Cartesian types. For instance, entailment rule rdfp4 for *transitive* properties p from (ter Horst, 2005)

$$p(x, y) \wedge p(y, z) \rightarrow p(x, z)$$

can be generalized as ($m = n = o$)

$$p(\times_{i=1}^{m} x_i, \times_{j=1}^{n} y_j) \wedge p(\times_{j=1}^{n} y_j, \times_{k=1}^{o} z_k) \rightarrow p(\times_{i=1}^{m} x_i, \times_{k=1}^{o} z_k)$$

×-Protégé not only keeps the distinction between the *domain* and *range* arguments of a relation, but also provides further distinct *annotation*-like arguments, called *extra* arguments which have been shown useful in various situations and which are accessible to entailment rules of the above kind. Consider a binary *symmetric* property q which we would like to generalize by the concept of *valid time* (the time in which an atemporal statement is true), thus the corresponding entailment rule needs to be extended by two further temporal arguments b and e:

$$q(x, y, b, e) \rightarrow q(y, x, b, e)$$

By assuming that the temporal arguments are part of the domain and/or range of q, we are running into trouble as symmetric properties require the same number of arguments in domain and range position. Thus, we *either* need to adjust this rule, i.e.,

$$q(\underline{x, b, e}, \underline{y, b, e}) \rightarrow q(\underline{y, b, e}, \underline{x, b, e})$$

or assume that b and e have a special "status". We decided for the latter and call such information *extra arguments*. As an example, the former marriedTo relation (a symmetric relation) is of that kind, thus having the following relation signature (assuming a biography ontology with class Person):

$$\underbrace{\text{Person}}_{\textit{domain}} \times \underbrace{\text{Person}}_{\textit{range}} \times \underbrace{\text{xsd:gYear} \times \text{xsd:gYear}}_{\textit{2 extra arguments}}$$

Other *non-temporal* examples of *extra arguments* might involve *space* (or *spacetime* in general), using further XSD custom types, such as point2D or point3D, in order to encode the position of a moving object over time (Keshavdas and Kruijff, 2014).

More linguistically-motivated examples include the *direct* representation of ditransitive and ergative verb frames, including adjuncts (Krieger, 2014). We will present an example of this at the end of Section 7. when defining the quaternary relation obtains. Such kinds of properties are often wrongly addressed in triple-based settings through *relation composition*, applied to the second argument of the corresponding binary relation. This does *not* work in general, but only if the original relation is *inverse functional*.

As a last example, we would like to mention the *direct* representation of *uncertain* statements in medicine or technical diagnosis in an extension of OWL (Krieger, 2016) which is far superior to various encodings described in (Schulz et al., 2014) which have accepted the boundaries of RDF triples in order to be compatible with an existing standard.

3. *Protégé*, *×-Protégé*, and *HFC*

Protégé is a free, open source ontology editor, providing a graphical user interface to define and inspect ontologies (http://protege.stanford.edu). *Protégé* version 4 has been designed as a modular framework through the use of the OSGi framework as a plugin infrastructure (https://www.osgi.org/developer/). For this reason, *×-Protégé* has been implemented as an EditorKitFactory plugin for *Protégé*, replacing the built-in OWL EditorKitFactory. The EditorKit is the access point for a particular type of model (in our case, a model based on n-tuples) to which a GUI has access to.

×-Protégé is divided into three separate components (Figure 1, large right box). The "bottom" layer is realized by *HFC* (Krieger, 2013), a bottom-up forward chainer and semantic repository implemented in Java which is comparable to popular systems such as Jena and OWLIM (http://www.dfki.de/lt/onto/hfc/). *HFC* supports RDFS and OWL reasoning à la (Hayes, 2004) and (ter Horst, 2005), but at the same time provides an expressive language for defining custom rules, involving functional and relational variables, complex tests and actions, and the replacement of triples in favour of tuples of arbitrary length. The query language of *HFC* implements a subset of SPARQL, but at the same time provides powerful custom M:N aggregates ($M, N \geq 1$), not available in SPARQL.

The data read in by *HFC* is preprocessed and transformed into an *×-Protégé* model. Among other things, it contains inheritance hierarchies for classes and properties which are directly used to visualize the ontology in the graphical user interface of *×-Protégé*.

This GUI consists of several workspaces (similar to *Protégé*, version 4.3), presenting the ontology itself, the classes, the properties, and the instances. User actions result in an update of the model and *HFC*'s n-tuple database.

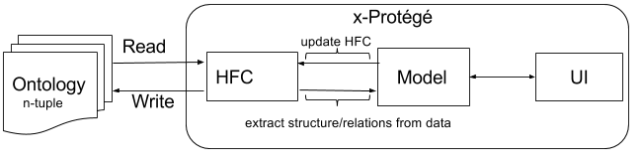

Figure 1: The three-layered structure of *×-Protégé*.

In the next section, we will look into some of these workspaces (or tabs), assuming the marriedTo example from Sections 1. and 2.

4. Class Tab

When starting ×-*Protégé* the class hierarchy consists of a unique, most general type, called Thing+ in the GUI which subsumes every other Cartesian type and which can be formally defined as

$$\text{Thing+} := \bigsqcup_{i=1}^{k} (\text{owl:Thing} \sqcup \text{xsd:AnyType})^i$$

For a given ontology, k is fixed (finite, of course). Initially, Thing+ has two direct subtypes, viz., owl:Thing and xsd:AnyType. *HFC* already provides a set of built-in XSD subtypes, such as xsd:gYear (Gregorian Year) or xsd:int (4 Byte integers), but also defines non-standard datatypes, such as xsd:monetary. As in a pure OWL setting, owl:Thing and xsd:AnyType are incompatible, but xsd:AnyType is made available under Thing+ in order to define Cartesian types, such as xsd:gYear × xsd:gYear for the two extra arguments of the marriedTo relation (or even Person × xsd:gYear × xsd:gYear for the sexternary relation q in Section 2.). This small type hierarchy is depicted in Figure 2.

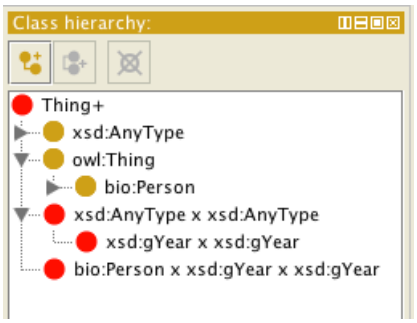

Figure 2: The class hierarchy for the marriedTo example.

Note that the non-singleton Cartesian types are highlighted using red colour and that xsd:gYear × xsd:gYear is correctly classified as a subclass of the Cartesian type xsd:AnyType × xsd:AnyType.

5. Property Tab

As in OWL, we distinguish between the property characteristics owl:DatatypeProperty and owl:ObjectProperty. We group these two classes under the super-property MixedProperty, as we do allow for further "mixed" property characteristics; e.g., properties which are instantiated with an XSD atom in first place or properties with Cartesian domain and range types which are a mixture of OWL classes and XSD types (and thus are neither datatype nor object properties). Since the quaternary relation marriedTo (binary relation plus two extra args) maps URIs onto URIs, it is classified as an object property (remember, the extra args neither belong to the domain nor range of a property). However, the ternary relation hasAge (binary relation plus one extra args) is a datatype property as it maps URIs onto XSD ints (the extra arg is the *transaction time*, the time when the birthdate was entered to *HFC*); cf. Figure 3.

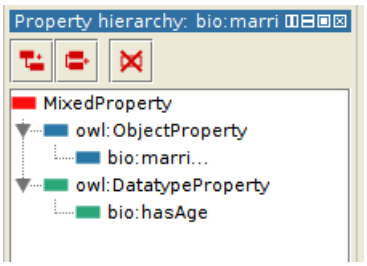

Figure 3: The property hierarchy for the marriedTo and hasAge relations.

Figure 4: The property signature for the marriedTo relation.

When defining a new property, a user is required to choose the right Cartesian types to complete the property signature. This is displayed in Figure 4 for the marriedTo relation. Depending on the kind of property, an ontology engineer is even allowed to associate further property characteristics with a property under definition; see Figure 5.

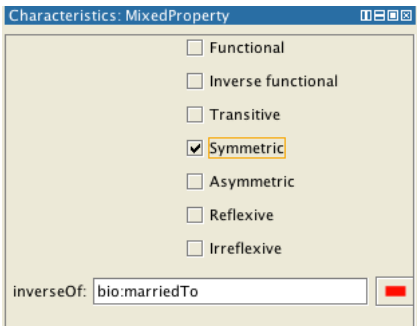

Figure 5: Further potential property characteristics for the marriedTo relation.

6. Instance Tab

We complete the overview of the workspace tabs by coming back to *Peter* and his relation to *Liz* and *Lisa* (cf. Section 1.). From the instance tab, we learn about his two marriages and that he is currently 53 years old (see Figure 6).
The symmetry of the marriedTo relation (see Figure 5) further guarantees that *Peter* is listed in the instance tabs of *Liz* and *Liza* as well.

7. N-Tuples & I/O Formats

As ×-*Protégé* allows us to deviate from pure binary relations, certain adjustments to the *N-triples* format (Carothers and Seaborne, 2014) are necessary, especially as extra arguments need to be represented. Assume a quaternary relation obtains between a person and a degree obtained from an educational organization at a specific time:

$$\text{obtains} \subseteq \underbrace{\text{Person}}_{\mathbb{D}} \times \underbrace{\text{Degree} \times \text{School}}_{\mathbb{R}_1 \times \mathbb{R}_2} \times \underbrace{\text{xsd:date}}_{\mathbb{A}}$$

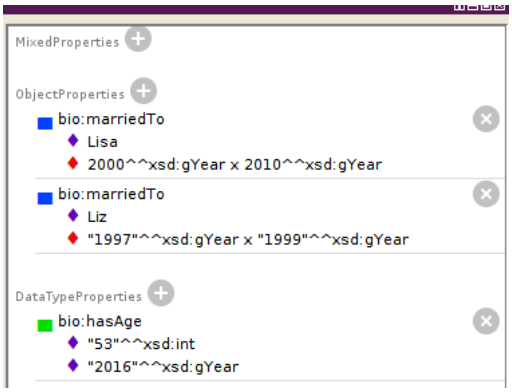

Figure 6: Facts about *Peter*.

In order to let the system know of how many arguments the domain, the range, and the extra part of a relation is composed of, we add further length-related information (infix notation):

```
obtains rdfs:domain Person
obtains rdfs:range Degree School
obtains nary:extra xsd:date
obtains nary:domainArity "1"^^xsd:int
obtains nary:rangeArity "2"^^xsd:int
obtains nary:extraArity "1"^^xsd:int
```

Notice that the rdfs:range keyword directly above is followed by *two* classes: Degree and School ($= \mathbb{R}_1 \times \mathbb{R}_2$). Not only is this kind of representation used in the RBox of an ontology, but also in the TBox, e.g.

```
Degree School rdfs:subClassOf owl:Thing owl:Thing
```

as

$$\text{Degree} \times \text{School} \sqsubseteq \top \times \top$$

is the case. ABox information is also affected by this style of representation, as, for instance

```
peter obtains phd stanford "1985"^^xsd:date
```

Besides providing such an (asymmetric) *infix* representation, ×-*Protégé* let the user decide whether a *prefix* representation is more appropriate for him/her. So, for instance, the last ABox statement above would then become

```
obtains peter phd stanford "1985"^^xsd:date
```

We finally like to stress the fact that once one decided to go for a direct representation of additional arguments and reason upon them, queries and rules will usually intermix tuples of different length. For example, in a *valid time* approach *universal* information from the TBox and RBox of an ontology is encoded as triples, whereas *assertional* knowledge will be represented as quintuples (Krieger, 2012); see *HFC* rule at the end of Section 8.

8. Future Work

Since ×-*Protégé* already uses functionality from *HFC* (see Section 3.), we would like to add further *query* and *rule definition* tabs to the next major version of ×-*Protégé* to support the construction of *HFC* queries and rules (see the two examples below).

The query support in ×-*Protégé* will ease the definition of SPARQL-like queries in *HFC* over n-tuples, using keywords such as SELECT, SELECTALL (for the *multiply-out* mode in *HFC* in case equivalence class reduction is enabled), DISTINCT, WHERE, FILTER, and AGGREGATE. Depending on the property signatures, ×-*Protégé* will then alarm a user if too less, too many, or wrong arguments have been specified in WHERE clauses, FILTER tests, or AGGREGATE functions. This helps to simply the construction of a query such as

```
SELECT DISTINCT ?partner
WHERE peter marriedTo ?partner ?start ?end
FILTER GreaterEqual ?start "1998"^^xsd:gYear &
       LessEqual ?end "2005"^^xsd:gYear
AGGREGATE ?noOfPartners = Count ?partner
```

which computes how many times *Peter* was married to distinct women between 1998 and 2005. The results of such queries (viz., tables) will also be displayed in this tab.

The rule support will provide means to define, maintain, and extend RDFS, OWL, and custom rule sets. Again, as is the case for queries, clauses, @test, and @action sections of rules in *HFC* will benefit from checking for the right number of arguments. For instance, the valid time extension of the entailment rule for *transitive* properties (ter Horst, 2005) in *HFC* looks as follows (Krieger, 2012):

```
?p rdf:type owl:TransitiveProperty     // triple
?x ?p ?y ?start1 ?end1                 // quintuple
?y ?p ?z ?start2 ?end2
→
?x ?p ?z ?start ?end
@test                                  // 3 LHS tests
?x != ?y
?y != ?z
IntersectionNotEmpty ?start1 ?end1 ?start2 ?end2
@action                                // 2 RHS actions
?start = Max2 ?start1 ?start2          // new RHS variable
?end = Min2 ?end1 ?end2                // new RHS variable
```

In both cases, we would also like to provide a *completion* mechanism for properties and URIs, as well as for external tests (see @test above) and value-returning functions (see @action above), an extremely useful functionality known from programming environments.

Our ultimate goal is thus to offer ×-*Protégé* as a front-end GUI for ontology-based systems, based on *HFC*.

9. Download

×-*Protégé* version 1.0 as of Monday Feb 15, 2016 can be downloaded from https://bitbucket.org/cwillms/x-protege/downloads/ and is made available under the Mozilla Public License. Here, you will also find a preliminary version of the user guide.

10. Acknowledgements

The research described in this paper has been partially financed by the European project PAL (Personal Assistant for healthy Lifestyle) under Grant agreement no. 643783-RIA Horizon 2020. This work was conducted using the *Protégé* resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health. We would like to thank the three reviewers for their detailed and useful suggestions.

11. Bibliographical References

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P. (2003). *The Description Logic Handbook*. Cambridge University Press, Cambridge.

Brickley, D. and Guha, R. (2014). RDF Schema 1.1. Technical report, W3C.

Carothers, G. and Seaborne, A. (2014). RDF 1.1 N-Triples. a line-based syntax for an RDF graph. Technical report, W3C.

Cyganiak, R., Wood, D., and Lanthaler, M. (2014). RDF 1.1 concepts and abstract syntax. Technical report, W3C.

Gangemi, A. and Presutti, V. (2013). A multi-dimensional comparison of ontology design patterns for representing n-ary relations. In *39th International Conference on Current Trends in Theory and Practice of Computer Science*, pages 86–105.

Hayes, P. and Welty, C. (2006). Defining N-ary relations on the Semantic Web. Technical report, W3C.

Hayes, P. (2004). RDF semantics. Technical report, W3C.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S. (2012). OWL 2 web ontology language primer (second edition). Technical report, W3C.

Keshavdas, S. and Kruijff, G.-J. (2014). Functional mapping for human-robot cooperative exploration. *International Journal of Computer and Applications*, 36(1).

Krieger, H.-U. and Declerck, T. (2015). An OWL ontology for biographical knowledge. representing time-dependent factual knowledge. In *Proceedings of the Workshop on Biographical Data in a Digital World*.

Krieger, H.-U. and Willms, C. (2015). Extending OWL ontologies by Cartesian types to represent N-ary relations in natural language. In *Proceedings of the IWCS Workshop on Language and Ontologies*.

Krieger, H.-U. (2012). A temporal extension of the Hayes/ter Horst entailment rules and an alternative to W3C's n-ary relations. In *Proceedings of the 7th International Conference on Formal Ontology in Information Systems (FOIS)*, pages 323–336.

Krieger, H.-U. (2013). An efficient implementation of equivalence relations in OWL via rule and query rewriting. In *Proceedings of the 7th IEEE International Conference on Semantic Computing (ICSC)*, pages 260–263.

Krieger, H.-U. (2014). A detailed comparison of seven approaches for the annotation of time-dependent factual knowledge in RDF and OWL. In *Proceedings of the 10th Joint ACL-ISO Workshop on Interoperable Semantic Annotation (ISA)*.

Krieger, H.-U. (2016). Capturing graded knowledge and uncertainty in a modalized fragment of OWL. In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART)*, pages 19–30.

Schmolze, J. G. (1989). Terminological knowledge representation systems supporting n-ary terms. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 432–443.

Schulz, S., Martínez-Costa, C., Karlsson, D., Cornet, R., Brochhausen, M., and Rector, A. (2014). An ontological analysis of reference in health record statements. In *Proceedings of the 8th International Conference on Formal Ontology in Information Systems (FOIS 2014)*.

Sider, T. (2001). *Four Dimensionalism. An Ontology of Persistence and Time*. Oxford University Press.

Smith, M. K., Welty, C., and McGuinness, D. L. (2004). OWL Web Ontology Language Guide. Technical report, W3C.

ter Horst, H. J. (2005). Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3:79–115.

Welty, C. and Fikes, R. (2006). A reusable ontology for fluents in OWL. In *Proceedings of 4th FOIS*, pages 226–236.