Ontology Design Pattern Language Expressivity Requirements

Matthew Horridge[1], Mikel Egaña Aranguren[2], Jonathan Mortensen[1], Mark Musen[1], and Natalya F. Noy[1]

[1] Stanford Center for BioInformatics Research (BMIR), Stanford University, California, USA
[2] Biological Informatics, Centre for Plant Biotechnology and Genomics (CBGP), Technical University of Madrid, (UPM), Spain

Abstract. In recent years there has been a large amount of research into capturing, publishing and analysing Ontology Design Patterns (ODPs). However, there has not been any analysis into the typical language expressivity required to represent ODPs and how these requirements sit with lightweight fragments of the widely used ontology language OWL. In this paper we therefore present a survey on the language expressivity required to express the ODPs contained in the two main ODP catalogs: ODP.org and ODPS.sf.net. We surveyed a total of 104 machine processable ODPs and found that the OWL representations of these patterns typically require highly expressive fragments of the OWL language such as $\mathcal{ALCHIN}$, $\mathcal{SHOIN}$, $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. We observed that most ODPs required the use of inverse properties, cardinality restrictions and universal restrictions, and that 10 patterns require OWL 2 constructs such as property chains, disjoint properties and qualified cardinality restrictions that are not available in OWL 1. Moreover, we found that most of the ODPs cannot be incorporated into ontologies that are constrained to fit into one of the OWL 2 profiles. Specifically, only 12 out of the 104 ODPs surveyed can be represented in OWL2EL, 13 in OWL2RL and 23 in OWL2QL. Despite this, we conjecture that it may be possible to rewrite and weaken some of them so that modellers using lightweight fragments of OWL can incorporate ODPs into their ontologies.

1 Introduction

Ontology Design Patterns (ODPs) are modelling solutions that solve recurrent ontology design problems [4]. Much of the work on ODPs has been inspired by the well known work on software design patterns from the mid nineties [3]; Gamma *et al.* presented and categorised small object-oriented models which are intended to be general solutions to specific but common problems in software design. The main benefit behind the use of software design patterns is that they decrease the incidence of poor modelling choices that could cause problems at a later date. Additionally, code that is based on software design patterns is more readable, more maintainable, and more reusable than code which is not. In many ways these are the exact reasons that motivate the use of design patterns as applied to ontologies and in recent years there has been a thrust of research aiming at inventing and promoting the use of ODPs in ontology engineering.

Two prominent sources of ODPs are the `ontologydesignpatterns.org` catalog (ODP.org), and the `odps.sourceforge.net` catalog on the SourceForge.net website (ODPS.sf.net). The first is a centralised repository, initiated by Gangemi *et al.*, which takes submissions from researchers and practitioners that work in a variety of research areas and application domains. At the time of writing, this repository contains over 150 submissions binned into different categories. The second is based on the experience of Egaña *et al.* at the University of Manchester in representing biomedical knowledge. They propose 17 ODPs designed to tackle modelling problems and ontology language limitations. Both of these catalogs are specific to the Web Ontology Language OWL [10]. That is, ODPs are presented and described in terms of fragments of OWL ontologies.

The focus on OWL, as opposed to a "generic" (logic) based ontology language, is unsurprising. OWL is one of the most widely used ontology languages, it has excellent tool support in terms of editors and reasoners, and it is a World Wide Web Consortium (W3C) recommendation. The latest version of OWL is OWL 2, which became a W3C recommendation in October 2009 and is based on Description Logics. This logical underpinning provides a precise semantics and makes it possible to specify various reasoning tasks such as consistency checking, satisfiability testing and general entailment checking. Off the shelf automated reasoners such as ELK, FaCT++, HermiT, Pellet and Racer can be used to perform these key reasoning tasks and ontology development environments such as Protégé and the NeOn Toolkit provide hooks for integrating third party reasoners. Since OWL 2 is a highly expressive language key reasoning tasks like consistency checking have an extremely high worst case complexity: 2NExpTime-Complete [8], *i.e.*, intractable. This, coupled with the fact that implementing a highly optimised and scalable reasoner for the full language is a non-trivial task, and the fact that several well known biomedical ontologies, such as SNOMED [13], fit within smaller tractable fragments of the language, led to the development of the three so-called "profiles": OWL2EL, OWL2RL and OWL2-QL [9]. Each profile was carefully designed with practical use cases in mind, but a key aspect of each one is that it restricts what can be modelled so as to limit expressivity and make it possible, and easy, to implement efficient and scalable reasoners.

When it comes to applying ODPs to a domain or application ontology, and in particular an ontology which is specifically designed to fit into one of the OWL 2 profiles, it is not clear whether it is possible to incorporate any of the cataloged ODPs without compromising language expressivity constraints. This is a real concern, particularly for biomedical ontologies. For example, the large and well known medical ontology SNOMED is constrained to fit within the lightweight OWL2EL profile [12]. Many of the ontologies in the NCBO BioPortal repository [2] fall into the OWL2EL profile [6], and a sizeable number of them fall into the OWL2RL profile. Whether these ontologies were deliberately constrained to these profiles or not, it seems plausible that application of ODPs to these ontologies could easily take them outside of these profiles, thus losing the possibility of efficient, scalable reasoning. Ultimately, little is known about the language expressivity required to represent cataloged ODPs and how many ODPs can be represented or used within one of the OWL 2 profiles. The aim of this paper is therefore to present a survey and discussion of ODP language expressivity requirements.

2 Preliminaries

OWL 2 An OWL 2 ontology is a set of axioms (statements) which state something about the domain of interest. For example, a *subclass* axiom states that one class is a subclass of another class (*e.g.* Car is a subclass of Vehicle); an *inverse properties* axiom states that one property is the inverse of another property (*e.g.* hasPart is the inverse of isPartOf); and a *disjoint classes* axiom states that one class is disjoint with another class (*e.g.* Plant is disjoint with Animal). The OWL 2 language is underpinned by a highly expressive Description Logic called $\mathcal{SROIQ}$ [7]. This gives statements made in OWL a precisely defined meaning and, for a given ontology, makes it possible to use automated reasoning to compute whether or not a statement follows from the ontology. Statements that follow from an ontology are known as *entailments*. The process of reasoning used to determine whether or not an entailment follows from an ontology is known as *entailment checking*. Generally speaking, expressivity comes at a price—as expressivity increases so does the complexity (difficulty) of various reasoning problems such as entailment checking.

Description Logics As mentioned, OWL is underpinned by a Description Logic called $\mathcal{SROIQ}$. Generally speaking, Description Logics (DLs) are decidable fragments of First Order Logic. There are many different DLs, with each one being defined by the class, property and axiom constructors that it admits. One of the simplest DLs is known as $\mathcal{AL}$ (Attributive Language). This DL supports concept intersection (`owl:intersectionOf`), universal quantification (`owl:allValuesFrom`), limited existential quantification (`owl:someValuesFrom` with a filler restricted to `owl:Thing`) and atomic negation (`owl:complementOf` to named classes). More expressive DLs can be obtained from $\mathcal{AL}$ by adding further constructors. Each constructor is given a specific letter which is used to derive a name for any particular DL. For example, adding full negation $\mathcal{C}$ to $\mathcal{AL}$ produces the DL $\mathcal{ALC}$. Adding property hierarchy $\mathcal{H}$ (`rdfs:subPropertyOf`) to $\mathcal{ALC}$ produces $\mathcal{ALCH}$. Adding nominals $\mathcal{O}$ (`owl:oneOf`), inverse properties $\mathcal{I}$ (`owl:inverseOf`) and number restrictions $\mathcal{N}$ (`owl:minCardinality`, `owl:maxCardinality` or `owl:cardinality`) to $\mathcal{ALCH}$ produces $\mathcal{ALCHOIN}$. Finally adding transitive properties (`owl:TransitiveProperty`) to $\mathcal{ALCHOIN}$ produces $\mathcal{SHOIN}$, as the combination of $\mathcal{ALC}$ with transitive properties is abbreviated to $\mathcal{S}$. $\mathcal{SHOIN}$ is the DL that underpins OWL 1. OWL 2 extends the expressivity of OWL 1 with qualified cardinality $\mathcal{Q}$ to give $\mathcal{SHOIQ}$, and complex chains (`owl:propertyChainAxiom`), reflexive (`owl:ReflexiveProperty`), irreflexive (`owl:IrreflexiveProperty`), and disjoint properties (`owl:disjointWith`) $\mathcal{R}$ to give $\mathcal{SROIQ}$. Distinct from the $\mathcal{AL}$ based family of DLs are $\mathcal{EL}$ based DLs. Rather than being based on universal quantification (`owl:allValuesFrom`), the $\mathcal{EL}$ family is based on existential quantification (`owl:someValuesFrom`), and universal quantification is prohibited. An important $\mathcal{EL}$ based DL is $\mathcal{EL}^{++}$ [1]. This is the DL which underpins OWL2EL and for which entailment checking can be efficiently performed in polynomial time w.r.t. the size of the input ontology.

OWL 2 Profiles As mentioned in the introduction, OWL 2 contains three profiles:

- OWL2EL—Based on the lightweight $\mathcal{EL}^{++}$ DL. OWL2EL is designed for representing large and moderately complex ontologies. In particular, it was designed with biomedical ontologies in mind.
- OWL2RL —This profile was designed to allow reasoning to be efficiently implemented with traditional rule engine based technologies.
- OWL2QL —Based on the lightweight DL-Lite family of DLs. This profile was designed for applications that combine a simple ontology with large amounts of instance data (possibly stored in a database).

Each profile limits the class, property and axiom constructors that it admits in order to achieve desirable properties in terms of reasoning. Of particular interest to this work is the OWL2EL profile, which is designed for representing large BioMedical ontologies. This profile prohibits the use of universal restrictions ($\forall$, `owl:allValuesFrom`), cardinality restrictions (`owl:minCardinality`, `owl:maxCardinality`, `owl:cardinality`), functional (`owl:FunctionalProperty`), inverse functional properties (`owl:InverseFunctionalProperty`), inverse properties (`owl:inverseOf`), disjunction ($\sqcup$, `owl:unionOf`), arbitrary negation ($\neg$, `owl:complementOf`), enumerations involving more that one individual (`owl:oneOf`), and disjoint, irreflexive and asymmetric properties. A full specification of each profile is beyond the scope of this paper—the interested reader is referred to the OWL 2 Profiles specification document [9].

Ontology Design Pattern Documents As mentioned in Section 1, the two main repositories of ODPs are the ODP.org repository and the ODPS.sf.net catalog. Each repository describes ODPs in a fairly standard way, *i.e.*, the name of the pattern, the problem the pattern is supposed to solve, limitations *etc.* Most of the patterns in each repository also contain a small ontology document that represents the pattern: a "pattern ontology". This pattern ontology document either (1) provides a domain specific example of the pattern, or (2) represents a small ontology that can be reused in the domain ontology. In the first case, the idea is that the pattern ontology document is copied into the domain ontology document with the appropriate translation of vocabulary from the example domain to the real domain. In the second case, the pattern ontology document is directly reused, by import, without modification. Patterns that fall into the second category are known as *Content* Design Patterns [11].

3 Materials and Method

Pattern Selection The ODPs that we considered for this study are the ones which contain a pattern ontology document as part of their pattern definition. That is, ODPs which are described with a parsable OWL ontology. Being able to parse a pattern (or an exemplar of a pattern) from an ontology document provides a clean way to automatically and unambiguously analyse the pattern, determine the expressivity of the language that is required to represent that pattern, and determine if this language falls into one of the three OWL 2 profiles. In the case of the ODP.org catalog, there are 91 proposed content ontology design patterns[3] that are represented as ontology documents. In the case of the ODPS.sf.net catalog[4] all 17 cataloged patterns are described with exemplar ontology documents.

Pattern Retrieval We accessed both catalogs on the 8th of August 2012. A total of 91 content patterns were listed in the ODP.org catalog. However, the ontology documents for the patterns PharmaInnova and BiologicalEntities could not be downloaded due to HTTP 404 (File Not Found) errors. Thus, we obtained a total of 89 pattern ontology documents from the ODP.org catalog. In the ODPS.sf.net catalog we retrieved all 17 pattern ontology documents. Out of these, we discarded the Normalisation and UpperLevelOntology pattern documents: these patterns do not require a fixed or minimum level of expressivity for instantiating them—the Normalisation pattern depends upon the class definitions that are appropriate to the domain being modelled, while the UpperLevelOntology pattern depends on the particular upper level ontology in question. This provided us with a total of 15 ontology pattern documents from the ODPS.sf.net catalog, and therefore a grand total of 104 pattern ontology documents for processing.

Pattern Processing and Analysis Each ontology document was parsed with the OWL API [5] (Version 3.3) in order to check that it was well formed. Next the OWL API Metrics and Profiles APIs were used to first compute the expressivity required to represent the pattern and then to check to see whether the pattern falls into OWL2DL and the three OWL profiles: OWL2EL, OWL2RL and OWL2-QL. The results are summarised in the two tables in Section 4 below.

4 Results

Results are summarised in Table 1 and Table 2. The columns display the name of the pattern, the expressivity required to represent the pattern, whether the pattern fits into OWL2DL (DL), OWL2EL (EL), OWL2RL (RL) or OWL2QL (QL), whether the pattern contains universal restrictions ($\forall$), or disjunctions ($\sqcup$), and whether or not OWL 2 constructs are required to represent the pattern

[3] http://ontologydesignpatterns.org/wiki/Category:ProposedContentOP
[4] http://odps.sourceforge.net/

(Req.OWL2). Patterns that fulfil these latter seven properties are denoted with tick (✔) in the appropriate cell, otherwise the cell is left empty. Usage of universal restrictions ($\forall$) and disjunction ($\sqcup$) has been singled out for presentation because these two constructors would otherwise be "lumped in" with the base language $\mathcal{AL}$. In addition to this, their use is prohibited in two out of the three OWL 2 profiles, namely OWL2EL and OWL2QL.

Table 1: Expressivity required for the ODPs from the ODP.org catalog.

Id	Name	Expressivity	DL	EL	RL	QL	$\forall$	$\sqcup$	Req.OWL2
1	EthnicGroup	$\mathcal{EL}^{++}$	✔	✔		✔			
2	RTMSMapping	$\mathcal{EL}^{++}$	✔	✔		✔			
3	SpeciesConservation	$\mathcal{EL}^{++}$	✔	✔		✔			
4	Airline	$\mathcal{EL}^{++}$	✔	✔	✔	✔			
5	ConceptGroup	$\mathcal{EL}^{++}$	✔	✔	✔	✔			
6	ConceptTerms	$\mathcal{EL}^{++}$	✔	✔	✔	✔			
7	Metonymy	$\mathcal{EL}^{++}$	✔	✔	✔	✔			
8	SpeciesNames	$\mathcal{EL}^{++}$	✔	✔	✔	✔			
9	GoTop	$\mathcal{EL}^{++}$	✔	✔	✔				
10	Invoice	$\mathcal{ALF}(\mathcal{D})$	✔		✔				
11	Classification	$\mathcal{ALI}$	✔			✔			
12	Collection	$\mathcal{ALI}$	✔			✔			
13	CollectionEntity	$\mathcal{ALI}$	✔			✔			
14	Constituency	$\mathcal{ALI}$	✔			✔			
15	ActingFor	$\mathcal{ALI}$	✔		✔	✔			
16	PartOf	$\mathcal{ALI}+$	✔						
17	Place	$\mathcal{ALI}+$	✔						
18	Set	$\mathcal{ALI}(\mathcal{D})$	✔			✔			
19	Region	$\mathcal{ALI}(\mathcal{D})$	✔				✔		
20	Parameter	$\mathcal{ALI}(\mathcal{D})$	✔		✔	✔			
21	SpeciesEat	$\mathcal{ALEI}$	✔			✔			
22	AOS	$\mathcal{ALUHIF}+(\mathcal{D})$	✔					✔	
23	HasPest	$\mathcal{ALUHIF}+(\mathcal{D})$	✔					✔	
24	TimeInterval	$\mathcal{ALHN}(\mathcal{D})$	✔		✔				
25	ObjectRole	$\mathcal{ALHI}$	✔			✔			
26	Componency	$\mathcal{ALHI}+$	✔				✔		
27	Sequence	$\mathcal{ALHI}+$	✔						
28	IntensionExtension	$\mathcal{ALIN}$	✔				✔		
29	Situation	$\mathcal{ALIN}$	✔						
30	TimeIndexedSituation	$\mathcal{ALHIN}(\mathcal{D})$	✔						
31	LiteralReification	$\mathcal{ALHIN}+(\mathcal{D})$	✔				✔		
32	CommunicationEvent	$\mathcal{ALEHOIN}(\mathcal{D})$	✔				✔		
33	TypesOfEntities	$\mathcal{ALC}$	✔					✔	
34	AquaticResources	$\mathcal{ALCI}$	✔			✔			
35	Participation	$\mathcal{ALCI}$	✔			✔			
36	SpeciesHabitat	$\mathcal{ALCI}$	✔			✔			
37	InformationRealization	$\mathcal{ALCI}$	✔				✔		
38	GearVessel	$\mathcal{ALCI}$	✔		✔		✔		
39	RoleTask	$\mathcal{ALCI}$	✔		✔		✔		
40	TaskRole	$\mathcal{ALCI}$	✔		✔		✔		
41	AgentRole	$\mathcal{ALCHI}$	✔			✔			
42	GearSpecies	$\mathcal{ALCHI}$	✔			✔			
43	GearWaterArea	$\mathcal{ALCHI}$	✔			✔			
44	Communities	$\mathcal{ALCHI}$	✔				✔		
45	VesselSpecies	$\mathcal{ALCHI}$	✔				✔		
46	VesselWaterArea	$\mathcal{ALCHI}$	✔				✔		
47	Move	$\mathcal{ALCHI}$	✔		✔	✔			
48	CoParticipation	$\mathcal{ALCIN}$	✔						
49	CountingAs	$\mathcal{ALCIN}$	✔						
50	Price	$\mathcal{ALCIN}(\mathcal{D})$	✔						
51	SpeciesBathymetry	$\mathcal{ALCIN}(\mathcal{D})$	✔						
52	Criterion	$\mathcal{ALCHIN}$	✔						
53	CriterionSetter	$\mathcal{ALCHIN}$	✔						

Table 1: Expressivity required for the ODPs from the ODP.org catalog.

Id	Name	Expressivity	DL	EL	RL	QL	∀	⊔	Req.OWL2
54	Description	$\mathcal{ALCHIN}$	✔						
55	DescriptionAndSituation	$\mathcal{ALCHIN}$	✔						
56	ParticipationRole	$\mathcal{ALCHIN}$	✔						
57	Persons	$\mathcal{ALCHIN}$	✔					✔	
58	SpeciesConditions	$\mathcal{ALCHIN}$	✔						
59	TaskExecution	$\mathcal{ALCHIN}$	✔				✔		
60	Bag	$\mathcal{ALCHIN}(\mathcal{D})$	✔						
61	BasicPlanExecution	$\mathcal{ALCHIN}(\mathcal{D})$	✔				✔		
62	ClimaticZone	$\mathcal{ALCHIN}(\mathcal{D})$	✔						
63	NaryParticipation	$\mathcal{ALCHIN}(\mathcal{D})$	✔						
64	Observation	$\mathcal{ALCHIN}(\mathcal{D})$	✔						
65	ResourceAbundanceObs	$\mathcal{ALCHIN}(\mathcal{D})$	✔						
66	ResourceExploitationObs	$\mathcal{ALCHIN}(\mathcal{D})$	✔						
67	Tagging	$\mathcal{ALCHIN}(\mathcal{D})$	✔				✔		
68	TimeIndexedClassification	$\mathcal{ALCHIN}(\mathcal{D})$	✔						
69	TimeIndexedParticipation	$\mathcal{ALCHIN}(\mathcal{D})$	✔						
70	TimeIndexedPersonRole	$\mathcal{ALCHIN}(\mathcal{D})$	✔						
71	VerticalDistribution	$\mathcal{ALCHIN}(\mathcal{D})$	✔						
72	LinneanTaxonomy	$\mathcal{SHI}$	✔				✔	✔	
73	SimpleOrAggregated	$\mathcal{SHI}$	✔					✔	
74	ControlFlow	$\mathcal{SHIN}$	✔				✔		
75	InformationObjects	$\mathcal{SHIN}$	✔				✔	✔	
76	SimpleTopic	$\mathcal{SHIN}$	✔				✔		
77	Topic	$\mathcal{SHIN}$	✔				✔		
78	Action	$\mathcal{SHIN}(\mathcal{D})$	✔				✔		
79	BasicPlan	$\mathcal{SHIN}(\mathcal{D})$	✔				✔		
80	List	$\mathcal{SHIN}(\mathcal{D})$	✔						
81	PlanConditions	$\mathcal{SHIN}(\mathcal{D})$	✔				✔		
82	TimeIndexedPartOf	$\mathcal{SHIN}(\mathcal{D})$	✔						
83	PeriodicInterval	$\mathcal{SHOIN}(\mathcal{D})$	✔						
84	CatchRecord	$\mathcal{SHIQ}(\mathcal{D})$	✔				✔	✔	✔
85	Transition	$\mathcal{SHIQ}(\mathcal{D})$	✔				✔		✔
86	AquaticResourceObs	$\mathcal{SHOIQ}(\mathcal{D})$	✔				✔	✔	✔
87	Roles	$\mathcal{SRIN}$	✔						✔
88	SocialReality	$\mathcal{SRIN}$	✔						✔
89	Reaction	$\mathcal{SRIQ}$					✔		✔

ODP.org results summary Out of the 89 content ODPs 9 fit into the lightweight $\mathcal{EL}^{++}$ DL and therefore the OWL2EL profile. 13 fit into OWL2RL profile, and 22 into OWL2QL. One pattern, Reaction, violates the OWL2DL global restrictions which forces any ontology that includes this pattern out of OWL2DL. This particular pattern contains some disjoint properties axioms which specify that some properties which happen to be *Non-Simple* are disjoint with each other[5]. In terms of other prominent constructs, 77 patterns require inverse properties ($\mathcal{I}$), 45 patterns require cardinality restrictions of some form—either plain ($\mathcal{N}$) or qualified ($\mathcal{Q}$) min, max, or exact cardinality restrictions, 3 patterns require the implicit use of cardinality restrictions through the use of functional properties ($\mathcal{F}$), and 27 patterns use universal restrictions ($\forall$). Six patterns require OWL 2 constructs, that are not present in OWL 1, for their representation. Specifically, the Reaction pattern uses disjoint properties axioms and complex property chains, Roles uses complex property chains and anonymous inverse properties, and

[5] Non-Simple properties may not be used in certain positions in certain axioms, for example as operands in a DisjointProperties axiom. Roughly speaking, a property is Non-Simple if it is implied by a property chain (or transitive property).

SocialReality uses complex property chains. Four patterns (CatchRecord, Transition, Reaction and AquaticResourceObs) require the use of *qualified* cardinality restrictions ($\mathcal{Q}$), which are only available in OWL 2.

Table 2: Expressivity required for the ODPs from the ODPS.sf.net catalog.

Id	Name	Expressivity	DL	EL	RL	QL	$\forall$	$\sqcup$	Req.OWL2
1	NaryRelationship	$\mathcal{EL}^{++}$	✔	✔		✔			
2	CompositePropertyChain	$\mathcal{EL}^{++}$	✔	✔					✔
3	DefinedClassDescription	$\mathcal{EL}^{++}$	✔	✔					
4	NaryDataTypeRelationship	$\mathcal{ALEF}(\mathcal{D})$	✔						
5	Closure	$\mathcal{ALC}$	✔				✔		
6	EntityPropertyQuality	$\mathcal{ALCF}$	✔					✔	
7	ValuePartition	$\mathcal{ALCF}$	✔						
8	Selector	$\mathcal{ALCHF}$	✔					✔	
9	Exception	$\mathcal{ALCN}$	✔					✔	
10	EntityFeatureValue	$\mathcal{ALCQ}$	✔					✔	✔
11	InteractorRoleInteraction	$\mathcal{ALCQ}$	✔				✔	✔	✔
12	EntityQuality	$\mathcal{ALCIQ}$	✔				✔	✔	✔
13	AdaptedSEP	$\mathcal{S}$	✔					✔	
14	Sequence	$\mathcal{SHF}$	✔						
15	List	$\mathcal{SHN}$	✔					✔	

ODPS.sf.net results summary Out of the 15 pattern ontology documents, 3 fall into the OWL2EL profile, 0 fall into the OWL2RL profile, and 1 falls into the OWL2QL profile. One of these OWL2EL ontologies requires property chains, which are an OWL 2 construct. One of the patterns, EntityQuality, uses inverse properties ($\mathcal{I}$), 4 patterns use some form of explicit plain ($\mathcal{N}$) or qualified ($\mathcal{Q}$) cardinality restriction, while 5 ontologies require implicit cardinality restrictions due to the use of functional properties ($\mathcal{F}$). Only 3 patterns use universal restrictions ($\forall$), while 8 patterns use disjunction ($\sqcup$).

5 Analysis

Pattern Expressivity Requirements As can be seen from Tables 1 and 2, patterns from both the ODP.org and the ODPS.sf.net catalogs require a range of language expressivity, from the lightweight $\mathcal{EL}^{++}$ to the highly expressive languages $\mathcal{ALCHIN}$, $\mathcal{SHIN}$, $\mathcal{SHOIQ}$, and $\mathcal{SROIQ}$. Both catalogs lean towards requiring more expressive fragments of OWL, with many patterns that use constructs which bump up the expressivity from the base languages of $\mathcal{EL}^{++}$ or $\mathcal{AL}$. For example, in the ODP.org catalog it is typically the case that patterns require the use of inverse properties (77 out of 89 patterns) and cardinality restrictions (45 out of 89 patterns). It is also evident that both universal restrictions ($\forall$) and disjunctions ($\sqcup$) are sprinkled throughout the patterns in both patterns catalogs, with notable use universal restrictions in the ODP.org catalog, and disjunction within the ODPS.sf.net catalog. Universal restrictions are typically used to "close off" possibilities or model the local range of a property, whereas disjunctions are used to model "choices" or options for a property, so at first glance it makes sense that they appear in many patterns.

Pattern Expressivity and the OWL 2 Profiles While both pattern catalogs contain *some* patterns that can be represented within one or more of the OWL 2 profiles, it is clear that most of the patterns (59 out of 89 patterns from the ODP.org catalog and 12 out of 15 patterns from ODPS.sf.net catalog) cannot be represented in any languages corresponding to the profiles. This means that large swaths of patterns from both catalogs are "off limits" for ontology engineers targeting a specific profile.

Only 9 out of 89 and 3 out of 15 patterns from the ODP.org and ODPS.sf.net catalogs respectivelly can be represented in the OWL2EL profile language. One of the startlingly obvious reasons for this is that OWL2EL prohibits the use of inverse properties ($\mathcal{I}$), the use of cardinality restrictions ($\mathcal{N}$, $\mathcal{Q}$ or $\mathcal{F}$) and the use of universal restrictions ($\forall$). Interestingly, as far as the ODP.org catalog is concerned, more patterns fall into the OWL2RL and OWL2QL profiles than the OWL2EL profile. Unlike the OWL2EL profile, both of these profiles admit the use of inverse property axioms. Despite this OWL2QL is not strictly more expressive than OWL2EL, in fact it is a lightweight language profile that arguably puts more constraints on modellers than OWL2EL. This would seem to indicate inverse properties do play a major role in patterns violating the OWL2EL profile.

Pattern Expressivity and BioMedical Ontology Expressivity As mentioned previously, the OWL2EL profile is a pertinent profile for modelling and reasoning with biomedical ontologies. In fact, because of the prominence and importance biomedical ontologies OWL2EL was designed with these kinds of ontologies in mind. The language that underpins OWL2EL is expressive enough that it can be used to model typical biomedical ontologies, but its expressivity is limited to guarantee efficient reasoning. Indeed, OWL2EL reasoners such as ELK are able to classify large ontologies like SNOMED in a few seconds. To put things into perspective, more than half of the BioMedical ontologies contained in the NCBO BioPortal repository are OWL2EL ontologies [6], and the large medical ontology SNOMED expressly targets a fragment of this language in order to guarantee efficient reasoning [12]. It is therefore somewhat unfortunate that only a handful of ODPs can be expressed in OWL2EL. In essence, there is currently a tension between staying within a profile that offers fast and efficient reasoning and using ODPs. An interesting aspect of this tension is that patterns in the ODPS.sf.net catalog were specifically designed with biomedical ontology engineering in mind[6].

6 Towards Profile Friendly Patterns

Given the tension between typical biomedical ontology expressivity and the expressivity required to represent ODPs, an argument can be made in favour of producing versions of patterns that require limited expressivity. This argument also holds for other domains and the other profiles OWL2RL and OWL2QL. One

[6] It should be noted that the ODPS.sf.net catalog was compiled before OWL2EL was designed and published.

way of going about this would be to take existing patterns and rewrite or weaken them to conform to the required expressivity. In what follows we provide some examples of the ways in which the current ODP definitions could be modified to make them more usable with profile constrained ontologies.

Rewrite Cardinality Restrictions There is an abundance of patterns that required cardinality restrictions (45 out of 89 ontologies in the ODP.org catalog and 5 out of 15 in the ODPS.sf.net catalog). With both catalogs, it is plausible that cardinality restrictions were introduced into patterns either due to side-effects of particular tools, or due to modeller taste. This is a reasonable conclusion to arrive at because there are a high number of patterns which use min-one cardinality restrictions that could be directly replaced with existential restrictions without any loss of information[7]—in the case of the ODP.org catalog 24 patterns could be rewritten and in the case of ODPS.sf.net all 5 patterns which use cardinality restrictions could be rewritten.

Replace Universal Restrictions with Range Axioms Another construct that is prevalent throughout the patterns in ODP.org catalog but is not permitted in either OWL2EL or OWL2QL is the universal restriction ($\forall$). Upon casting an eye over patterns that use universal restrictions, it appears that many of them use these kinds of restrictions as simple local range constraints. For example, SubClassOf(A ObjectAllValuesFrom(hasPart B)) imposes a local range of B on the property hasPart for the class A. In patterns where the properties in these universal restrictions have limited usage (*i.e.* only one such universal restriction per property) it *may* be possible to replace the universal restriction with a global property range axiom (*i.e.*ObjectPropertyRange(hasPart B))[8]—these kinds of axioms are permitted in both OWL2EL and OWL2QL. Out of the 27 patterns from the ODP.org catalog that use universal restrictions, 23 patterns use them to impose one local range constraint on one property for one class. For these patterns it would be possible to write these local range constraints as global OWL range axioms without affecting the consistency and intended semantics of the pattern ontology.

Make Inverse Properties Optional The inverse properties axiom ($\mathcal{I}$) is prevalent throughout the ODP.org catalog and increases pattern expressivity leading to OWL2EL profile violation. From a philosophical point of view it is hard to say whether or not inverse properties are intrinsic to the ODPs in this catalog. However, an analysis of inverse property usage suggests that in some cases such usage could be due to the routine modelling practice of always declaring an in-

[7] The restriction ObjectMinCardinality(1 hasPart A) is semantically equivalent to the existential restriction ObjectSomeValuesFrom(hasPart A).

[8] Obviously, some care must be taken to ensure that multiple local ranges for a given class and property (possibly asserted in different patterns or in a domain ontology) do not produce an unsatisfiable range when converted to a global range and intersected with each other.

verse for a property[9]—irrespective of whether the inverse is used in a meaningful manner elsewhere in the ontology. As an illustration, consider that the axioms below appear in some pattern.

```
ObjectProperty(hasPart)
ObjectPropertyDomain(hasPart, A)
ObjectPropertyRange(hasPart, B)
...
ObjectInverseOf(hasPart, isPartOf)
ObjectPropertyDomain(isPartOf, B)
ObjectPropertyRange(isPartOf, A)
```

The property hasPart will be declared and used throughout the pattern, but 3 additional axioms will be also added: the inverse of hasPart (isPartOf) and the domain and range for this inverse as the "reverse" of the domain and range for hasPart (the primary property). In these cases, inverses could be made into an optional part of the design pattern that could be "bolted on" for domain or application ontologies that specifically require them, but left out for (biomedical) ontology engineers who want to remain with a profile such as OWL2EL. This optionality could be realised by offering different versions of the ODP or by splitting the ODP into separate ontologies which can be selected and imported by modellers only as required. Out of the 71 ontologies that contain inverse property axioms, 13 ontologies contain inverse usage as described above. A further 22 ontologies contain this "good practice" use of inverses plus asserting a property hierarchy for the inverses so that the inverse property hierarchy mirrors the primary property hierarchy. This hierarchy based kind of inverse usage could also be made optional.

7 Conclusions

In this paper we presented a study of the language expressivity required for using OWL ODPs. ODPs from the two main catalogs, ODP.org and ODPS.sf.net, were studied. Although there are a handful of patterns that can be represented using lightweight fragments of OWL most patterns require more heavyweight fragments containing inverse properties, cardinality restrictions, universal restrictions and disjunction. A small number of patterns require constructs that are only available in OWL 2, including property chains, disjoint properties and qualified cardinality restrictions. What is evident is that very few patterns can be represented in fragments of the language that are contained within one or more of the OWL 2 profiles. In particular, very few patterns, including ones specifically targeted at biomedical ontology construction, conform to the OWL2-EL profile. This means that there is a tension between using a language that was designed with biomedical ontologies in mind and using design patterns that were

[9] Often perceived as a good practice.

designed for biomedical ontologies. More generally, since all three OWL profiles were designed with the goal of supporting fast and efficient reasoning, modellers must currently make a choice between taking advantage of ODPs or taking advantage of high performance tools. An initial analysis of the constructs which lead to the high expressivity requirements of patterns suggest that some of these issues could be dealt with by rewriting patterns to use different constructs and making parts of patterns, especially those that use inverse properties, optional. As future work, it would be interesting to assess the impact of such changes on ontology engineering.

Acknowledgements This work was supported by the NIH Grants GM086587, HG004028, and LM007033. Mikel Egaña Aranguren is funded by the Marie Curie-COFUND Programme (FP7) of the European Union.

References

1. Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *IJCAI 05, Edinburgh, Scotland, UK, July 30-August 5, 2005*, 2005.
2. Natalya F. Noy et al. BioPortal: Ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*, 37(suppl 2):W170–W173, May 2009.
3. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994.
4. Aldo Gangemi and Valentina Presutti. *Ontology Design Patterns.* International Handbooks on Information Systems. Springer, 2009.
5. Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1):11–21, February 2011.
6. Matthew Horridge, Bijan Parsia, and Ulrike Sattler. The state of biomedical ontologies. In *BioOntologies 2011 15th–16th July, Vienna Austria*, 2011.
7. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *KR 2006, Lake District, United Kingdom*, pages 57–67. AAAI Press, June 2006.
8. Yevgeny Kazakov. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In *KR 2008, Sydney, Australia, September 16-19, 2008.*
9. Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language Profiles. W3C Recommendation, W3C – World Wide Web Consortium, October 2009.
10. Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 Web Ontology Language structural specification and functional style syntax. W3C Recommendation, W3C – World Wide Web Consortium, October 2009.
11. Valentina Presutti and Aldo Gangemi. Content Ontology Design Patterns as Practical Building Blocks for Web Ontologies. In *Proceedings of the 27th International Conference on Conceptual Modeling*, ER '08, pages 128–141, Berlin, Heidelberg, 2008. Springer-Verlag.
12. Kent A. Spackman. An examination of OWL and the requirements of a large health care terminology. In *OWLED 2007.*
13. Kent A. Spackman and Keith E. Campbell. SNOMED RT: A reference terminology for health care. In *In Proc. of AMIA Annual Fall Symposium*, 1997.