Extending OWL Ontologies by Cartesian Types to Represent N-ary Relations in Natural Language

Hans-Ulrich Krieger & Christian Willms
German Research Center for Artificial Intelligence (DFKI)
krieger@dfki.de | c.willms1@gmx.de

Abstract

Arbitrary n-ary relations ($n \geq 1$) can in principle be realized through binary relations obtained by a reification process that 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. Additional arguments for representing, e.g., valid time, grading, uncertainty, negation, trust, sentiment, or additional verb roles (for ditransitive verbs and adjuncts) are often better modeled in relation and information extraction systems as direct arguments of the relation instance, instead of being hidden in deep structures. In order to address non-binary relations directly, ontologies must be extended by *Cartesian* types, ultimately leading to an extension of the standard entailment rules for RDFS and OWL. In order to support ontology construction, ontology editors such as Protégé have to be adapted as well.

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 for relations of more than two arguments, some of them still retaining decidability and coming up with a better memory footprint and a better complexity for the various inference tasks than their triple-based relatives (Krieger, 2012). 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 (Krieger, 2014), 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 a *container* object to encode the range information (`ppt1` and `ppt2` being the new individuals):

```
peter marriedTo ppt1                    peter marriedTo ppt2
ppt1 rdf:type nary:PersonPlusTime       ppt2 rdf:type nary:PersonPlusTime
ppt1 nary:value liz                     ppt2 nary:value lisa
ppt1 nary:starts "1997"^^xsd:gYear      ppt2 nary:starts "2000"^^xsd:gYear
ppt1 nary:ends "1999"^^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 clearly time consuming, as it requires further classes, redefines property signatures, and rewrites relation instances, as shown by the *marriedTo* example. 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 relations.

2 What this Paper is (Not) About & Related Approaches

We would like to make clear that this paper is *not* about developing a theory for yet another new DL which permits n-ary relations. The approach presented here suggests that the concepts of *domain* and *range* of a relation are still useful when extending a binary relation with more arguments, instead of talking about the *arity* of a relation in general. We furthermore suggest in Section 6 to introduce so-called *extra arguments* which neither belong to the domain nor the range of a relation, and can be seen, as well, should be used as a kind of relation instance annotation. In the course of the paper, we also indicate that most of the entailment rules for RDFS (Hayes, 2004) and OWL Horst/OWL 2 RL (ter Horst, 2005; Motik et al., 2012) can be extended by Cartesian types and n-ary relations, and present an incomplete set of rules in Figure 1. Our approach takes a liberal stance in that it neither ask for the "nature" or "use" of the arguments (e.g., whether they are time points), nor for a (sound, complete, terminating, ...) set of tableau or entailment rules. In fact, if we would take this into account, we would end up in a potentially infinite number of different sets of rules, some of them requiring additional (lightweight) tests and actions, going beyond simple symbol matching; see (Krieger, 2012) for such a set of rules that model valid time, turning binary relations into quaternary ones. For various reasons, we propose a general restriction on the use of Cartesian types in Section 5, viz., to avoid typing individuals with Cartesian types and to maintain still singleton typing. The practical accomplishment of this paper lies in an extension of the Protégé editor for Cartesian types and n-ary relations that should be complemented by application-independent, but also domain-specific rules for a given application domain (e.g., to address *valid time*).

Since the early days of KL-ONE, DLs supporting relations with more than two arguments have been discussed, e.g., NARY[KANDOR] (Schmolze, 1989), $\mathcal{CIFR}$ (De Giacomo and Lenzerini, 1994), $\mathcal{DLR}$ (Calvanese et al., 1997), or $\mathcal{GF}1^{-}$ (Lutz et al., 1999). Especially Schmolze (1989) argued that "the advantages for allowing *direct* representation of n-ary relations far outweigh the reasons for the restriction" (i.e., restricting $n \leq 2$). To the best of our knowledge and with the exception of NARY[KANDOR], these DL languages have still remained theoretical work. In (Krieger, 2013), we presented an implemented theory-agnostic forward chainer, called *HFC*, which is comparable to popular semantic repositories such as Jena or OWLIM and which supports arbitrary n-tuples. The engine is able to run non-builtin entailment rule sets à la OWL Horst/OWL 2 RL and comes with a conservative extension of these OWL

dialects for *valid time* (Krieger, 2012). Further rule regimes are possible as long as they are expressible in *HFC*'s rule language which permits standard symbol matching, additional LHS tests, and RHS actions.

3 Extending Ontologies through Cartesian Types

Modern ontologies make use of standards, defined and coordinated by the W3C, such as XSD, RDF, RDFS, or OWL. OWL, as an instance of the description logic family, describes a domain in terms of classes (concepts), binary properties (roles), and instances (individuals). Complex expressions, so-called *axioms*, are defined via concept-forming operators (viz., subsumption and equivalence). The entirety of all such axioms which can be separated into those dealing with terminological knowledge (TBox), relational knowledge (RBox), and assertional knowledge (ABox), is usually called an *ontology* today.

Ontology editors which are geared towards RDF and OWL are thus *not* able to define n-ary relations *directly* in the RBox, nor are they capable of stating arbitrary tuples (instances of n-ary relations) in the ABox (together with Cartesian types in the TBox; see below). This would require an extension of the triple data model, or equivalently, allowing for n-ary relations ($n > 2$).

Formally, the extension of a binary relation p, can be seen as a (potentially infinite) set of pairs (s, o), coming from the Cartesian product of its *domain* $\mathbb{D}$ and *range* $\mathbb{R}$: $p \subseteq \mathbb{D} \times \mathbb{R}$. We then often say that a relation p is *defined* on $\mathbb{D}$, say, the *marriedTo* relation is *defined* on *Person*.

Now, in order to allow for more than two arguments, we decompose $\mathbb{R}$, leading to $p \subseteq \mathbb{D} \times \mathbb{R}_1 \times \cdots \times \mathbb{R}_n$. Note that we still make a distinction between domain $\mathbb{D}$ and range $\mathbb{R} = \mathbb{R}_1 \times \cdots \times \mathbb{R}_n$, and still say that p is *defined* on $\mathbb{D}$. Coming back to the previous section and the quaternary *marriedTo* relation, we can say that

$$\textit{marriedTo} \subseteq \textit{Person} \times \textit{Person} \times \textit{Year} \times \textit{Year}$$

For reasons that will become clear in a moment, not only the range but also the domain of a relation can, in principle, be deconstructed: $p \subseteq (\mathbb{D}_1 \times \cdots \times \mathbb{D}_m) \times (\mathbb{R}_1 \times \cdots \times \mathbb{R}_n)$. When it is clear from the context, we often omit the parentheses and simply write $p \subseteq \mathbb{D}_1 \times \cdots \times \mathbb{D}_m \times \mathbb{R}_1 \times \cdots \times \mathbb{R}_n$. We then say that the domain of p is $\mathbb{D}_1 \times \cdots \mathbb{D}_m$ and the range is $\mathbb{R}_1 \times \cdots \times \mathbb{R}_n$, thus p becomes an $(m + n)$-ary relation. Again, we say that p is defined on $\mathbb{D}_1 \times \cdots \mathbb{D}_m$.

Graphically, such an extension is easy write down. Let us start, again, with binary relations and let us picture the resulting graph for the following set $\{p(a, b), q(b, c), q(b, d), r(b, e)\}$ of binary relation instances by using directed labeled edges:

$$\begin{array}{ccccc} & & & & c \\ & & & \overset{q}{\nearrow} & \\ a & \xrightarrow{p} & b & \xrightarrow{q} & d \\ & & & \underset{r}{\searrow} & \\ & & & & e \end{array}$$

Ontology editors such as Protégé (Horridge, 2004) essentially use such a representation: properties are defined on certain classes and ontology or ABox) population reduces to filling missing range arguments for specific instances.

But how do we depict the following set of relation instances

$$\{r((a, b, c), (d)), p((a, b, c), (a, x)), q((a, x), (y, z)\}$$

of arity 4 and 5, resp? Quite easy, simply by replacing individuals (= singles) in domain and range position through general tuples:

$$(d) \xleftarrow{r} (a, b, c) \xrightarrow{p} (a, x) \xrightarrow{q} (y, z)$$

The "problem" with this kind of graph representation is that we are still using a kind of container (denoted by the parentheses) which groups both domain elements $\mathbb{D}_i$ ($1 \leq i \leq m$) and range elements $\mathbb{R}_j$ ($1 \leq j \leq n$). But this is something we want to avoid as explicated before (recall the *N-ary relation encoding* example from Section 1).

The answer to all this is already laying before us and has already been introduced, viz., *Cartesian* types (remember the $\times_i \mathbb{D}_i$ and $\times_j \mathbb{R}_j$ notation). This, however, will require to extend the descriptive expressiveness of the TBox, RBox, and ABox of an ontology.

4 Cartesian Types in TBox, RBox, and ABox

Protégé (and other ontology editors such as TopBraid) displays the *class subsumption hierarchy* using *indentation*, e.g.,

```
▽ Entity
  ▽ Object
    ▽ Agent
      ▷ Group
      ▽ Person
        ▷ Man
        ▷ Woman
```

These concepts can be seen as *singles* (or singletons), representing a Cartesian product of only one element. Thus the class `Person` can be seen as the tuple `(Person)`, consisting of one tuple element. Similarly, when considering the *marriedTo* relation, we might view the range type as the Cartesian type `(Person,Year,Year)`. Clearly, neither does `(Person)` subsume `(Person,Year,Year)`, nor does the opposite case hold—they are incompatible, for which we write

$$(\texttt{Person}) \bowtie (\texttt{Person}, \texttt{Year}, \texttt{Year})$$

However, the following subsumption relations do hold, given the above type hierarchy:

$$(\texttt{Man}, \texttt{Year}, \texttt{Year}) \sqsubseteq (\texttt{Person}, \texttt{Year}, \texttt{Year})$$
$$(\texttt{Woman}, \texttt{Year}, \texttt{Year}) \sqsubseteq (\texttt{Person}, \texttt{Year}, \texttt{Year})$$
$$(\texttt{Person}, \texttt{Year}, \texttt{Year}) \sqsubseteq (\texttt{Agent}, \texttt{Year}, \texttt{Year})$$
$$(\texttt{Group}, \texttt{Year}, \texttt{Year}) \sqsubseteq (\texttt{Agent}, \texttt{Year}, \texttt{Year})$$

Now let $\mathcal{C}$ denote the set of concepts, $\mathcal{R}$ denote the set of all relations, and $\mathcal{I}$ denote the set of all instances. Quite naturally, the subsumption relation for concepts $\sqsubseteq \subseteq \mathcal{C} \times \mathcal{C}$ can be easily extended to Cartesian types:

$$\times_{i=1}^{m} C_i \sqsubseteq \times_{j=1}^{n} D_i \textbf{ iff } m = n \textbf{ and } C_i \sqsubseteq D_i, \textbf{ for all } i \in \{1, \ldots, m\}$$

Given such an extension, many of the standard entailment rules from (Hayes, 2004) and (ter Horst, 2005) can be easily adjusted, but also two new rules, called (ctsub) and (ctequiv), need to be introduced which propagate Cartesian type subsumption and equivalence down to their component classes (see Figure 1 for a representative, non-complete set of extended rules).

5 A Restriction on the Use of Cartesian Types

The extension introduced so far would even allow us to type *individuals* $a \in \mathcal{I}$ with any Cartesian type $\times_{i=1}^{m} C_i$ ($m \geq 1$) for which we might then write $\times_{i=1}^{m} C_i(a)$. This would make it possible to naturally extend, e.g., the universal instantiation schema (rdfs9) from Hayes (2004) with Cartesian types, viz.,

$$\text{(rdfs9)} \quad \times_{i=1}^{m} C_i(a) \wedge \times_{i=1}^{m} C_i \sqsubseteq \times_{i=1}^{m} D_i \rightarrow \times_{i=1}^{m} D_i(a)$$

Such an extension is attractive, but has severe drawbacks. It makes domain and range inference more complex and would require a stronger descriptive apparatus, as it will become necessary to group and access *parts* of the domain and/or range arguments in order to indicate the true number of arguments of a relation, but also to indicate the proper argument types. This would become important when checking relation instances against their relation signature.

Consider, for instance, a quaternary relation $p \subseteq \mathbb{D} \times \mathbb{R}_1 \times \mathbb{R}_2 \times \mathbb{R}_3$ that seems to come with three *range* arguments. However, by typing individuals with Cartesian types, the above relation can be binary, ternary (two possibilities), or quaternary, depending on how we interpret the range arguments:

- $p \subseteq \mathbb{D} \times (\mathbb{R}_1 \times \mathbb{R}_2 \times \mathbb{R}_3)$
- $p \subseteq \mathbb{D} \times \mathbb{R}_1 \times \mathbb{R}_2 \times \mathbb{R}_3$
- $p \subseteq \mathbb{D} \times (\mathbb{R}_1 \times \mathbb{R}_2) \times \mathbb{R}_3$
- $p \subseteq \mathbb{D} \times \mathbb{R}_1 \times (\mathbb{R}_2 \times \mathbb{R}_3)$

And there are even further complex embeddings possible (remember type theory), such as

- $p \subseteq \mathbb{D} \times (\mathbb{R}_1 \times (\mathbb{R}_2 \times \mathbb{R}_3))$
- $p \subseteq \mathbb{D} \times ((\mathbb{R}_1 \times \mathbb{R}_2) \times \mathbb{R}_3)$

(ctsub) $\times_{i=1}^{m} C_i \sqsubseteq \times_{i=1}^{m} D_i \rightarrow \bigwedge_{i=1}^{m} C_i \sqsubseteq D_i$

(rdfs11) $\times_{i=1}^{m} C_i \sqsubseteq \times_{i=1}^{m} D_i \wedge \times_{i=1}^{m} D_i \sqsubseteq \times_{i=1}^{m} E_i \rightarrow \times_{i=1}^{m} C_i \sqsubseteq \times_{i=1}^{m} E_i$

(ctequiv) $\times_{i=1}^{m} C_i \equiv \times_{i=1}^{m} D_i \rightarrow \bigwedge_{i=1}^{m} C_i \equiv D_i$

(rdfp12c) $\times_{i=1}^{m} C_i \sqsubseteq \times_{i=1}^{m} D_i \wedge \times_{i=1}^{m} D_i \sqsubseteq \times_{i=1}^{m} C_i \rightarrow \times_{i=1}^{m} C_i \equiv \times_{i=1}^{m} D_i$

(rdfs2) $\forall P^{-}.\, \times_{i=1}^{m} C_i \wedge P(\times_{i=1}^{m} a_i, \times_{j=1}^{n} b_j) \rightarrow \bigwedge_{i=1}^{m} C_i(a_i)$

(rdfs3) $\forall P.\, \times_{j=1}^{n} D_j \wedge P(\times_{i=1}^{m} a_i, \times_{j=1}^{n} b_j) \rightarrow \bigwedge_{j=1}^{n} D_j(b_j)$

(rdfs7x) $P \sqsubseteq Q \wedge P(\times_{i=1}^{m} a_i, \times_{j=1}^{n} b_j) \rightarrow Q(\times_{i=1}^{m} a_i, \times_{j=1}^{n} b_j)$

(rdfp1) $\leq 1 P \wedge P(\times_{i=1}^{m} a_i, \times_{j=1}^{n} b_j) \wedge P(\times_{i=1}^{m} a_i, \times_{j=1}^{n} c_j) \rightarrow \bigwedge_{j=1}^{n} \{b_j\} \equiv \{c_j\}$

(rdfp3) $P \equiv P^{-} \wedge P(\times_{i=1}^{m} a_i, \times_{i=1}^{m} b_i) \rightarrow P(\times_{i=1}^{m} b_i, \times_{i=1}^{m} a_i)$

(rdfp4) $P^{+} \sqsubseteq P \wedge P(\times_{i=1}^{m} a_i, \times_{i=1}^{m} b_i) \wedge P(\times_{i=1}^{m} b_i, \times_{i=1}^{m} c_i) \wedge \rightarrow P(\times_{i=1}^{m} a_i, \times_{i=1}^{m} c_i)$

Figure 1: Entailment rules using Cartesian types ($C_i, D_j, E_k \in \mathcal{C}$; $P, Q \in \mathcal{R}$; $a., b., c. \in \mathcal{I}$). Note that the notation $P(\times_{i=1}^{m} a_i, \times_{j=1}^{n} b_j)$ in the above rules does *not* indicate that P is a binary relation, but instead is of arity $m + n$ *and* $a_1, \ldots, a_m$ are the domain and $b_1, \ldots, b_n$ the range arguments for this specific relation instance of P. The names for the extended rule schemata are taken from (Hayes, 2004) and (ter Horst, 2005). (ctsub) and (ctequiv) are brand-new entailment rules for Cartesian types. The correctness of (rdfp4), addressing the transitivity of P, depends on the interpretation of the application domain (for instance, whether certain arguments are employed for expressing the validity of a fluent (the atemporal fact) over time; see also Section 6).

Mainly for this reason, we enforce that atomic *individuals* from $\mathcal{I}$ can only be typed to *single* concepts (singletons), and thus the relation signature

$$p \subseteq \mathbb{D}_1 \times \cdots \times \mathbb{D}_m \times \mathbb{R}_1 \times \cdots \times \mathbb{R}_n$$

is intended to mean that p takes *exactly* m domain arguments and *exactly* n range arguments, such that $\mathbb{D}_1, \ldots, \mathbb{D}_m, \mathbb{R}_1, \ldots, \mathbb{R}_n \in \mathcal{C}$ must be the case.

6 Extra Arguments

This section deals with what we call *extra arguments*, arguments that neither belong to the domain nor the range of an $(m + n)$-ary relation, but can be seen as a kind of additional *annotation*, belonging to specific relation instances.[1]

Let us start with a binary relation ($m, n = 1$) and consider, again, the non-temporal version of *marriedTo* which is a true *symmetric* relation, expressed by the following instantiated entailment rule:

$$\textit{marriedTo}(i, j) \rightarrow \textit{marriedTo}(j, i)$$

Now, if we add time (b = begin; e = end), it becomes a quaternary relation as indicated before (for better readability, we separate the domain and range arguments from one another by using parentheses):

$$\checkmark\ \textit{marriedTo}(i, (j, b, e)) \rightarrow \textit{marriedTo}(j, (i, b, e))$$

In this sense, the temporal interval $[b, e]$ specifies the valid time in which the fluent (the atemporal statement) $\textit{marriedTo}(i, j)$ is true. By applying the extended rule (rdfp3) from Figure 1 for symmetry, we see that something clearly goes wrong:

$$\lightning\ \textit{marriedTo}(i, (j, b, e)) \rightarrow \textit{marriedTo}((j, b, e), i)$$

[1] This is like having annotation properties for *relation instances*, but OWL unfortunately offers this service only for classes, properties, and individuals.

as symmetric relations assume the same number of arguments in domain and range position! Our example above thus needs to be modified. One solution would be to reduplicate the starting and ending points, so we would end up in a sexternary relation:

$$\textit{marriedTo}((i,\underline{b,e}),(j,\underline{b,e})) \rightarrow \textit{marriedTo}((j,\underline{b,e}),(i,\underline{b,e}))$$

This is *not* an appealing solution as the structures become larger, and rules and queries are harder to formulate, read, debug, and process. We thus like to extend relations $p \subseteq \mathbb{D}_1 \times \cdots \times \mathbb{D}_m \times \mathbb{R}_1 \times \cdots \times \mathbb{R}_n$ by further arguments $\mathbb{A}_1 \times \cdots \times \mathbb{A}_o$, so that p becomes

$$p \subseteq \mathbb{D}_1 \times \cdots \times \mathbb{D}_m \times \mathbb{R}_1 \times \cdots \times \mathbb{R}_n \times \mathbb{A}_1 \times \cdots \times \mathbb{A}_o$$

or simply write $p \subseteq \mathbb{D} \times \mathbb{R} \times \mathbb{A}$. For the *marriedTo* example, we might choose Person from the ontology above and the XSD type gYear: $\mathbb{D}$ = Person, $\mathbb{R}$ = Person, $\mathbb{A}$ = gYear $\times$ gYear.

Thus by having these *extra* arguments, we can keep the entailment rules from Figure 1, extended, of course, by the additional annotations.[2] Besides having extra arguments for *valid time*, other areas are conceivable here, viz., *transaction time, space, sentiment, uncertainty, negation, vagueness, or graded information*.

7 Extensions to Protégé

In order to make Cartesian types available in Protégé, we will extend the *OWL Classes*, *Properties*, and *Individuals* tabs.

TBox: *OWL Classes* Tab

- *subclass explorer* pane (left column)
  extension of the subclass hierarchy towards Cartesian types.
- *class editor* pane (right column)
  depicting the right properties defined on a Cartesian type (domain); depicting the right Cartesian range types for the defined properties.

RBox: *Properties* Tab

- *property browser* pane (left column)
  extension of the property hierarchy towards Cartesian types.
- *property editor* pane (right column)
  extension of the domain and range boxes towards Cartesian types.
- **new:** *extra arguments* (part of the *property editor* pane)
  further definition box for the extra arguments.

ABox: *Individuals* Tab

- *class browser* pane (left column)
  extension of the subclass hierarchy towards Cartesian types.
- *instance browser* pane (middle column)
  possibility to generate sequence instances defined on Cartesian types (= sequences of instances of singleton types).
- *property editor* pane (right column)
  depicting the right properties defined on a sequence instance; allowing to choose or construct the range arguments; allowing to choose or construct the extra arguments.

Not only the graphical user interface needs to be extended, but also the internal representation (representation of tuples instead of triples), together with a modification of the input and output routines. We plan to have finished a first version of the extensions to Protégé in Spring 2015 and to present it at the workshop.

[2] Depending on the application domain, these annotations might find their way as (potentially aggregated) extra arguments in the relation instances of the consequence of a rule, e.g., in (rdfp4). We will look into this in more detail at the workshop.

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. The authors have profited from discussions with our colleague Bernd Kiefer and would like to thank the three reviewers for their suggestions.

References

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

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

Calvanese, D., G. De Giacomo, and M. Lenzerini (1997). Conjunctive query containment in description logics with n-ary relations. In *Proceedings of the International Workshop on Description Logics*, pp. 5–9.

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

De Giacomo, G. and M. Lenzerini (1994). Description logics with inverse roles, functional restrictions, n-ary relations. In *Proceedings of the 4th European Workshop on Logics in Artificial Intelligence*, pp. 332–346.

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

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

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

Horridge, M. (2004). A practical guide to building OWL ontologies with the Protégé-OWL plugin. Technical report, University of Manchester.

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)*, pp. 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)*, pp. 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*.

Lutz, C., U. Sattler, and S. Tobies (1999). A suggestion for an n-ary description logic. In *Proceedings of the International Workshop on Description Logics*, pp. 81–85.

Motik, B., B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz (2012). OWL 2 web ontology language profiles. Technical report, W3C.

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*, pp. 432–443.

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

Smith, M. K., C. Welty, and D. L. McGuinness (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.