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.

Ontologies such as NCI and GALEN are used in biomedical applications as *reference ontologies*, i.e., ontology developers reuse and customise them for their specific needs. For example, concepts from NCI or GALEN are used and refined (e.g., by adding sub-concepts), generalised (e.g., by adding super-concepts), or referred to when expressing a property of some other concept (e.g., by defining the concept Polyarticular_JRA via reference to the concept Joint from GALEN).

One of such use cases is the development within the Health-e-Child project of an ontology, called JRAO, to describe a kind of arthritis called JRA (Juvenile Rheumatoid Arthritis).[6] Following the ILAR[7], JRAO describes the kinds of JRA. Those are distinguished by several factors such as the joints affected or the occurrence of fever, and each type of JRA requires a different 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 gives a fragment of NCI that defines JRA. It also shows our reuse scenario, where $C_1, \ldots, C_7$ refer to the kinds 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. This project aims at creating a repository of ontologies that can be used by clinicians in various applications.
[7] Int. League of Associations for Rheumatology http://www.ilarportal.org/

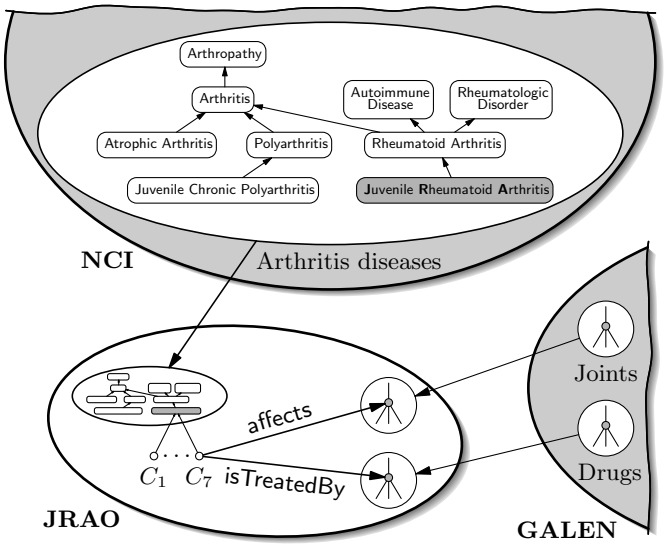

Figure 1. Constructing the ontology JRAO reusing fragments of GALEN and NCI

GALEN, NCI, and JRAO are written in OWL DL; hence their semantics should be taken into account for ontology reuse. More precisely, the following guarantees should be provided. First, when reusing knowledge from NCI and GALEN, the JRAO developers do not want to change the original meaning of the reused concepts. For example, due to (b) and (c) above, if it followed from the union of JRAO and NCI that JRA is a genetic disorder, then this should also follow from NCI alone. Second, only small parts of large ontologies like NCI and GALEN are relevant to the sub-types of JRA. For efficiency and succinctness, the JRAO developers want to import only those axioms from NCI and GALEN that are relevant for JRAO. By importing only fragments of NCI and GALEN, one should not lose important information; for example, if it follows from the union of JRAO and NCI that JRA is a rheumatologic disorder, then this also follows from the union of JRAO and the chosen fragment of NCI.

Our scenario has two main points in common with other ontology design scenarios: the ontology developer wants to reuse knowledge without changing it, and also to import only the relevant parts of an existing ontology. To support these scenarios whilst providing the two above guarantees, a logic-based approach to reuse is required. Current tools that support reuse, however, do not implement a logic-based solution and thus do not provide the above guarantees—and neither do existing guidelines and "best practices" for ontology design.

In this paper, we propose a methodology for ontology design in scenarios involving reuse which is based on a well-understood logic-based framework [1]. We describe a tool that implements this methodology and report on experiments.

2 Preliminaries on Modularity

Based on the scenario in Section 1, we define the notions of a *conservative extension*, *safety*, and *module* [2, 1]. For simplicity, we restrict ourselves to $\mathcal{SHIQ}$. In this and the following section, we have omitted a few 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`.

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 concepts. This requirement can be formalised using the notion of a *conservative extension* [2, 3]. In the following, we use $\mathsf{Sig}()$ to denote the signature of an ontology or an axiom.[8]

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 $\mathbf{S} = \mathsf{Sig}(\mathcal{O}_1)$.*[9]

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 symbols reused from NCI, such as JRA and $\mathsf{Rheumatologic\_Disorder}$, and α stands for any axiom over the reused symbols only, e.g., $\mathsf{JRA} \sqsubseteq \mathsf{Rheumatologic\_Disorder}$.

Definition 1 assumes that the reused ontology is static. In practice, however, ontologies such as NCI are under development and may evolve beyond the control of the JRAO developers. Thus, it is convenient to make the axioms of NCI available on demand via a reference such that the developers of the JRAO need not commit to a particular version of NCI. The notion of *safety* [1] is a stronger version of conservative extension that abstracts from the particular ontology to be reused and focuses only on the reused *symbols*.

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, by importing only a fragment of NCI and GALEN, one should not lose important information. This idea can be formalised using the notion of a *module* [1]. Intuitively, when checking an arbitrary entailment over the signature of the JRAO, importing a module of NCI should give exactly the same answers as if the whole NCI had been imported.

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 $\mathsf{Sig}(\mathcal{O})$.*

[8] $\mathcal{SHIQ}$-axioms are concept or role inclusions, or transitivity statements.
[9] $\mathcal{SHIQ}$ is a monotonic logic; hence the "only if" in "$\mathcal{O} \models \alpha$ iff $\mathcal{O}_1 \models \alpha$" is trivial.

The notions of safety and module are related as follows: if $\mathcal{O}' \setminus \mathcal{O}'_1$ is safe for $\mathbf{S} \cup \mathsf{Sig}(\mathcal{O}'_1)$, then $\mathcal{O}'_1$ is an $\mathbf{S}$-module in $\mathcal{O}'$ [1].

2.2 Locality Conditions

The decision problems associated with conservative extensions, safety and modules—are undecidable for $\mathcal{SHOIQ}$ [4, 1]. Sufficient conditions for safety have been proposed: if an ontology satisfies such conditions, then we can guarantee that it is safe, but the converse does not necessarily hold [1]. Such conditions could be used for extracting modules. A particularly useful condition is *locality* [1]: it is widely applicable in practice and it can be checked syntactically.

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

$$\mathsf{Polyarticular\_JRA} \sqsubseteq \underline{\mathsf{JRA}} \quad (\bot\text{-local}) \tag{1}$$

$$\underline{\mathsf{Juvenile\_Chronic\_Polyarthritis}} \sqsubseteq \mathsf{Polyarticular\_JRA} \quad (\top\text{-local}) \tag{2}$$

where the underlined concepts are reused from NCI, see Figure 1. These axioms imply $\mathsf{Juvenile\_Chronic\_Polyarthritis} \sqsubseteq \mathsf{JRA}$, and therefore an ontology containing axioms (1) and (2) is not safe w.r.t. $\mathbf{S} = \{\mathsf{JRA}, \mathsf{Juvenile\_Chronic\_Polyarthritis}\}$. Thus, when designing sufficient conditions for safety, we are faced with a fundamental choice depending on whether the ontology designer wants to reuse or generalise the reused concepts. Each choice leads to a different locality condition.

The following definition introduces these conditions and refers to Figure 2: $A^\dagger$ and $R^\dagger$ are concept and role names not in $\mathbf{S}$; C and R denote arbitrary concepts and roles.

Definition 4 (Syntactic $\bot$-Locality and $\top$-Locality). *Let $\mathbf{S}$ be a signature. An axiom α is* $\bot$-local w.r.t. $\mathbf{S}$ *($\top$-local w.r.t $\mathbf{S}$) if $\alpha \in \mathsf{Ax}(S)$, as defined in Figure 2 (a) ((b)). An ontology $\mathcal{O}$ is $\bot$-local ($\top$-local) w.r.t. $\mathbf{S}$ if α is $\bot$-local ($\top$-local) w.r.t. $\mathbf{S}$ for all $\alpha \in \mathcal{O}$.*

Axiom (2) is $\top$-local w.r.t. $\mathbf{S} = \{\mathsf{Juvenile\_Chronic\_Polyarthritis}\}$, and Axiom (1) is $\bot$-local w.r.t. $\mathbf{S} = \{\mathsf{JRA}\}$. Note that the locality conditions allow us to refer to a reused concept for expressing a property of some other concept; for example, the axiom $\mathsf{Polyarticular\_JRA} \sqsubseteq \geqslant 5\, \mathsf{affects}.\underline{\mathsf{Joint}}$ is $\bot$-local w.r.t. $\mathbf{S} = \{\mathsf{Joint}\}$.

Both $\top$-locality and $\bot$-locality are sufficient for safety and locality can be used for defining modules: we say that $\mathcal{O}_1 \subseteq \mathcal{O}$ is a $\bot$-module ($\top$-module) in $\mathcal{O}$ for $\mathbf{S}$ if $\mathcal{O} \setminus \mathcal{O}_1$ is $\bot$-local ($\top$-local) w.r.t. $\mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_1)$ [1].

It is clear that $\bot$-modules and $\top$-modules satisfy Definition 3: if $\mathcal{O}_1$ is $\bot$-module or a $\top$-module for $\mathbf{S}$ in $\mathcal{O}$, then $\mathcal{O}_1$ is an $\mathbf{S}'$-module in $\mathcal{O}$ for $\mathbf{S}' = \mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_1)$ [1]. These modules enjoy a property which determines their scope: let $\mathcal{O}_1$ ($\mathcal{O}_2$) be a $\bot$-module ($\top$-module) for $\mathbf{S}$ in $\mathcal{O}$, then $\mathcal{O}_1$ ($\mathcal{O}_2$) contains all super-concepts (sub-concepts) in $\mathcal{O}$ of all concepts in $\mathbf{S}$–that is if $\alpha := (X \sqsubseteq Y)$, $\beta :=$

(a) $\bot$-Locality	Let $C^\dagger \in \mathbf{Con}(\bar{\mathbf{S}})$, $C^{\mathbf{s}}_{(i)} \in \mathbf{Con}(\mathbf{S})$
$\mathbf{Con}(\bar{\mathbf{S}})$ ::=	$A^\dagger \mid \neg C^{\mathbf{s}} \mid C \sqcap C^\dagger \mid C^\dagger \sqcap C \mid \exists R.C^\dagger \mid \geqslant n\, R.C^\dagger \mid \exists R^\dagger.C \mid \geqslant n\, R^\dagger.C$
$\mathbf{Con}(\mathbf{S})$ ::=	$\neg C^\dagger \mid C^{\mathbf{s}}_1 \sqcap C^{\mathbf{s}}_2$
$\mathsf{Ax}(\mathbf{S})$ ::=	$C^\dagger \sqsubseteq C \mid C \sqsubseteq C^{\mathbf{s}} \mid R^\dagger \sqsubseteq R \mid \mathsf{Trans}(R^\dagger)$

(b) $\top$-Locality	Let $C^{\mathbf{s}} \in \mathbf{Con}(\mathbf{S})$, $C^\dagger_{(i)} \in \mathbf{Con}(\bar{\mathbf{S}})$
$\mathbf{Con}(\mathbf{S})$ ::=	$\neg C^\dagger \mid C \sqcap C^{\mathbf{s}} \mid C^{\mathbf{s}} \sqcap C \mid \exists R.C^{\mathbf{s}} \mid \geqslant n\, R.C^{\mathbf{s}}$
$\mathbf{Con}(\bar{\mathbf{S}})$ ::=	$A^\dagger \mid \neg C^{\mathbf{s}} \mid C^\dagger_1 \sqcap C^\dagger_2 \mid \exists R^\dagger.C^\dagger \mid \geqslant n\, R^\dagger.C^\dagger$
$\mathsf{Ax}(\mathbf{S})$ ::=	$C^{\mathbf{s}} \sqsubseteq C \mid C \sqsubseteq C^\dagger \mid R \sqsubseteq R^\dagger \mid \mathsf{Trans}(R^\dagger)$

Figure 2. Syntactic locality conditions

($Y \sqsubseteq X$), for X, Y concept names, then $\mathcal{O}_1 \models \alpha$ iff $\mathcal{O} \models \alpha$ and $\mathcal{O}_2 \models \beta$ iff $\mathcal{O} \models \beta$ [1]. For example, if we reused the concept JRA from NCI as shown in Figure 1 by extracting a $\bot$-module for a signature that contains JRA, such a module would contain all the super-concepts of JRA in NCI, namely Rheumatoid_Arthritis, Autoimmune_Disease, Rheumatologic_Disorder, Arthritis, and Arthropathy. Since such a fragment is a module, it will contain the axioms necessary for entailing those subsumption relations between the listed concepts that hold in NCI.

Finally, given $\mathcal{O}$ and $\mathbf{S}$, there is a unique minimal $\bot$-module and a unique minimal $\top$-module for $\mathbf{S}$ in $\mathcal{O}$, which can be computed in polynomial time [1]. We denote these modules by $\mathsf{UpMod}(\mathcal{O}, \mathbf{S})$ and $\mathsf{LoMod}(\mathcal{O}, \mathbf{S})$.

3 A Novel Methodology for Ontology Reuse

Based on our scenario in Section 1 and the theory of modularity summarised in Section 2, we propose a novel methodology for designing an ontology when knowledge is to be borrowed from several external ontologies. This methodology provides precise guidelines for ontology developers to follow, and ensures that logical guarantees will hold at certain stages of the design process. We propose the working cycle given in Figure 3. It 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. This 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 symbols to be reused, and associates to each symbol the external ontology from which it will be borrowed. In Figure 3 this step is represented in the *Repeat* loop: each $\mathbf{S}_i \subseteq \mathbf{S}$ stands for the external symbols 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 symbols related to joints and drugs. This part of

the offline phase may involve an "online" component where the developer may browse through the external ontologies selecting symbols for import.

In the subsequent *For* loop, the ontology developer decides, for each $\mathbf{S}_i$, whether she wants to refine or generalise the symbols from $\mathbf{S}_i$. In the example in Figure 1, the concept JRA from NCI is refined by sub-concepts $C_1, \ldots, C_7$. In both cases, the user may also reference the external symbols via roles; in our example, certain types of JRA are defined by referencing concepts in GALEN (e.g., joints) via the roles affects and isTreatedBy. As argued in Section 1, refinement and generalisation, combined with reference, are 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 intention.

At this stage, we want to ensure that the designer of $\mathcal{O}$ does not change the original meaning of the reused concepts, independently of what their meaning is in the external ontologies. This requirement can be formalised using safety:

Definition 5 (Safety Guarantee). *The ontology* $\mathcal{O}$ guarantees safety w.r.t. the signatures $\mathbf{S}_1, \ldots, \mathbf{S}_n$ *if* $\mathcal{O}$ *is safe for* $\mathbf{S}_i$ *for all* $1 \leq i \leq n$.

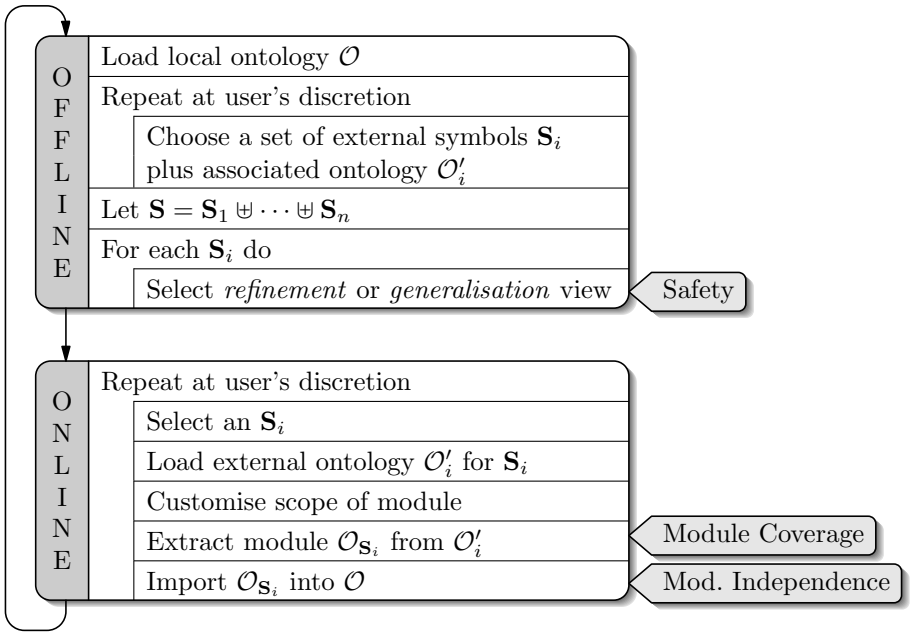

Figure 3. The two phases of import with the required guarantees

In the Online Phase, the relevant knowledge from each external ontology is imported. We aim at extracting only those fragments from the external ontologies that are relevant to the reused symbols, and therefore these fragments should be modules in the sense of Definition 3.

As Figure 3 shows, the import for each external ontology $\mathcal{O}'_i$ consists of four steps. First, $\mathcal{O}'_i$ is loaded—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—the ontology engineer obtains a view of $\mathcal{O}'_i$ and may extend $\mathbf{S}_i$ by giving requirements such as "The module has to contain the concept Joint, all its direct super-concepts and two levels of its sub-concepts". Third, a fragment of $\mathcal{O}'_i$ is extracted. At this stage, we should ensure that this fragment is a module for the customised signature according to Definition 3. Therefore, the following guarantee should be provided for each external ontology and customised signature:

Definition 6 (Module Coverage Guarantee). *The ontology $\mathcal{O}'_{\mathbf{S}}$* guarantees coverage of the signature $\mathbf{S}$ in $\mathcal{O}' \supseteq \mathcal{O}'_{\mathbf{S}}$ *if $\mathcal{O}'_{\mathbf{S}}$ is a module for $\mathbf{S}$ in $\mathcal{O}'$.*

Finally, the actual module $\mathcal{O}_{\mathbf{S}_i}$ is imported, and $\mathcal{O}$ evolves to $\mathcal{O} \cup \mathcal{O}_{\mathbf{S}_i}$. This new ontology might violate safety. Such an effect is obviously undesirable. Hence the following guarantee should be provided:

Definition 7 (Module Independence Guarantee). *Let $\mathcal{O}$ be an ontology and $\mathbf{S}_1, \mathbf{S}_2$ be signatures.* $\mathcal{O}$ guarantees module independence *if, for all $\mathcal{O}_1$ with $\mathsf{Sig}(\mathcal{O}) \cap \mathsf{Sig}(\mathcal{O}_1) \subseteq \mathbf{S}_1$ and for all $\mathcal{O}_2$ with $\mathsf{Sig}(\mathcal{O}) \cap \mathsf{Sig}(\mathcal{O}_2) \subseteq \mathbf{S}_2$ and $\mathsf{Sig}(\mathcal{O}_1) \cap \mathsf{Sig}(\mathcal{O}_2) = \emptyset$, it holds that $\mathcal{O} \cup \mathcal{O}_1 \cup \mathcal{O}_2$ is a conservative extension of $\mathcal{O} \cup \mathcal{O}_1$.*

Note that, if we dropped the requirement $\mathsf{Sig}(\mathcal{O}_1) \cap \mathsf{Sig}(\mathcal{O}_2) = \emptyset$, then module independence would hold if, for all $\mathcal{O}_1$ as above, $\mathcal{O} \cup \mathcal{O}_1$ were safe for $\mathbf{S}_2$. However, this would be a meaningless definition because no $\mathcal{O}$ would guarantee module independence for any $\mathbf{S}_2$ with more than one concept name. In practice, it is safe to assume that different reference ontologies usually have different namespaces.

In order to provide the necessary guarantees of our methodology, we will now make use of the locality conditions introduced in Section 2.2 and some general properties of conservative extensions, safety and modules.

The Safety Guarantee. In Section 2.2, we argue that the simultaneous refinement and generalisation of an external concept may violate safety. To preserve safety, we propose to use two locality conditions: $\bot$-locality, suitable for refinement, and $\top$-locality, suitable for generalisation. These conditions can be checked syntactically using the grammars defined in Figure 2, and therefore they can be easily implemented. In order to achieve the safety guarantee at the end of the offline phase, we propose to follow the procedure sketched in Figure 4.

The following holds upon completion of the procedure: for each $\mathbf{S}_i$, if the user selected the refinement (generalisation) view, then $\mathcal{O}$ is $\bot$-local ($\top$-local) w.r.t. $\mathbf{S}_i$, which is sufficient to guarantee safety:

Proposition 8. *Let $\mathcal{O}$ be an ontology and $\mathbf{S} = \mathbf{S}_1 \uplus \ldots \uplus \mathbf{S}_n$ be the union of disjoint signatures. If, for each $\mathbf{S}_i$, either $\mathcal{O}$ is $\bot$-local or $\top$-local w.r.t. $\mathbf{S}_i$, then $\mathcal{O}$ guarantees safety w.r.t. $\mathbf{S}_1, \ldots, \mathbf{S}_n$.*

The Module Coverage Guarantee. The fragment extracted for each customised signature in the online phase must satisfy the module coverage guarantee. Given an external ontology $\mathcal{O}'$ and customised signature $\mathbf{S}_i$, the scope

Input Ontology $\mathcal{O}$, disjoint signatures $\mathbf{S}_1, \ldots, \mathbf{S}_n$
a choice among refinement and generalisation for each $\mathbf{S}_i$

Output an ontology $\mathcal{O}_1$ that guarantees safety

1: $\mathcal{O}_1 := \mathcal{O}$
2: **while** exists $\mathbf{S}_i$ such that $\mathcal{O}$ **not** local according to the selection for $\mathbf{S}_i$ **do**
3: check $\begin{cases} \bot\text{-locality of } \mathcal{O}_1 \text{ w.r.t. } \mathbf{S}_i \text{ if } \mathbf{S}_i \text{ is to be refined} \\ \top\text{-locality of } \mathcal{O}_1 \text{ w.r.t. } \mathbf{S}_i \text{ if } \mathbf{S}_i \text{ is to be generalised} \end{cases}$
4: **if** non-local **then**
5: $\mathcal{O}_1 :=$ repair $\mathcal{O}_1$ until it is local for $\mathbf{S}_i$ according to the choice for $\mathbf{S}_i$
6: **end if**
7: **end while**
8: **return** $\mathcal{O}_1$

Figure 4. A procedure for checking safety

of the $\bot$-module and $\top$-module is determined: the $\bot$-module contains all the super-concepts in $\mathcal{O}'$ of the concepts in $\mathbf{S}_i$, whereas the $\top$-module contains the sub-concepts. The extraction of these modules, however, may introduce unnecessary symbols not in $\mathbf{S}_i$. To reduce the size of the modules, we proceed as follows: first, extract the minimal $\bot$-module $\mathcal{O}'_1$ for $\mathbf{S}$ in $\mathcal{O}'$; then, extract the minimal $\top$-module $\mathcal{O}'_2$ for $\mathbf{S}$ in $\mathcal{O}'_1$. Now, $\mathcal{O}'_2$ satisfies the module coverage guarantee:

Proposition 9. *Let* $\mathcal{O}'_1 = \mathsf{UpMod}(\mathcal{O}', \mathbf{S})$*, and* $\mathcal{O}'_2 = \mathsf{LoMod}(\mathcal{O}'_1, \mathbf{S})$*. Then* $\mathcal{O}'_2$ *guarantees coverage of* $\mathbf{S}$ *in* $\mathcal{O}'$*.*

The Module Independence Guarantee. When a module is imported in the online phase (see Figure 3), module independence should be guaranteed—that is, after importing a module for a signature $\mathbf{S}_i$ from an external ontology $\mathcal{O}'_i$ into $\mathcal{O}$, the extended ontology $\mathcal{O}$ should still satisfy safety.

Proposition 10. *Given* $\mathcal{O}$ *and disjoint signatures* $\mathbf{S}_1, \mathbf{S}_2$*, if, for each* $i = 1, 2$*,* $\mathcal{O}$ *is* $\bot$*-local or* $\top$*-local w.r.t.* $\mathbf{S}_i$*, then* $\mathcal{O}$ *guarantees module independence.*

4 The Ontology Reuse Tool

We have developed a Protégé 4[10] plugin that supports the methodology presented in Section 3. The plugin and user manual can be downloaded from `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 associated external ontology URI. This information is stored in the ontology using OWL 1.1 ontology/entity annotation axioms [5]. The set of external entities with the same external ontology URI can be viewed as one of the $\mathbf{S}_i$.

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

ProSÉ allows for specifying refinement or generalisation for each external ontology. The tool then provides safety checking of the ontology w.r.t. each group of external symbols separately. This check uses $\bot$-locality/$\top$-locality for refinement/generalisation. The non-local axioms are displayed for a possible repair.

Figure 5 shows the *ProSÉ* tab with the signature subgroups in the top left corner, and the non-local axioms in the bottom left corner. In this phase, the user may work completely offline, without the need of extracting and importing external knowledge, and even without knowing exactly from which ontology the reused entities will be taken. The specification of the URIs of the external ontologies is optional at this stage. Even if a URI is given, it may not refer to a real ontologies—it may simply act as a temporary name.

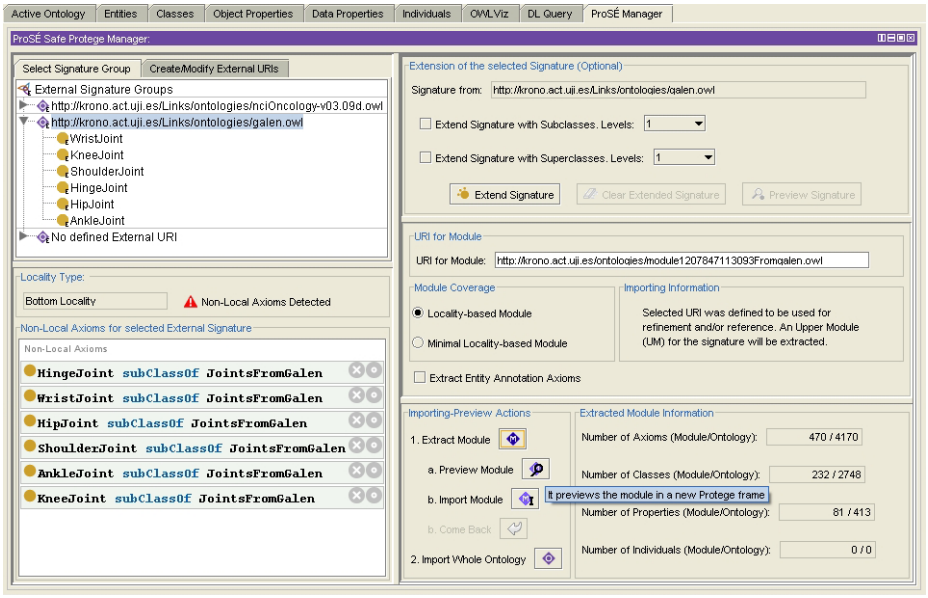

Figure 5. ProSÉ—a Protégé-4 *P*lugin for *R*eusing *O*ntologies: *S*afe and *É*conomique

In the *online phase*, the user chooses external ontologies and will import axioms from them. At this stage, the groups $\mathbf{S}_i$ of external symbols to be imported should refer to the location of a "real" external ontology. Once an $\mathbf{S}_i$ has been selected, it can be customised by adding super-concepts and sub-concepts of the selected symbols. The tool allows for previewing the extended $\mathbf{S}_i$. Next, a module for the extended $\mathbf{S}_i$ can be extracted, which is computed using the procedure in Proposition 9. The user can compute the module, preview it in a separate frame, and either import it or cancel the process and come back to the signature customisation stage. It is also possible to import the whole external ontology instead of a module. Currently, modules are imported "by value"—they will become independent from the original ontology. If the external ontology on the

Web evolves, this will not affect the module. The right hand side of the ProSÉ tab reflects the workflow of the online phase.

So far, we have demonstrated our tool to various ontology developers[11], who have expressed great interest. We are currently working on a proper user study. In `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 reasonably sized compared to the whole ontology.

5 Related Work

Ontology Engineering Methodologies. Several ontology engineering methodologies can be found in the literature, e.g., 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.

Ontology Segmentation and Ontology Integration Techniques. Recently, a growing body of work has been developed addressing ontology modularisation, mapping and alignment, merging, integration and segmentation, see [9, 10, 11] for surveys. This diverse field originates from different communities. In particular, numerous techniques for extracting fragments of ontologies are known. Most of them, such as [12, 13, 14], rely on syntactic heuristics for detecting relevant axioms. These techniques do not attempt to formally specify the intended outputs and do not provide any guarantees.

Ontology Reuse techniques. There are various proposals for "safely" combining modules; most of them, 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, we assume that reuse is performed by simply building the logical union of the axioms in the modules under the standard semantics. We provide the user with a collection of reasoning services, such as safety testing, to check for side-effects. Our paper is based on theoretical work [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 symbols to reuse: it will allow to browse an ontology for symbols 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

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

"by value" mechanism: modules are extracted at the user's request. In addition, we would like to support import "by reference": a feature that checks whether the imported ontology has changed and thus a new import is necessary.

Acknowledgements. This work has been 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] 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
[2] 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
[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)