Safe and Economic Re-Use of Ontologies: A Logic-Based Methodology and Tool Support

Ernesto Jiménez-Ruiz[1], Bernardo Cuenca Grau[2], Ulrike Sattler[3], Thomas Schneider[3], and Rafael Berlanga[1]

[1] Universitat Jaume I, Spain, {berlanga,ejimenez}@uji.es
[2] University of Oxford, UK, berg@comlab.ox.ac.uk
[3] University of Manchester, UK, {sattler,schneider}@cs.man.ac.uk

1 Motivation

Ontology design and maintenance require an expertise in both the domain of application and the ontology language. Realistic ontologies typically model different aspects of an application domain at various levels of granularity; prominent examples are the National Cancer Institute Ontology (NCI)[4], which describes diseases, drugs, proteins, etc., and GALEN[5], which represents knowledge mainly about the human anatomy, but also about other domains such as drugs.

Established ontologies such as NCI and GALEN are used in various applications as *reference ontologies*, i.e., ontology developers reuse these ontologies and customise them for their specific needs. For example, ontology designers use classes from NCI or GALEN and refine them (e.g., add new sub-classes), generalise them (e.g., add new super-classes), or refer to them when expressing a property of some other class (e.g., define the class Polyarticular_JRA by referring to the class Joint from GALEN).

One of such reuse cases is the development of an ontology, called JRAO, that describes a kind of arthritis called JRA (Juvenile Rheumatoid Arthritis) within the Health-e-Child project.[6] Following the ILAR[7], JRAO describes in detail various kinds of JRA by means of the joints affected, the occurrence of fever, and the required treatment. GALEN and NCI contain information that is relevant to JRA such as detailed descriptions of the human joints as well as diseases and their symptoms. Figure 1 shows a fragment of NCI including the class for JRA as well as our reuse scenario, where $C_1, \ldots, C_7$ stand for subclasses of JRA to be defined in JRAO.

The JRAO developers want to reuse knowledge from NCI and GALEN for three reasons: (a) they want to save time through reusing existing ontologies rather than writing their own; (b) they value knowledge that is commonly accepted by the community and used in similar applications; (c) they are not experts in all areas covered by NCI and GALEN.

[4] Online browser: http://nciterms.nci.nih.gov/NCIBrowser/Dictionary.do, latest version: ftp://ftp1.nci.nih.gov/pub/cacore/EVS/NCI_Thesaurus
[5] http://www.co-ode.org/galen
[6] See http://www.health-e-child.org.
[7] Int. League of Associations for Rheumatology http://www.ilarportal.org/

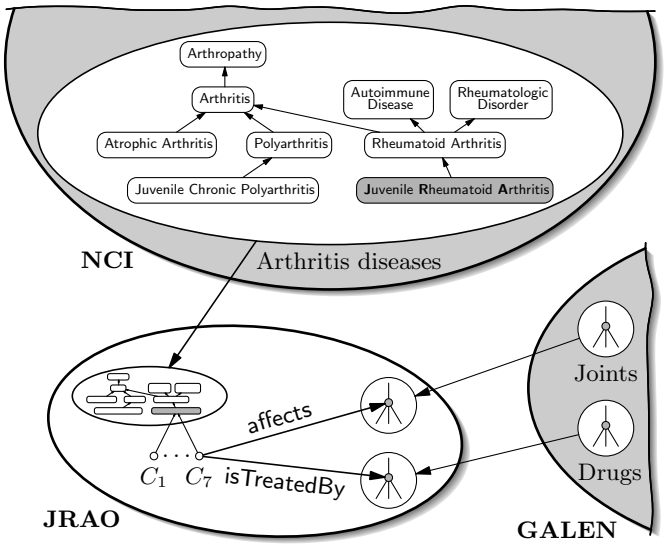

Figure 1. Constructing the ontology JRAO reusing fragments of GALEN and NCI

GALEN, NCI, and JRAO are written in OWL DL, and hence the semantics of OWL should be taken into account for ontology reuse. First, the developers of JRAO do not want to change the original meaning of classes reused from NCI or GALEN. For example, due to (b) and (c) above, if it followed from the union of JRAO and NCI that JRA is a subclass of GeneticDisorder and both are classes from NCI, then it should also follow from NCI alone. Second, only small parts of large ontologies like NCI and GALEN are relevant for our refinement of JRA. For efficiency and succinctness, the JRAO developers want to import as few axioms from NCI and GALEN as possible, yet they want to make sure that they import all axioms that are relevant for JRAO. For example, if it follows from the union of JRAO and the whole NCI that JRA is a subclass of RheumatologicDisorder, then this should also follow from the union of JRAO and the chosen fragment of NCI. That is, the JRAO developers should not see any difference between only adding the chosen fragments or the whole ontology to JRAO – apart from the fact that JRAO is smaller in the former case.

The following three guarantees summarise the above observations:

- (G1) The meaning of the imported classes and properties is not changed through axioms in the importing ontology.
- (G2↑) The fragment of the ontology to be imported is such that the ontology designer will see no difference between adding only this fragment or the whole ontology to their ontology.
- (G2↓) The fragment of the ontology to be imported is as small as possible.

(G1) and (G2↑) motivated "safe" in our title, and (G2↓) motivated "economic". In the following, we sketch the logical background of these guarantees and the

concepts we have developed to help provide these guarantees, and we propose a methodology for ontology design in reuse scenarios which is based on these concepts and guarantees and supported by a Protégé 4[8] plugin we are developing.

2 Logical Background – a Sketch

Based on the scenario in Section 1, we define the notions of a *conservative extension*, *safety*, and *module* [1, 2]. For simplicity, we restrict ourselves to OWL without individual names but with cardinality constraints, i.e., to the $\mathcal{SHIQ}$ fragment of OWL1.1. Moreover, we use $\sqsubseteq$ as the DL shorthand for subClassOf, $\mathsf{Sig}()$ to denote the *signature* of an ontology or an axiom, i.e., the set of class and property names used in this ontology or axiom, and we use $\models$ for the usual entailment relation, i.e., $\mathcal{O} \models \alpha$ means that $\mathcal{O}$ entails α. In this and the following section, we have omitted most of the technical details, which can be found in a technical report available at `http://www.cs.man.ac.uk/~schneidt/publ/safe-eco-reuse-report.pdf`. Finally, for simplicity, we assume that terms are reused literally, i.e., we ignore mapping rules but come back to them in Section 6.

2.1 Conservative Extensions, Safety and Modules

When reusing knowledge from NCI and GALEN, the developer of JRAO should not change the original meaning of the reused classes, see (G1). We formalise this requirement using the notion of a *conservative extension* [1, 3] and *safety* [2].

Definition 1 (Conservative Extension). *Let $\mathcal{O}_1 \subseteq \mathcal{O}$ be ontologies, and $\mathbf{S}$ a signature. We say that $\mathcal{O}$ is an* $\mathbf{S}$-conservative extension *of $\mathcal{O}_1$ if, for every axiom α with $\mathsf{Sig}(\alpha) \subseteq \mathbf{S}$, we have $\mathcal{O} \models \alpha$ iff $\mathcal{O}_1 \models \alpha$; $\mathcal{O}$ is a* conservative extension *of $\mathcal{O}_1$ if $\mathcal{O}$ is an $\mathbf{S}$-conservative extension of $\mathcal{O}_1$ for $\mathbf{S} = \mathsf{Sig}(\mathcal{O}_1)$.*

Definition 1 applies to our example as follows: $\mathcal{O}_1$ = NCI is the ontology to be reused, $\mathcal{O}$ is the union of JRAO and NCI, $\mathbf{S}$ represents the terms from NCI to be reused in JRAO such as JRA and Rheumatologic_Disorder, and α stands for any axiom over the reused terms only, e.g., JRA $\sqsubseteq$ Rheumatologic_Disorder. Now if $\mathcal{O}$ is an $\mathbf{S}$-conservative extension of NCI, then all entailments from $\mathcal{O}$ (JRAO $\cup$ NCI) concerning terms from NCI are already entailments of NCI and vice versa.

Definition 1 assumes that the reused ontology is static, i.e., that we know both $\mathcal{O}$ and $\mathcal{O}_1$. In practice, however, ontologies such as NCI are under development and may evolve beyond the control of the JRAO developers. Thus, it would be convenient to reuse the axioms of NCI on demand via a reference in JRAO such that the developers of JRAO need not commit to a particular version of NCI. The notion of *safety* [2] is a stronger version of conservative extension that abstracts from the particular ontology to be reused and focuses only on the way *terms* are reused. In our example, we would like to make sure that JRAO is *safe* for the class and property names $\mathbf{S}$ it uses from NCI because this guarantees

[8] Ontology Editor Protégé 4: `http://www.co-ode.org/downloads/protege-x/`

that JRAO does not change their meaning as defined in NCI—regardless of the version of NCI under consideration. Hence safety ensures (G1).

Definition 2 (Safety for a Signature). *Let $\mathcal{O}$ be an ontology and* $\mathbf{S}$ *a signature. We say that* $\mathcal{O}$ is safe for $\mathbf{S}$ *if, for every ontology* $\mathcal{O}'$ *with* $\mathsf{Sig}(\mathcal{O}) \cap \mathsf{Sig}(\mathcal{O}') \subseteq \mathbf{S}$*, we have that* $\mathcal{O} \cup \mathcal{O}'$ *is a conservative extension of* $\mathcal{O}'$*.*

As mentioned in Section 1, we would also like to import only (hopefully small) fragments of NCI and GALEN—provided that we can be sure to not lose relevant information. We formalise this idea using the notion of a *module* [2]. In our example, assume we have added to JRAO only a module of NCI for a given signature $\mathbf{S}$, and we want check an entailment over JRAO that uses only NCI terms from $\mathbf{S}$. Then we would get exactly the same answers as if we had added the whole NCI to JRAO. Hence every module guarantees (G2↑) from above, and we will see below how we can compute small such modules to satisfy (G2↓).

Definition 3 (Module for a Signature). *Let $\mathcal{O}_1' \subseteq \mathcal{O}'$ be ontologies and* $\mathbf{S}$ *a signature. We say that* $\mathcal{O}_1'$ is a module for $\mathbf{S}$ in $\mathcal{O}'$ *(or an* $\mathbf{S}$-module in $\mathcal{O}'$*) if, for every ontology* $\mathcal{O}$ *with* $\mathsf{Sig}(\mathcal{O}) \cap \mathsf{Sig}(\mathcal{O}') \subseteq \mathbf{S}$*, we have that* $\mathcal{O} \cup \mathcal{O}'$ *is a conservative extension of* $\mathcal{O} \cup \mathcal{O}_1'$ *for* $\mathsf{Sig}(\mathcal{O})$*.*

2.2 Checking safety and computing modules

Summing up, using terms $\mathbf{S}$ in a *safe* way guarantees that we do not change the meaning of terms in $\mathbf{S}$ (G1), and importing a *module* $\mathcal{O}' \subseteq \mathcal{O}$ for $\mathbf{S}$ instead of $\mathcal{O}$ ensures that we have imported all information about $\mathbf{S}$ from $\mathcal{O}$, (G2↑). Next we discuss how we can test safety and compute hopefully small modules (G2↓). First the bad news: the decision problems associated with conservative extensions, safety and modules—e.g., given a fragment $\mathcal{O}' \subseteq \mathcal{O}$, is $\mathcal{O}'$ an $\mathbf{S}$-module of $\mathcal{O}$?—are undecidable for $\mathcal{SHOIQ}$ [4, 2]. Now the good news: we have found sufficient conditions for safety and modules. That is, if an ontology satisfies these conditions, then we can guarantee that it is safe/a module, but the converse does not necessarily hold [2], i.e., we guarantee (G1) and (G2↑), but we might be a bit too restrictive regarding (G1) and we only approximate (G2↓) in the sense that our module might contain superfluous axioms. A particularly useful condition is *locality* [2] since it is widely applicable in practice and it can be checked syntactically.

As mentioned in Section 1, when using a term from NCI or GALEN, the JRAO developers may refine it, extend it, or refer to it for expressing a property of another term. The simultaneous refinement and generalisation of a given "external" term, however, may compromise (G1). For example, JRAO cannot simultaneously contain the following axioms:

$$\mathsf{Polyarticular\_JRA} \sqsubseteq \underline{\mathsf{JRA}} \tag{1}$$

$$\underline{\mathsf{Juvenile\_Chronic\_Polyarthritis}} \sqsubseteq \mathsf{Polyarticular\_JRA} \tag{2}$$

where the underlined class names are reused from NCI, see Figure 1. These axioms imply that Juvenile_Chronic_Polyarthritis is a subclass of JRA, and therefore may *change* the meaning of the reused terms, e.g., if this subclass relationship is not a consequence of NCI or, even worse, if NCI implies that they are disjoint: in this case, a class from NCI becomes unsatisfiable through our import. Hence an ontology containing axioms (1) and (2) is not safe w.r.t. $\mathbf{S} = \{\mathsf{JRA}, \mathsf{Juvenile\_Chronic\_Polyarthritis}\}$. Thus, to guarantee safety, we need to ask the ontology designer whether she wants to refine or generalise the reused terms from an ontology, and then either ask her to restrict herself to use these terms only in a *refinement-safe* way, e.g., in axioms like (1), or only in a *generalisation-safe* way, e.g., in axioms like (2), but not in both. In this context, if we want to reference external terms in our ontology and guarantee their safety, then we can choose refinement-safety: e.g., the axiom $\mathsf{Polyarticular\_JRA} \sqsubseteq \geqslant 5\, \mathsf{affects}.\underline{\mathsf{Joint}}$ is refinement-safe w.r.t. $\mathbf{S} = \{\mathsf{Joint}\}$. The definition of what it means for an axiom to be refinement- or generalisation-safe uses the notion of locality and can be found in [2]. For our purpose here, it suffices to say that these are syntactic conditions and that they can indeed be used to guarantee (G1): e.g., if JRAO uses all terms from NCI in a refinement-safe way, then it is guaranteed that no terms from NCI change their meaning through their usage in JRAO.

Similarly, making use of locality, we have defined *upper* and *lower* modules for a signature $\mathbf{S}$ from an ontology $\mathcal{O}$, and proved that these modules are indeed $\mathbf{S}$-modules in $\mathcal{O}$ as defined in Definition 3. Upper and lower modules are, again, defined syntactically—see the report for details, where they are called $\bot$- and $\top$-modules. They can be computed in polynomial time [2]. Hence we can guarantee (G2$\uparrow$) but only approximate (G2$\downarrow$) since our modules might not be minimal.

Upper and lower modules enjoy a property which determines their scope: let $\mathcal{O}_1$ ($\mathcal{O}_2$) be an upper module (lower module) for $\mathbf{S}$ in $\mathcal{O}$, then $\mathcal{O}_1$ ($\mathcal{O}_2$) contains all super-classes (sub-classes) in $\mathcal{O}$ of all classes in $\mathbf{S}$. That is, if $\alpha := (X \sqsubseteq Y)$, $\beta := (Y \sqsubseteq X)$, for X, Y class names, then $\mathcal{O}_1 \models \alpha$ iff $\mathcal{O} \models \alpha$ and $\mathcal{O}_2 \models \beta$ iff $\mathcal{O} \models \beta$ [2]. For example, if we were to reuse the class JRA from NCI as shown in Figure 1, the upper module for a signature that contains JRA would also contain all the super-classes of JRA in NCI, namely Rheumatoid_Arthritis, Autoimmune_Disease, Rheumatologic_Disorder, Arthritis, and Arthropathy. Since an upper module is a module, it will contain all axioms necessary for entailing these subclass relations.

Finally, given $\mathcal{O}$ and $\mathbf{S}$, there is a unique minimal upper module and a unique minimal lower module for $\mathbf{S}$ in $\mathcal{O}$, and both can be computed efficiently [2]. In Section 4.1, we will report on experiments that show that our modules are often small, yet sometimes indeed not minimal.

3 A Novel Methodology for Ontology Reuse

Based on our scenario in Section 1 and the theory of modularity sketched in Section 2, we propose a novel methodology for designing an ontology when knowledge is to be borrowed from several external ontologies. This methodology pro-

vides precise guidelines for ontology developers to follow, and incorporates our guarantees. We propose the working cycle given in Figure 2. This cycle consists of an *offline phase*—which is performed independently from the current contents of the external ontologies—and an *online phase*—where knowledge from the external ontologies is extracted and transferred into the current ontology. Note that the separation between offline and online is not strict: The first phase is called "offline" simply because it does not need to be performed online. However, the user may still choose to do so.

The Offline Phase starts with the ontology $\mathcal{O}$ being developed, e.g., JRAO. The ontology engineer specifies the set $\mathbf{S}$ of terms to be reused from external ontologies and associates, to each term, the external ontology from which it will be borrowed. In Figure 2 this signature selection is represented in the *Repeat* loop: each $\mathbf{S}_i \subseteq \mathbf{S}$ represents the external terms to be borrowed from a particular ontology $\mathcal{O}'_i$; in our example, we have $\mathbf{S} = \mathbf{S}_1 \uplus \mathbf{S}_2$, where $\mathbf{S}_1$ is associated with NCI and contains JRA, and $\mathbf{S}_2$ is associated with GALEN and contains terms related to joints and drugs. This part of the offline phase may involve an "online" component since the developer may browse through the external ontologies to choose the terms she wants to import.

Next, the ontology developer decides, for each $\mathbf{S}_i$, whether she wants to refine or generalise the terms from this set. For instance, in the reuse example shown in Figure 1, the terms from NCI are refined since we introduce subclasses $C_1, \ldots, C_7$ of the NCI term JRA. In both cases, the user may also reference the external terms in the domain or range of some property; in our example, certain types of JRA are defined by referencing classes in GALEN (e.g., joints) via properties affects and isTreatedBy. As argued in Section 1, refinement and generalisation, combined with reference, constitute the main possible intentions when reusing external knowledge. Therefore it is reasonable for the user, both from the modelling and tool design perspectives, to declare her intentions. These declarations are made in the *For* loop in Figure 2.

At this stage, we want to ensure (G1), i.e., that the designer of $\mathcal{O}$ does not change the original meaning of the reused classes and properties, independently of their meaning in the external ontologies. This is achieved if, for each set of external terms $\mathbf{S}_i$, $\mathcal{O}$ uses all terms from $\mathbf{S}_i$ in a refinement-safe (generalisation-safe) way in case that the designer has chosen the refinement (generalisation) view for $\mathbf{S}_i$. In case this test fails, say for $\mathbf{S}_j$, the designer of $\mathcal{O}$ may choose to give up on (G1) (and possibly change the meaning of terms from $\mathbf{S}_j$) or to go back and "repair" the axioms that caused this failure.

In the Online Phase, the relevant knowledge from each external ontology is imported into $\mathcal{O}$. Here we aim at ensuring (G2↑) and (G2↓), i.e., we want to import as few axioms as possibly, while not missing any axiom relevant to the reused terms: i.e., we want small modules in the sense of Definition 3.

As shown in Figure 2, the import for each external ontology $\mathcal{O}'_i$ is performed in four steps. First, $\mathcal{O}'_i$ is loaded; by doing so, the ontology engineer commits to a particular version of it. Second, the scope of the module to be extracted from $\mathcal{O}'_i$ is customised; in practice, this means that the ontology engineer is given a view of

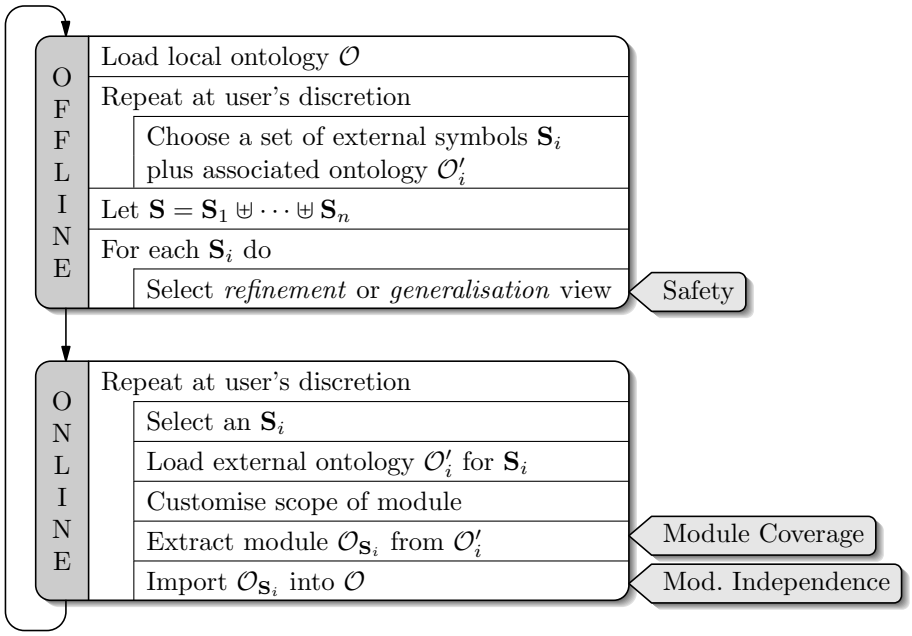

Figure 2. The two phases of import with the required guarantees

$\mathcal{O}'_i$ and enabled to extend $\mathbf{S}_i$ by specifying, e.g., that "the module has to contain the class 'joint', all its direct super-classes and two levels of its sub-classes". In the third step, the actual fragment of $\mathcal{O}'_{\mathbf{S}_i}$ is extracted from $\mathcal{O}'_i$. At this stage, we should ensure that the extracted fragment is a module for the customised signature according to Definition 3, i.e., it should ensure (G2↑), and we refer the reader to [2] for a description of how this can be achieved. Finally, the actual module $\mathcal{O}_{\mathbf{S}_i}$ is imported, and $\mathcal{O}$ evolves to $\mathcal{O} \cup \mathcal{O}_{\mathbf{S}_i}$. As a consequence, it could be the case that (G1) with respect to the other external ontologies might be compromised. This is clearly undesirable and hence we formulate the following guarantee:

(G3) the other guarantees (G1) and (G2↑) are independent of the order in which we actually extend our ontology with modules. That is, if our ontology is (refinement- or generalisation-)safe for a set of terms $\mathbf{S}_i$, then it is still (refinement- or generalisation-)safe after importing other modules.

(G3) is guaranteed in case that all signatures $\mathbf{S}_i$ are disjoint and $\mathcal{O}$ is local w.r.t. every $\mathbf{S}_i$.

4 The Ontology Reuse Tool

We are developing a Protégé 4 plugin that supports the methodology presented in Section 3. The plugin and user manual are available at `http://krono.act.uji.es/people/Ernesto/safety-ontology-reuse`.

The *offline phase* first involves the selection of the external entities. Our plugin provides functionality for declaring entities as external and for defining the external ontology URI for the selected entities; this information is stored in the ontology using OWL 1.1 annotations [5]. We use an ontology annotation axiom per external ontology, an entity annotation axiom to declare an entity external, and an entity annotation axiom per external entity to indicate its external ontology. The set of external entities with the same external ontology URI can be viewed as one of the $\mathbf{S}_i$. The UI of the plugin also allows for the specification, for each external ontology, whether it will be refined or generalised. The tool then allows for safety checking of the ontology w.r.t. each group of external terms separately. The safety check uses refinement-safety (generalisation-safety) for signature groups that adopt the refinement (generalisation) view. Axioms violating safety conditions are appropriately displayed. In this phase, our tool does allow the user to work completely offline, without the need of extracting and importing external knowledge, and even without knowing exactly from which ontology the reused entities will come from. Indeed, the specification of the URI of the external ontologies is optional at this stage, and, even if indicated, such URI may not refer to a real ontology, but it may simply act as a temporary name.

In the *online phase*, the user chooses external ontologies and imports axioms from them. At this stage, the groups of external terms to be imported should refer to the location of a "real" external ontology. Once an external ontology has been selected for import, the signature selected for it can be customised by adding super- and sub-classes of the selected classes. The tool provides previews of the class hierarchy of the external ontology for this purpose. Once the signature under consideration has been customised, a module satisfying (G2$\uparrow$) and approximating (G2$\downarrow$) is extracted. The user can preview it in a separate frame, and either import it or cancel the process and come back to the signature customisation stage. The user can also import the whole external ontology instead of importing a module. Finally, since we guarantee (G3), the order in which modules are imported is irrelevant.

Currently, the import of a module is done "by value", in the sense that the module becomes independent from the original ontology: if the external ontology on the Web evolves, the previously extracted module will not change.

4.1 Evaluation

So far, we have demonstrated our tool to various ontology developers[9] who have expressed great interest and provided us with useful feedback regarding its

[9] Thanks to Elena Beißwanger, Sebastian Brandt, Alan Rector, and Holger Stenzhorn for valuable comments and feedback.

improvements. In the technical report `http://www.cs.man.ac.uk/~schneidt/publ/safe-eco-reuse-report.pdf`, we describe experiments we have performed to show that locality-based modules are good approximations for (G2↓), i.e., they may not be minimal, but they are reasonably sized compared to the whole ontology and compared to other fragment extraction mechanisms.

5 Related Work

Several ontology engineering methodologies have been proposed; prominent examples are Methontology [6], On-To-Knowledge (OTK) [7], and ONTOCLEAN [8]. These methodologies, however, do not address ontology development scenarios involving reuse. Our proposed methodology is complementary and can be used in combination with them.

In the last few years, a growing body of work has been developed addressing ontology modularisation, mapping, alignment, merging, integration, and segmentation, see [9, 10, 11] for surveys. This field is diverse and has originated from different communities. In particular, numerous techniques have been proposed for extracting fragments of ontologies. Most of them, such as [12, 13, 14], rely on syntactic heuristics for detecting relevant axioms and do not attempt to formally specify the intended outputs and thus it is unclear which guarantees they provide.

Finally, there are various proposals for "safely" combining ontologies or modules; most of these proposals, such as $\mathcal{E}$-connections, Distributed Description Logics and Package-based Description Logics propose a specialised semantics for controlling the interaction between the importing and the imported modules to avoid side-effects; for an overview see [15]. In contrast, here we assume that reuse is performed by simply building the logical union of the axioms in the modules under the standard semantics; instead, we provide the user with a collection of reasoning services, such as safety testing, to check for side-effects. Our paper is based on other work on modular reuse of ontologies [16, 17, 4, 3] which enables us to provide the necessary guarantees. We extend this work with a methodology and tool support.

6 Future Work

We aim at extending the tool support so that the user can "shop" for terms to reuse: it will allow to browse an ontology for terms to reuse and provide a simple mechanism to pick them and, on "check-out", will compute the relevant module. Next, we plan to carry out a user study to assess the usefulness of the interface and how to improve it. Finally, our current tool support implements a "by value" mechanism: modules are extracted at the user's request. In addition, we would like to support reuse "by reference", i.e., we want to store information in the importing ontology that allows for automated updates of the imported modules. Finally, we plan to extend this approach with a mapping mechanism so that we can (safely) rename external terms.

Acknowledgements

This work was partially supported by the PhD Fellowship Program of the *Generalitat Valenciana*, by the *Fundació Caixa Castelló-Bancaixa*, and by the UK EPSRC grant no. EP/E065155/1.

References

[1] Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In Doherty, P., Mylopoulos, J., Welty, C., eds.: Proc. of KR-06, AAAI Press (2006) 187–197

[2] Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. of Artificial Intelligence Research **31** (2008) 273–318

[3] Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: Proc. of IJCAI-07, AAAI (2007) 453–459

[4] Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: A logical framework for modularity of ontologies. In: Proc. of IJCAI-07, AAAI (2007) 298–304

[5] Motik, B., Patel-Schneider, P.F., Horrocks, I.: OWL 1.1 Web Ontology Language Structural Specification and Functional-Style Syntax. W3C Member Submission (2007)

[6] M. Fernandez, A. Gomez-Perez, e.a.: Methontology: From ontological art towards ontological engineering. In: AAAI, Stanford, USA. (1997)

[7] Sure, Y., Staab, S., Studer, R.: On-to-knowledge methodology. In: In Handbook on Ontologies. Edited by S. Staab and R. Studer (eds.). Springer. (2003)

[8] Guarino, N., Welty, C.: Evaluating ontological decisions with ontoclean. Commun. ACM **45**(2) (2002) 61–65

[9] Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: The state of the art. The Knowledge Engineering Review **18** (2003) 1–31

[10] Noy, N.F.: Semantic integration: A survey of ontology-based approaches. SIGMOD Record **33**(4) (2004) 65–70

[11] Noy, N.F.: Tools for mapping and merging ontologies. In Staab, S., Studer, R., eds.: Handbook on Ontologies. International Handbooks on Information Systems. Springer (2004) 365–384

[12] Noy, N., Musen, M.: The PROMPT suite: Interactive tools for ontology mapping and merging. Int. J. of Human-Computer Studies **6**(59) (2003)

[13] Seidenberg, J., Rector, A.L.: Web ontology segmentation: analysis, classification and use. In: Proc. of WWW 2006, ACM (2006) 13–22

[14] Jiménez-Ruiz, E., Berlanga, R., Nebot, V., Sanz, I.: Ontopath: A language for retrieving ontology fragments. In: Proc. of ODBASE, LNCS. (2007) 897–914

[15] Cuenca Grau, B., Kutz, O.: Modular ontology languages revisited. In: Proc. of the Workshop on Semantic Web for Collaborative Knowledge Acquisition. (2007)

[16] Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: extracting modules from ontologies. In Williamson, C.L., Zurko, M.E., Patel-Schneider, P.F., Shenoy, P.J., eds.: WWW, ACM (2007) 717–726

[17] Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Ontology reuse: Better safe than sorry. In: Proc. of DL 2007. Volume 250 of CEUR WS Proc. (2007)