Modularity and Web Ontologies

Bernardo Cuenca Grau * Information Management Group School of Computer Science University of Manchester, UK bcg@cs.man.ac.uk

Bijan Parsia and **Evren Sirin** and **Aditya Kalyanpur** Maryland Information and Network Dynamics Lab. 8400 Baltimore Av. College Park, MD, 20740 USA bparsia@isr.umd.edu, {evren,aditya}@cs.umd.edu

Abstract

Modularity in ontologies is key both for large scale ontology development and for distributed ontology reuse on the Web. However, the problems of formally characterizing a modular representation, on the one hand, and of automatically identifying modules within an OWL ontology, on the other, has not been satisfactorily addressed, although their relevance has been widely accepted by the Ontology Engineering and Semantic Web communities.

In this paper, we provide a notion of modularity grounded on the semantics of OWL-DL. We present an algorithm for automatically identifying and extracting modules from OWL-DL ontologies, an implementation and some promising empirical results on real-world ontologies.

Introduction and Motivation

In Ontology Engineering, as in Software Engineering, modularity is a much praised virtue. Modular representations (or programs) are easier to understand, verify, debug, extend, reuse parts of, and thus facilitate collaborative development. For Web ontologies, where the collaboration is, in large part, uncoordinated, it is often not enough that the ontology be modular in a general sense, but that, for a large ontology, there are extractable parts that can be reused outside the context of the original ontology. Furthermore, there is the expectation that those fragments are not *arbitrary*, but maintain some relation to the meaning of those parts in the original context. Indeed, if the fragments are "modules", one would expect that their extraction preserves key aspects of their embedded meaning.

However, the problems of *formally* characterizing a modular representation, on the one hand, and of automatically identifying modules within an ontology, on the other, have not been satisfactorily addressed in the Ontology Engineering and Semantic Web literature, although their relevance has been widely accepted by those communities.

Basic to a clear notion of modular decomposition of a logical theory (such as an ontology) is an account of the the *correctness* of that decomposition. In (Garson 1989), James Garson proposed a criterion of validity for fragments of a logical theory. A fragment $\mathcal{T}'$ of a theory $\mathcal{T}$ is a *logical module* just if, for some background logic:

- It is *locally correct*, i.e. any sentence provable in $\mathcal{T}'$ should be provable in $\mathcal{T}$.
- It is *locally complete*, i.e. every sentence in the signature of $\mathcal{T}'$ that is provable in $\mathcal{T}$ should be provable in $\mathcal{T}'$.

The intuition is simple: modular fragments of a theory should entail all and only the entailments regarding its "subject matter" that the original theory entailed. When a logical module is extracted from its original context, no consequences in the signature of the module are lost and no new consequences are obtained. Thus, from a model-theoretic perspective, logical modules are *self-contained* units within an ontology that can be safely extracted without adding or removing entailments in the signature of other modules.

Local correctness is a direct consequence of the *monotonicity* of a logic and it is a trivial property to show for the ontology languages we are concerned with. Local completeness is a strengthening of *uniform interpolation* (Pitts 1992) (Wolter 1998) in that the interpolant $\mathcal{T}'$ is required to be a subset of the parent theory $\mathcal{T}$. Contrary to local correctness, the notion of local completeness poses two major difficulties:

1. Given that, in FOL as well as in Description Logics, contradictions entail everything, every consistent fragment of an inconsistent ontology will fail to be locally complete. Garson uses this fact, plus the difficulty of determining the consistency of a large evolving FOL theory, to argue that FOL is not a proper logic for modular KR.
2. Very few logics are known to have uniform interpolation and it is unlikely that the expressive Description Logics underlying OWL-Lite and OWL-DL do. In particular, the solution, as well as the theoretical solvability, of the following problems remains an open question for the logics underlying OWL-Lite and OWL-DL:
   (a) Given a fragment $\mathcal{T}'$ of an ontology $\mathcal{T}$, is $\mathcal{T}'$ a uniform interpolant of $\mathcal{T}$?
   (b) Given an ontology $\mathcal{T}$ and a signature $\mathbf{S} \subseteq \mathsf{Sig}(\mathcal{T})$, is there a uniform interpolant $\mathcal{T}'$ of $\mathcal{T}$ such that $\mathsf{Sig}(\mathcal{T}') = \mathbf{S}$?

In this paper, we address these issues as follows: first, in a Description Logic setting, where there is a decision pro-

*This author is is supported by the EU Project TONES (Thinking ONtologiES) ref: IST-007603.

cedure for consistency checking that is practical for realistic KBs (Horrocks & Sattler 2005)(Horrocks, Sattler, & Tobies 2000), it is reasonable to demand that an ontology be consistent; second, although we do not provide a general solution for 2a) and 2b), we will be able to ensure that the modules obtained using our algorithms are indeed uniform interpolants of the parent ontology.

Unlike (Garson 1989) and (MacCartney *et al.* 2003), we are concerned with *reusing* parts of ontologies, not just for improving reasoning performance, but also for the sake of intelligibility for humans and effective reuse. Thus, we aim at fulfilling the following additional desiderata:

- Modules should be easily *reusable* for applications.
- Modules should be *intelligible*, that is, they should make sense to ontology engineers seeking to (re)use them.

In particular, in Semantic Web applications, reuse often boils down to the following task: given a concept name in the ontology that we want to "borrow", provide the axioms in the ontology that are "relevant" to its meaning.

Enforcing modules to be logical, in Garson's sense, is not sufficient for addressing these additional requirements.

In order to provide effective reuse, our definition of a module will be relative to a concept name in the signature of the ontology, such that each name will be assigned a (single) module. Reusing a concept then boils down to retrieving its corresponding module within the ontology. In order to improve intelligibility, our goal will be to obtain modules that deal with a well-defined subject matter within the ontology.

Thus, in this paper, we aim at *formalizing* and *solving* the following problem:

- Given a concept name A and an ontology $\mathcal{T}$, retrieve a fragment $\mathcal{T}'_A \subseteq \mathcal{T}$ such that:
  1. it is a logical module of $\mathcal{T}$.
  2. it captures the meaning of A in $\mathcal{T}$ in a sensible, well-understood way.
  3. it represents a well-defined subject matter.

Formalizing **1)** is straightforward. The formalization of **2)** and **3)** is indeed more controversial.

In this paper, we argue that a notion of modularity that meets the requirements listed in this section is indeed achievable and propose a formal definition of module as well as an algorithm for quickly identifying and retrieving modules within an OWL-DL ontology. We investigate which OWL-DL ontologies can be "safely" modularized according to our notion of modularity; in particular, we enforce the ontologies to:

- be consistent.
- contain no unsatisfiable concepts.
- contain no "dangerous" General Concept Inclusion Axioms (GCIs).

Ontology inconsistency and concept unsatisfiability can be effectively determined using a DL reasoner, such as RACER, FaCT++ or Pellet, and are considered to be serious semantic defects that significantly corrupt the intended semantics of the ontology. We understand that these defects need to be resolved and assume that they have been fixed prior to the modularization process. A more controversial issue is how to characterize "dangerous" GCIs. Intuitively, GCIs may impose semantic constraints on the ontology *as a whole*; extracting a fragment from its context in the presence of these GCIs may yield to unexpected consequences. In this paper, we provide a formal notion of *safe* ontology. For safe ontologies, we guarantee local correctness and completeness for the retrieved modules and show that modules can be identified in polynomial time without any user intervention. We describe an implementation of our modularization algorithm based on Manchester's OWL-API (Bechhofer, Lord, & R.Volz 2003), and the open source ontology editor Swoop (Kalyanpur *et al.* 2005), as well as some empirical results on real-world ontologies. Finally, we provide an insight on how to interpret the retrieved modules from a modeling perspective.

Preliminaries

In this Section, we introduce the logic $\mathcal{SHOIQ}$ (Horrocks & Sattler 2005) and the notions of uniform interpolation and logical module in the context of DLs.

Before going into formal details, it is worth mentioning that OWL-DL is a notational variant of the Description Logic $\mathcal{SHOIN}(D)$ (Horrocks, Patel-Schneider, & van Harmelen 2003). Our results apply to the logic $\mathcal{SHOIQ}$ instead, which presents some subtle differences with respect to OWL-DL:

1. $\mathcal{SHOIQ}$ generalizes the cardinality restrictions in OWL-DL to *qualified* cardinality restrictions.
2. OWL-DL provides support for *datatypes*.

For ease of presentation, we have decided not to consider datatypes in this paper. Our results, however, can be easily extended, and datatypes are indeed supported in our implementation.

The Description Logic $\mathcal{SHOIQ}$

Let $\mathbf{C}, \mathbf{R}$ be countably infinite and pair-wise disjoint sets of *concept* and *role names* and let $\mathbf{I} \subseteq \mathbf{C}$ be a set of *nominals*. We will denote concept and role names with capital letters A, B and R, S respectively; nominals will be denoted with lowercase letters a, b, c.

The set of $\mathcal{SHOIQ}$-roles (roles, for short) is the set $\mathbf{R} \cup \{\mathsf{Inv}(R) | R \in \mathbf{R}\}$, where $\mathsf{Inv}(R)$ denotes the inverse of a role R. Concepts are inductively using the following grammar:

$$C \leftarrow A | \neg C | C_1 \sqcap C_2 | \exists R.C | \geq nS.C$$

where A ranges over concept names (including nominals), $C_{(i)}$ over concepts, R over roles, S over *simple* roles [1], and n over positive integers. We use the following abbreviations: $C \sqcup D$ stands for $\neg(\neg C \sqcap \neg D)$; $\top$ and $\bot$ stand for $A \sqcup \neg A$ and $A \sqcap \neg A$ respectively; finally, we use $\forall R.C$ and $\leq nS.C$ as a shorthand for $\exists R.\neg C$ and $\neg(\geq n+1 S.C)$ respectively.

[1] See (Horrocks & Sattler 2005) for a precise definition of simple roles.

A *role inclusion axiom* is an expression of the form $R_1 \sqsubseteq R_2$, where R_1, R_2 are roles. A *transitivity axiom* is an expression of the form $\mathsf{Trans}(R)$, where $R \in \mathbf{R}$. For C, D concepts, a *general concept inclusion axiom* (GCI) is an expression of the form $C \sqsubseteq D$. A TBox $\mathcal{T}$ is a finite set of concept inclusion axioms, role inclusion axioms and transitivity axioms.

An *interpretation* $\mathcal{I}$ is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, called the *domain* of the interpretation, and $.^{\mathcal{I}}$ is the *interpretation function*. The interpretation function assigns to each $A \in \mathbf{C}$ a subset of $\Delta^{\mathcal{I}}$ and to each $R \in \mathbf{R}$ a subset of of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. For a nominal a, the set $a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ is additionally required to be a singleton. The interpretation function extends to complex concept as follows:

$$\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \\
(\geq nR.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \\
&\quad \#(\{y \in \Delta^{\mathcal{I}} \mid \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}) \geq n\}
\end{aligned}$$

The satisfaction of a $\mathcal{SHOIQ}$ axiom α in an interpretation $\mathcal{I}$, denoted $\mathcal{I} \models \alpha$, is defined as follows: (1) $\mathcal{I} \models R_1 \sqsubseteq R_2$ iff $(R_1)^{\mathcal{I}} \subseteq (R_2)^{\mathcal{I}}$; (2) $\mathcal{I} \models \mathsf{Trans}(R)$ iff for every $x, y, z \in \Delta^{\mathcal{I}}$, if $\langle x, y\rangle \in R^{\mathcal{I}}$ and $\langle y, z\rangle \in R^{\mathcal{I}}$, then $\langle x, z\rangle \in R^{\mathcal{I}}$; (3) $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. The interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ if it satisfies all the axioms in $\mathcal{T}$.

A concept C is *satisfiable relative to* $\mathcal{T}$ if there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$. We say that *C subsumes D relative to* $\mathcal{T}$ if, for every model $\mathcal{I}$ of $\mathcal{T}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Logical Modules and Uniform Interpolation

We now introduce uniform interpolation for TBoxes. A signature $\mathbf{S} \subseteq \mathbf{C} \cup \mathbf{R}$ is a finite set of concept and role names. The signature $\mathsf{Sig}(\alpha)$ (respectively $\mathsf{Sig}(\mathcal{T})$) of an axiom α (respectively of a TBox $\mathcal{T}$) is the set of concept and role names occurring in it. Given a signature $\mathbf{S}$, we use $\mathsf{Con}(\mathbf{S})$ and $\mathsf{Rol}(\mathbf{S})$ to denote respectively the set of $\mathcal{SHOIQ}$-concepts and roles that can be built using only symbols in $\mathbf{S}$.

Definition 1 ***(Uniform interpolation for TBoxes)***
A TBox $\mathcal{T}'$ is a uniform interpolant of a consistent TBox $\mathcal{T}$ if the following conditions hold:

- $\mathcal{T} \models \mathcal{T}'$.
- *For every axiom α such that $\mathsf{Sig}(\alpha) \subseteq \mathsf{Sig}(\mathcal{T}')$, if $\mathcal{T} \models \alpha$, then $\mathcal{T}' \models \alpha$.*

Garson's notion of a logical module can be defined by using uniform interpolation to formalize local completeness and by requiring, additionally, the interpolant $\mathcal{T}'$ to be a subset of the parent ontology $\mathcal{T}$.

Definition 2 ***(Logical Module)***
Let $\mathcal{T}$ be consistent. A TBox $\mathcal{T}' \subseteq \mathcal{T}$ is a ***logical module*** *of $\mathcal{T}$ if, for every axiom α such that $\mathsf{Sig}(\alpha) \subseteq \mathsf{Sig}(\mathcal{T}')$:*

$$\mathcal{T} \models \alpha \text{ iff } \mathcal{T}' \models \alpha$$

Note that, as mentioned before, we will always require *consistency* of the ontologies to be modularized.

The Notion of a Module

In this Section, we formalize the notion of a *module*, $\mathcal{T}'_A$, for a concept name A in the context of a $\mathcal{SHOIQ}$ ontology $\mathcal{T}$.

As a first requirement, we will enforce $\mathcal{T}'_A$ to be a logical module of $\mathcal{T}$.

As argued before, logical modules represent, from a model-theoretic perspective, self-contained units within the ontology. However, Garson's notion of local correctness and completeness is not sufficient to address all our requirements, since:

- it does not determine the *scope* of the module (i.e. which symbols should be included in its signature) and, consequently, its *size*.
- it does not provide an insight on how to interpret the module from a *modeling* perspective.

In order to address these issues, our definition specifies a class of logical consequences of the input ontology to be preserved in the extracted module. The goal is to:

1. force the relevant knowledge about the concept to be included in its module.
2. make sure that the module represents a well-defined subject matter and, consequently, that is self-contained from a modeling perspective.

This class of entailments determines the *scope* of the module and thus the axioms of $\mathcal{T}$ that must be included in $\mathcal{T}'_A$.

Including the Relevant Information About the Concept
In traditional DL settings, not all entailments are equally valued. Indeed, there is a set of standard inference services which DL-focused systems expose and emphasize, namely:

1. *satisfiability of concept names* determines whether a concept name A in the KB is satisfiable, i.e. if there is a model $\mathcal{I}$ of the KB for which $A^{\mathcal{I}} \neq \emptyset$.
2. *classification* computes the subsumption partial ordering of all the concept names in the KB.
3. *instantiation and retrieval* determine whether an individual is an instance of a concept name and retrieve all the instances of an atomic concept respectively. In $\mathcal{SHOIQ}$, instantiation and retrieval can be seen as a particular case of concept subsumption and classification, since individuals are represented by nominal concepts.

For ontology engineers, it is especially important to ensure that a module extracted from an OWL ontology for reuse or maintenance purposes preserves the results of these reasoning tasks. In other words, if we are to reuse a concept name A and retrieve a fragment $\mathcal{T}'_A$ of the original ontology $\mathcal{T}$, we want to make sure that A, as well as all its sub-concepts, super-concepts and instances are included in $\mathcal{T}'_A$. We argue that such a fragment reasonably captures the meaning of A in $\mathcal{T}$.

Ensuring Self-Containment from a Modeling Perspective Ontologies typically contain knowledge about different subject matters. An example is the ontology used in the OWL documentation: the Wine Ontology (Smith, Welty, & McGuiness 2004). This ontology describes different kinds of wines according to various criteria, like the area they are produced in, the kinds of grapes they contain, their flavor and color, etc. Thus, the Wine Ontology does not contain information about wines only, but also information about regions, wineries, colors, grapes, and so on. This illustrates a common pattern in OWL ontologies: although ontologies usually refer to a core application domain, they also contain "side" information about other secondary domains. These domains, although related, are mostly self-contained in the sense that they only deal with a single "topic".

This modeling paradigm is not only characteristic of small and medium sized ontologies, but also occurs in large, high-quality knowledge bases, written by groups of experts. A prominent example is the NCI (National Cancer Institute) ontology (Golbeck *et al.* 2003), a huge, highly structured ontology dealing with the biomedical domain. NCI is a reference terminology covering areas of basic and clinical science, built with the goal of facilitating translational research in Cancer. The NCI ontology is mainly focused on genes, but it also contains some information about many other subject matters, like professional organizations, funding, research programs, etc.

In order to ensure that our modules are coherent and self-contained, we require that no subsumption relations exist between concepts *inside* the module (i.e., contained in its signature) and concepts *outside* the module. Such a condition enforces a logical separation between the module and its context.

The intuitions described in this Section yield to the following notion of module:

Definition 3 ***(Module)***
A TBox $\mathcal{T}'_A \subseteq \mathcal{T}$ is a ***module*** *for a concept name $A \in \mathsf{Sig}(\mathcal{T})$ if:*

1. *$\mathcal{T}'_A$ is a logical module in $\mathcal{T}$.*
2. *for every concept $B \in \mathsf{Sig}(\mathcal{T})$, the following holds:*
   (a) $\mathcal{T}'_A \models (A \sqsubseteq B) \Leftrightarrow \mathcal{T} \models (A \sqsubseteq B)$.
   (b) $\mathcal{T}'_A \models (B \sqsubseteq A) \Leftrightarrow \mathcal{T} \models (B \sqsubseteq A)$.
3. *There are no concept names $D, E \in \mathsf{Sig}(\mathcal{T})$ such that $D \in \mathsf{Sig}(\mathcal{T}'_A), E \notin \mathsf{Sig}(\mathcal{T}'_A)$ and either $\mathcal{T} \models D \sqsubseteq E$, or $\mathcal{T} \models E \sqsubseteq D$.*

We argue that this formal notion of a module satisfies our requirements and, hence, it makes perfect sense, both from a logical and a modeling perspective, to retrieve $\mathcal{T}'_A$ instead of $\mathcal{T}$ whenever we need to reuse A.

Safe OWL-DL Ontologies

Given our notion of a module, we show that there is a class of "safe" OWL-DL theories that *can* be modularized. In this Section, we investigate, both from a logical and a modeling perspective, when an OWL-DL ontology can be considered to be safe. In order to understand the potential effect of "dangerous" GCIs, let us consider the the following simple ontology, which is *not* safe:

$$\mathcal{T} = \{\top \sqsubseteq bob; bob \sqsubseteq Person; bob \sqsubseteq \exists Drives.Car; Car \sqsubseteq Vehicle\}$$

with bob being a nominal. In the absence of the first axiom, the TBox $\mathcal{T}_{Car} = \{Car \sqsubseteq Vehicle\}$ is a module for the concept name Car in $\mathcal{T}$, according to Definition 3. However, in the presence of the GCI $\top \sqsubseteq bob$, our definition of module is violated, since $\mathcal{T} \models bob \sqsubseteq Car$, but $\mathcal{T}_{Car} \not\models bob \sqsubseteq Car$. The problem, in this case, is caused by the ability of GCIs to fix the size of the interpretation domain in every model of the ontology. The reader should note that merely including the problematic GCI in $\mathcal{T}_{Car}$ does not help, since $\mathcal{T} \models Car \sqsubseteq Person$, but $\mathcal{T}_{Car} \cup \{\top \sqsubseteq bob\} \not\models Car \sqsubseteq Person$. In fact, it is not hard to see that the only module for Car in $\mathcal{T}$ is precisely $\mathcal{T}$.

In order to assess the "globality" of a GCI, we introduce the notion of a *domain expansion*.

Definition 4 ***(Domain Expansion)***
Let $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ and $\mathcal{J} = (\Delta^{\mathcal{J}}, .^{\mathcal{J}})$ be interpretations such that: ***1)*** *$\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}} \cup \Phi$, with Φ a non-empty set disjoint with $\Delta^{\mathcal{I}}$;* ***2)*** *$A^{\mathcal{J}} = A^{\mathcal{I}}$ for each concept name;* ***3)*** *$R^{\mathcal{J}} = R^{\mathcal{I}}$ for each role name.*
We say that $\mathcal{J}$ is the ***expansion*** *of $\mathcal{I}$ with Φ.*

Intuitively, the interpretation $\mathcal{J}$ is identical to $\mathcal{I}$ except for the fact that it contains some additional elements in the interpretation domain. These elements do not participate in the interpretation of concepts or roles. The following question naturally arises: if $\mathcal{I}$ is a model of $\mathcal{T}$, is $\mathcal{J}$ also a model of $\mathcal{T}$? Safe ontologies are precisely those whose models are closed under domain expansions.

Definition 5 ***(Safety)***
Let $\mathcal{T}$ be consistent. We say that $\mathcal{T}$ is ***safe*** *if, for every $\mathcal{I} \models \mathcal{T}$ and every set Φ disjoint with $\Delta^{\mathcal{I}}$, the expansion $\mathcal{J}$ of $\mathcal{I}$ with Φ is a model of $\mathcal{T}$.*

Examples of unsafe axioms are GCIs that:

- fix the size of the domain in every model of the ontology, e.g. $\top \sqsubseteq bob$.
- establish the existence of a "universal" named concept, i.e., one that is equivalent to $\top$. For example, $\top \sqsubseteq Car$.

Examples of safe GCIs are role domain and range and concept disjointness.

The Modularization Algorithm

In this section, we present an algorithm that, given an input ontology $\mathcal{T}$ and a concept A, retrieves a module for A in $\mathcal{T}$.

The main idea of the algorithm is to generate a *partitioning* of the input ontology $\mathcal{T}$, represented as a directed labeled graph (the *partitioning graph*) and then use the graph to find the module for each concept in $\mathcal{T}$.

The algorithm consists of three main steps: a *safety check*, the generation of a partitioning graph G and the identification and extraction of modules from G.

The Safety Check

In this Section, we show how to detect the presence of "unsafe" GCIs. We start by introducing the notion of *locality* of a concept:

Definition 6 ***(Locality)***
*A concept C is **local** if, for every interpretation $\mathcal{I}$ for C and every non-empty set Φ disjoint with $\Delta^{\mathcal{I}}$, the expansion $\mathcal{J}$ of $\mathcal{I}$ with Φ verifies:*

$$C^{\mathcal{J}} = C^{\mathcal{I}}$$

*Otherwise, we say that C is **non-local**. For $\mathbf{S}$ a signature, we denote by $\mathsf{local}(\mathbf{S})$ the set of local concepts that can be constructed using the symbols in $\mathbf{S}$.*

Thus, local concepts are those whose interpretation remains invariant under domain expansions. The following theorem establishes the syntactic countepart to the notion of locality:

Theorem 1 *Let $\mathbf{S}$ be a signature and C a concept in $\mathsf{Con}(\mathbf{S})$, then:*

- *If C is a concept name, then $C \in \mathsf{local}(\mathbf{S})$.*
- *If C is of the form $\exists R.D$ or $\geq nR.D$, then $C \in \mathsf{local}(\mathbf{S})$.*
- *If C of the form $D \sqcap E$, then $C \in \mathsf{local}(\mathbf{S})$ iff any of D, E is in $\mathsf{local}(\mathbf{S})$.*
- *If C of the form $\neg D$, then, $C \in \mathsf{local}(\mathbf{S})$ iff $D \notin \mathsf{local}(\mathbf{S})$.*

Furthermore, if $C \notin \mathsf{local}(\mathbf{S})$, $C^{\mathcal{J}} = C^{\mathcal{I}} \cup \Phi$ for every possible pair of interpretations $\mathcal{I}, \mathcal{J}$ s.t. $\mathcal{J}$ is an expansion of $\mathcal{I}$ with Φ.

As a consequence of the theorem, the problem of deciding whether $C \in \mathsf{local}(\mathbf{S})$ for some signature $\mathbf{S}$ can be solved in polynomial time w.r.t. the length $|C|$ of the concept C.

Using the theorem above, we can find an effective procedure for deciding safety:

Theorem 2 *Let $\mathcal{T}$ be consistent. Then, $\mathcal{T}$ is unsafe iff it explicitly contains a GCI $C \sqsubseteq D$ such that C is non-local and D is local.*

As a direct consequence of Theorems 1 and 2, the problem of deciding safety of a consistent ontology $\mathcal{T}$ is polynomial w.r.t the size $|\mathcal{T}|$ of $\mathcal{T}$.

The Partitioning Algorithm

In case of a positive result in the safety check, the algorithm generates a partitioning of the input ontology. In general (MacCartney *et al.* 2003), $\{\mathcal{T}_i\}_{1 \leq i \leq n}$ is a *partitioning* of a logical theory $\mathcal{T}$ if $\mathcal{T} = \bigcup_i \mathcal{T}_i$. Each individual $\mathcal{T}_i$ is called a *partition* and contains a distinct subset of the axioms of $\mathcal{T}$.

We represent the partitioning by means of a labeled directed graph $G = (\mathbf{V}, \mathbf{E}, \mathcal{L}, \mathcal{V})$. Each node $v \in \mathbf{V}$ is labeled with a non-empty partition $\mathcal{L}(v) \subseteq \mathcal{T}$. The labels of two different nodes are disjoint ($\mathcal{L}(v_i) \cap \mathcal{L}(v_j) = \emptyset$ for $i \neq j$) and the union of the labels of all the nodes in the graph is precisely $\mathcal{T}$ (i.e. $\bigcup_{v \in \mathbf{V}} \mathcal{L}(v) = \mathcal{T}$).

Each edge $e = \langle v, w \rangle$ is labeled with a non-empty set of roles $\mathcal{L}(e)$ occurring in $\mathcal{T}$. Given an edge $e = \langle v, w \rangle$, we denote its first and second elements v, w by $\mathsf{First}(e)$, $\mathsf{Second}(e)$ respectively, and we use e^- to denote its inverse (i.e. $e^- = \langle w, v \rangle$). We assume that the labels of different edges are disjoint ($\mathcal{L}(e) \cap \mathcal{L}(e') = \emptyset$ for $e \neq e'$).

Given two partitions, their respective signatures may *intersect* and, consequently, we need a mechanism to devise the "home" partition of a concept. We introduce a mapping $\mathcal{V}$ in the graph that assigns to each concept and role occurring in $\mathcal{T}$ a *single* node and edge in G, respectively.

Since each symbol is mapped through $\mathcal{V}$ into a single node or edge, the function $\mathcal{V}$ allows to "disambiguate" the shared symbols. This mapping will reveal key for determining which axioms from the original ontology will be grouped together in the same partition as well as for retrieving the module for each concept from the partitioning graph.

The algorithm performs a succession of *partitioning steps*, as shown in Figure 1. Each step involves *a pair* of nodes in the graph: the node v_0, called the *source* node, which initially contains in its label the input ontology and from which axioms are removed, and a the node v, the *target* node, generated from scratch, to which these are added. Note that the source node is always v_0 and the target node is different is each step.

At the beginning of each partitioning step (see Figure 2), the algorithm selects non-deterministically a symbol X in the signature of $\mathcal{L}(v_0)$ and changes the value of $\mathcal{V}(X)$. In the case of a concept, for example, $\mathcal{V}(X)$ is updated to v, which intuitively means that the concept is "moved" to the

-Algorithm Partition($\mathcal{T}$)
-Input: A $\mathcal{SHOIQ}$ ontology $\mathcal{T}$
-Output: A partitioning graph $G = (\mathbf{V}, \mathbf{E}, \mathcal{L}, \mathcal{V})$

$G \leftarrow (\{v_0\}, \emptyset, \mathcal{L}, \mathcal{V})$, with:
  $\mathcal{L}(v_0) = \mathcal{T}$
  $\mathcal{V}(C) = v_0$ for each concept C in $\mathcal{T}$
  $\mathcal{V}(R) = \langle v_0, v_0 \rangle$ for each role R in $\mathcal{T}$
if $\mathcal{T}$ not safe, **return** G
for each role R occurring in $\mathcal{T}$, $\mathsf{BoundTo}(R) \leftarrow \emptyset$
Repeat
  $G \leftarrow \mathsf{DoPartitioningStep}(G)$
until $\mathcal{L}(v_0) = \emptyset$
$\mathbf{V} \leftarrow \mathbf{V} - \{v_0\}$
return G

Figure 1: Partitioning Algorithm

-Algorithm DoPartitioningStep(G)
-Input: A partitioning graph G
-Output: Updated graph G

Create new node v with $\mathcal{L}(v) = \emptyset$ and do $\mathbf{V} \leftarrow \mathbf{V} \cup \{v\}$
Select non-deterministically a concept X with $\mathcal{V}(X) = v_0$,
  or a role X with $\mathcal{V}(X) = \langle v_0, v_0 \rangle$
if X a concept, **then** $\mathcal{V}(X) \leftarrow v$
if X a role **then** $\mathcal{V}(X) \leftarrow \langle v, v_0 \rangle$
$G \leftarrow \mathsf{moveTerms}(G, v)$
$G \leftarrow \mathsf{moveAxioms}(G, v)$
return (G)

Figure 2: Partitioning Steps

target partition.

This initial change will trigger new ones, according to Figure 3.

-**Algorithm MoveTerms**(G, v)
-**Input:** A partitioning graph $G = (\mathbf{V}, \mathbf{E}, \mathcal{L}, \mathcal{V})$
The target node v in the current partitioning step
-**Output:** A partitioning graph with updated mapping $\mathcal{V}$

Repeat
for all concept C occurring in $\mathcal{T}$ with $\mathcal{V}(C) = v_0$
if any of the following conditions holds:
1)$(C \sqsubseteq D)$ or $(D \sqsubseteq C) \in \mathcal{L}(v_0)$, and $\mathcal{V}(D) = v$
2)$\exists R.C$ or $\geq nR.C \in \mathcal{L}(v_0)$ and $\mathsf{Second}(\mathcal{V}(R)) = v$
3)$(C \sqcap D) \in \mathcal{L}(v_0)$ and $\mathcal{V}(D) = v$
4)C of the form $D \sqcap E$ and $\mathcal{V}(E) = v$ or $\mathcal{V}(D) = v$
5)C of the form $\exists R.D, \geq nR.D$ and $\mathsf{First}(\mathcal{V}(R)) \neq v_0$
6)$(\neg C) \in \mathcal{L}(v_0)$ and $\mathcal{V}(\neg C) = v$
7)$C, E \in \mathsf{BoundTo}(R)$ and $\mathcal{V}(E) = v$
then $\mathcal{V}(C) \leftarrow v$
if 2) has held, **then** $\mathsf{BoundTo}(R) \leftarrow \mathsf{BoundTo}(R) \cup \{C\}$
for all role R with $\mathsf{First}(\mathcal{V}(R)) = v_0$ or $\mathsf{Second}(\mathcal{V}(R)) = v_0$
if $(R \sqsubseteq S)$ or $(S \sqsubseteq R) \in \mathcal{L}(v_0)$ and $\mathcal{V}(R) \neq \mathcal{V}(S)$
then $\mathcal{V}(R) \leftarrow \mathcal{V}(S)$
if D of the form $\exists R.C$, $\geq nR.C$ and $\mathcal{V}(D) = v$
then $\mathsf{First}(\mathcal{V}(R)) \leftarrow v$
if $\exists R.C$ or $\geq nR.C \in \mathcal{L}(v_0)$, $\mathcal{V}(C) = v$
then $\mathsf{Second}(\mathcal{V}(R)) \leftarrow v$
if $\mathcal{V}(R) \neq (\mathcal{V}(\mathsf{Inv}(R)))^-$
then $\mathcal{V}(R) \leftarrow (\mathcal{V}(\mathsf{Inv}(R)))^-$
until no change in $\mathcal{V}$ is triggered
return G

Figure 3: Moving Concepts and Roles

Depending on the final value of the $\mathcal{V}$ function, some of the axioms in $\mathcal{L}(v_0)$ are removed from $\mathcal{L}(v_0)$ and added to $\mathcal{L}(v)$ and the labels of the edges involving the target and the source nodes are updated accordingly.

In Figure 5, we provide the content of the partitioning graph at the end of each partitioning step for an example ontology. The reader should be able to reproduce these results using the the algorithms in Figures 1, 3 and 4.[2]

Significance of the Partitioning Graph It is worth taking a closer look to the partitioning graph generated in Figure 5. The graph contains four partitions $\mathcal{L}(v_1), ..., \mathcal{L}(v_4)$. A quick examination of the axioms they contain reveals that the partitions describe intuitively disjoint subject matters, namely courses, publications, departments and students respectively.

The correspondence of each partition to a well-defined application domain, intuitively disjoint from the rest, is a general property of the partitions generated using our algorithm and can be observed in large, real-world ontologies, such as NCI.

The decomposition obtained for NCI can be obtained in less than 45 seconds using our implementation and is shown

[2]As a remark, the set $\mathsf{BoundTo}(P)$ represents the set of terms that are "forced" to end up in the same partition due to the fact that a role P cannot appear in the label of two different edges.

-**Algorithm MoveAxioms**(G,v)
-**Input:** A partitioning graph G
The target node v in the current partitioning step
-**Output:** An updated partitioning graph G

for each Axiom $\alpha \in \mathcal{L}(v_0)$
if α is of any of the following forms:
1)$C \sqsubseteq D$ and $\mathcal{V}(C) = \mathcal{V}(D) = v$
2)$R \sqsubseteq S$, and $\mathcal{V}(R) = \mathcal{V}(S)$, with $\mathsf{First}(V(R)) \neq v_0$
then $\mathcal{L}(v_0) \leftarrow \mathcal{L}(v_0) - \{\alpha\}$ **and** $\mathcal{L}(v) \leftarrow \mathcal{L}(v) \cup \{\alpha\}$
for each R with $\mathcal{V}(R) = \langle v_0, v \rangle$,
do $\mathcal{L}(\langle v_0, v \rangle) \leftarrow \mathcal{L}(\langle v_0, v \rangle) \cup \{R\}$
for each $R \in \mathcal{L}(\langle v_j, v_0 \rangle)$ with $v_j \neq v$
if $\exists C \in \mathsf{BoundTo}(R)$ with $\mathcal{V}(C) = v$ **then**
$\mathcal{L}(\langle v_j, v_0 \rangle) \leftarrow \mathcal{L}(\langle v_j, v_0 \rangle) - \{R\}$
if $\mathcal{L}(\langle v_j, v_0 \rangle) = \emptyset$ **then** $\mathbf{E} \leftarrow \mathbf{E} - \langle v_j, v_0 \rangle$
if $\langle v_j, v \rangle \notin \mathbf{E}$ **then**
$\mathbf{E} \leftarrow \mathbf{E} \cup \langle v_j, v \rangle$
$\mathcal{L}(\langle v_j, v \rangle) \leftarrow \mathcal{L}(\langle v_j, v \rangle) \cup \{R\}$
return G

Figure 4: Moving Axioms

on the left hand side of Figure 6. The figure uses the graph layout in the ontology editor Swoop for visualizing partitioning graphs. In such a layout, the size of the nodes is proportional to the size of the partitions. Isolated nodes are represented in white, leaf nodes in gray and nodes with outgoing edges in black.

The partitions of NCI represent a well-defined subdomain within the ontology. For example, the knowledge about genes, drugs, medical techniques, etc. are each associated to a different partition. These domains are pair-wise disjoint in the sense that they do not share objects (a drug is not a gene and vice-versa). The connections suggest which domains within the ontology are most relevant. For example, highly interconnected partitions, such as the ones dealing with genes and diseases, are central to the ontology. Other nodes, like the one dealing with anatomical structures, are leaves in the graph, and hence represent "secondary" subject matters.

The following theorem justifies why this fact is generally observed:

Theorem 3 *Let $\mathcal{T}$ be safe and $G = \mathsf{Partition}(\mathcal{T})$ with $G = (\mathbf{V}, \mathbf{E}, \mathcal{L}, \mathcal{V})$ and $|\mathbf{V}| = n$, then there exists a model $\mathcal{J} = (\Delta^{\mathcal{J}}.^{\mathcal{J}})$ of $\mathcal{T}$ such that:*

- $\Delta^{\mathcal{J}} = \bigcup_{i=1,...,n} \Delta_i^{\mathcal{J}}$ *with* $\Delta_i^{\mathcal{J}} \cap \Delta_k^{\mathcal{J}} = \emptyset$ *for* $i \neq k$, *and* $\Delta_i^{\mathcal{J}} \neq \emptyset$.
- $A^{\mathcal{J}} \subseteq \Delta_i^{\mathcal{J}}$, *for each concept name* $A \in \mathsf{Sig}(\mathcal{T})$ *such that* $\mathcal{V}(A) = v_i$.
- $R^{\mathcal{J}} \subseteq \Delta_i^{\mathcal{J}} \times \Delta_j^{\mathcal{J}}$, *for each role name* $R \in \mathsf{Sig}(\mathcal{T})$ *such that* $\mathcal{V}(R) = \langle v_i, v_j \rangle$.

The theorem establishes the existence of a very special family of models for $\mathcal{T}$. These models evaluate each partition in a different *logical* sub-domain, disjoint from the rest. We argue that there exists a very close correspondence between the ability to distinguish disjoint logical sub-domains

$\mathcal{T} = \{$	$\mathbf{V} = \{v_1, v_2, v_3, v_4\}$
$St \sqsubseteq \exists enrolledIn.Co$	$\mathbf{E} = \{\langle v_4, v_1\rangle, \langle v_4, v_3\rangle, \langle v_3, v_4\rangle\}$
$St \sqsubseteq Person$	$\mathcal{L}(v_1) = \{\ Co \sqsubseteq \top\ \}$; $\mathcal{L}(v_2) = \{\ Paper \sqsubseteq Pub\}$
$Prof \sqsubseteq \exists teaches.Co \sqcap \exists memberOf.Dept$	$\mathcal{L}(v_3) = \{Dept. \sqsubseteq \exists memberOf^{-}.St\}$;
$Paper \sqsubseteq Pub$	$\mathcal{L}(v_4) = \{\ St \sqsubseteq \exists enrolledIn.Co; St \sqsubseteq Person;$
$Dept. \sqsubseteq \exists memberOf^{-}.St$	$Prof \sqsubseteq \exists teaches.Co \sqcap \exists memberOf.Dept$
$\exists enrolledIn.\top \sqsubseteq Person\}$	$\exists enrolledIn.\top \sqsubseteq Person\}$

Figure 5: A Decomposition into a Partitioning Graph

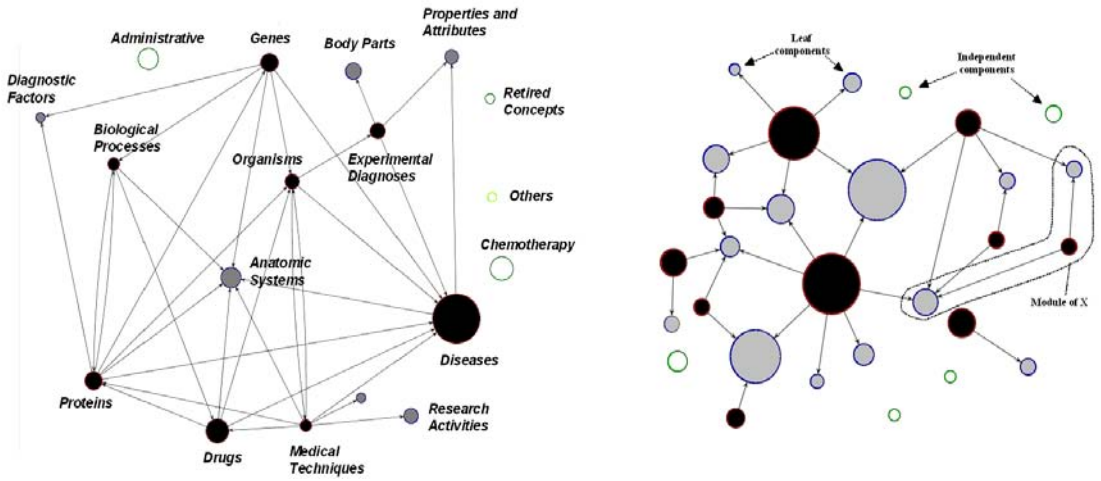

Figure 6: **Partitioning Graph for NCI (left) and OWL-S (right)**

and the existence of different subject matters within an OWL ontology.

The theorem provides an insight about the way the ontology has been modeled. In particular, it suggests one of the following: either the partitions correspond to actual non-overlapping subject matters, intended by the ontology engineer, or the ontology is underspecified and some of the partitions correspond to "unused information". In the latter case, these partitions identify parts of the ontology that probably need to be further developed.

An example of the latter case are the SWEET-JPL ontologies, which constitute NASA's effort for providing a formalization of the Earth Science domain. The SWEET ontologies include several thousand terms, spanning a broad extent of Earth Science and related concepts using OWL [3].

The resulting partitioning graph is shown in Figure 7. The partitioning reveals a significant number of small independent nodes. The existence of these small, independent fragments is hard to detect by direct inspection of the original ontologies and is not desirable from a modeling perspective, unless one actually wanted to evolve them separately.

The existence of the class of models identified in Theorem 3 makes it possible to identify axioms that *cannot* be entailed by the ontology $\mathcal{T}$:

Theorem 4 *Let $\mathcal{T}$ be safe and $G = \mathsf{Partition}(\mathcal{T})$ with $G = (\mathbf{V}, \mathbf{E}, \mathcal{L}, \mathcal{V})$, then the axioms of the following form **cannot** be entailed by $\mathcal{T}$: **1)** $C \sqsubseteq D$, with C, D local and $\mathcal{V}(C) \neq \mathcal{V}(D)$; **2)** $R \sqsubseteq S$ with $\mathcal{V}(R) \neq \mathcal{V}(S)$.*

We will use this result to show that the retrieved modules verify property 3) in Definition 3.

Identification and Extraction of Modules

The *module* for each concept is obtained from the partitioning graph using the algorithm in Figure 8. According to the Figure, if $\mathcal{V}(A) = v_i$, the **module** for A in $\mathcal{T}$ is the union of all the axioms contained in the nodes that are accessible from v_i through a directed path in G. There are cases, however, where the module for X computed this way does not satisfy Definition 3. For example, consider the following ontology:

$$\mathcal{T} = \{C \sqsubseteq \forall R.B\ ;\ B \sqsubseteq E;\ a \sqsubseteq C\ ;\ a \sqsubseteq \exists R.b\}$$

The partitioning algorithm would generate a graph with two nodes v, w, with $\mathcal{L}(v) = \{C \sqsubseteq \forall R.B; a \sqsubseteq C; a \sqsubseteq \exists R.b\}$ and $\mathcal{L}(w) = \{B \sqsubseteq E\}$ connected by an edge $\langle v, w\rangle$ with $\mathcal{L}(\langle v, w\rangle) = \{R\}$. The module $\mathcal{T}'_B$ for B would be just $\mathcal{T}'_B = \mathcal{L}(w)$; however $\mathcal{T} \models b \sqsubseteq B$, which is not entailed in $\mathcal{T}'_B$, thus violating Definition 3. The problem is caused by the presence of nominals. When the label of a node contains nominals, we need to "backtrack" in the graph and consider its predecessors as well (see Figure 8).

The correctness of our approach is based on the following theorems:

[3]The ontologies can be downloaded from http://sweet.jpl.nasa.gov/sweet

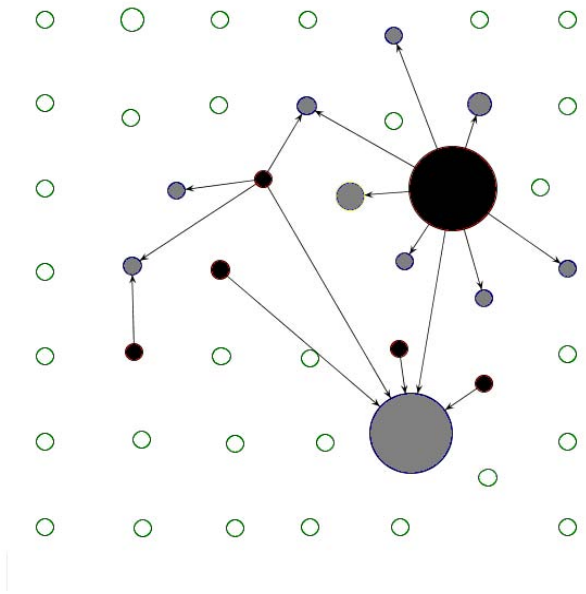

Figure 7: **Partitioning Graph for SWEET-JPL**

-Algorithm $\mathsf{GenerateModule}(G, C)$
-Input: The partition graph G
A concept C in $\mathcal{T}$
-Output: The module $\mathcal{T}'$ for C in $\mathcal{T}$

$v \leftarrow \mathcal{V}(C)$
$\mathcal{T}' \leftarrow \mathcal{L}(v)$
Add to $\mathcal{T}'$ all axioms in the label of the nodes accessible from v.
if $\mathcal{L}(v)$ has nominals, **then**
for each predecessor w of v in G:
Select any concept D in $\mathcal{L}(w)$
$\mathcal{T}' \leftarrow \mathcal{T}' \cup \mathsf{GenerateModule}(G, D)$
return $\mathcal{T}'$

Figure 8: Generation of Modules

Theorem 5 *The ontology* $\mathcal{T}' = \mathsf{GenerateModule}(G, C)$ *is a logical module of* $\mathcal{T}$.

Theorem 6 *The ontology* $\mathcal{T}' = \mathsf{GenerateModule}(G, C)$ *with* $G = \mathsf{Partition}(\mathcal{T})$ *is a module for* C *w.r.t.* $\mathcal{T}$.

It is not hard to verify that our modularization algorithm is worst-case quadratic in the size of the input ontology and hence the module for a concept in a consistent ontology can be obtained in polynomial time.

As an example of module extraction from a partitioning graph, consider Figure 6, which shows the decomposition for the OWL-S ontologies, describing Web Services. The ontology exhibits a nice decomposition, since a significant proportion of nodes correspond to independent or leaf nodes (white and gray nodes respectively), which is ideal for reuse. Interestingly, there is a improvement in modularity for every concept, in the sense that every module is *strictly smaller* than the ontology as a whole. Finally, note that the whole modularization process is *completely automatic*. No user intervention is required at any stage of the process.

Related Work

The problem of modularity in Web ontologies has been recently addressed in (StuckenSchmidt & Klein 2004), (Noy & Musen 2003) and (Seidenberg & Rector 2006).

In (StuckenSchmidt & Klein 2004), the output of the modularization process is presented as a graph visualization of the different kinds of information contained in the input ontology. However, the heuristics used to generate the visualization only consider a small fragment of OWL-DL and no correspondence between the nodes of the graph and sets of axioms is provided.

(Noy & Musen 2003) and (Seidenberg & Rector 2006) describe different structural techniques for extracting relevant fragments of ontologies. Although the output in these cases, as opposed to (StuckenSchmidt & Klein 2004), is a set of axioms, a formal characterization of their properties is lacking and hence no notion of correctness of the process is established.

(MacCartney *et al.* 2003) explores partitioning FOL theories to improve theorem proving performance. The work rigorously addresses logical issues, such as interpolation. However, the focus is on improving reasoning performance *only* and, thus, does not address reuse tasks. Our goal in this paper has been very different, since we have examined modularization primarily for reuse purposes.

In our previous work (Cuenca-Grau, Parsia, & E.Sirin 2005), we proposed $\mathcal{E}$-Connections (Kutz *et al.* 2004) as a suitable formalism for *combining* (rather than decomposing) OWL ontologies describing largely disjoint subject matters. There is indeed a tight relationship between $\mathcal{E}$-Connections and our partitioning algorithm. In fact, the partitioning graph can be seen *syntactically* as a knowledge base in the language of an $\mathcal{E}$-Connection, with the roles in the edges of the graph corresponding to *link relations*. This syntactic correspondence provides an intuition on why Theorems 3 and 4 hold. The reader should note, however, that the $\mathcal{E}$-Connections framework defines its own semantics; in fact, *all* the models of an $\mathcal{E}$-Connected KB are enforced to be of the form given in Theorem 3. In this paper, however, we see $\mathcal{E}$-Connections as a way of guiding the partitioning process, rather than as a logical formalism.

Conclusion

Ontology engineers need a clear notion of what to expect from a modularization process, both from a logical and a modeling perspective. Without such an understanding, the ontology engineer is at a loss. The result is the adoption of ad-hoc and highly unpredictable techniques as a common practice, which often leads to undesired results.

In this paper, we have presented a method for automatically identifying and extracting relevant fragments of ontologies, called modules, with precise semantic guarantees. Our method encompasses the full expressive power of OWL-DL and provides a good computational performance. Our initial experimental results with real-world ontologies

show that, for most concepts, the modules we obtain can be notably smaller than the original ontology, which facilitates re-use, processability, understandability and maintenance.

References

Bechhofer, S.; Lord, P.; and R.Volz. 2003. Cooking the semantic web with the OWL API. In *Proc. of the Second International Semantic Web Conference (ISWC-2003)*.

Cuenca-Grau, B.; Parsia, B.; and E.Sirin. 2005. Combining OWL ontologies using $\mathcal{E}$-connections. *Elsevier's Journal On Web Semantics* 4(1).

Garson, J. 1989. Modularity and relevant logic. *Notre Dame Journal of Formal Logic* 30(2):207–223.

Golbeck, J.; Fragoso, G.; Hartel, F.; Hendler, J.; Parsia, B.; and Oberthaler, J. 2003. The national cancer institute's thesaurus and ontology. *Elsevier's Journal of Web Semantics* 1(1).

Horrocks, I., and Sattler, U. 2005. A tableaux decision procedure for SHOIQ. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*. Morgan Kaufman.

Horrocks, I.; Patel-Schneider, P. F.; and van Harmelen, F. 2003. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Elsevier's Journal of Web Semantics* 1(1):7–26.

Horrocks, I.; Sattler, U.; and Tobies, S. 2000. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL* 8(3):239–263.

Kalyanpur, A.; Parsia, B.; E.Sirin; Cuenca-Grau, B.; and Hendler, J. 2005. Swoop: A web editing browser. *Elsevier's Journal On Web Semantics* 4(2).

Kutz, O.; Lutz, C.; Wolter, F.; and Zakharyaschev, M. 2004. $\mathcal{E}$-connections of abstract description systems. *Artificial Intelligence 156(1):1-73*.

MacCartney, B.; McIlraith, S. A.; Amir, E.; and Uribe, T. 2003. Practical partition-based theorem proving for large knowledge bases. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*.

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

Patel-Schneider, P.; Hayes, P.; and I.Horrocks. 2004. Web ontology language OWL Abstract Syntax and Semantics. *W3C Recommendation*.

Pitts, A. 1992. On an interpretation of second order quantification in first-order intuitionistic propositional logic. *Journal of Symbolic Logic* 1(57):33–52.

Seidenberg, J., and Rector, A. 2006. Web ontology segmentation: Analysis, classification and use. In *Proc. of the 2006 International World Wide Web Conference (WWW-2006)*.

Smith, M.; Welty, C.; and McGuiness, D. 2004. OWL Web Ontology Language Guide. *W3C Recommendation*.

StuckenSchmidt, H., and Klein, M. 2004. Structure-based partitioning of large class hierarchies. In *Proc. of the Third International Semantic Web Conference (ISWC 2004)*.

Wolter, F. 1998. Fusions of modal logics revisited. In Kracht, M.; de Rijke, M.; Wansing, H.; and Zakharyaschev, M., eds., *Advances in Modal Logic*. CSLI.

Appendix A: Proofs

Proof for Theorem 1

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ be an interpretation for C and $\mathcal{J} = (\Delta^{\mathcal{J}}, .^{\mathcal{J}})$ an expansion of $\mathcal{I}$ with set Φ.

First, it is easy to see that, for every role R (a role name or its inverse), $R^{\mathcal{J}} = R^{\mathcal{I}}$. Then, we proceed by induction on the structure of C. At the base of the induction, we have that, if C is a concept name then, by definition of $\mathcal{J}$, $C^{\mathcal{J}} = C^{\mathcal{I}}$ and $C \in \mathsf{local}(\mathbf{S})$.

We now verify the induction step:

- Let C be of the form $\neg D$. If $D \in \mathsf{local}(\mathbf{S})$, by induction, $D^{\mathcal{J}} = D^{\mathcal{I}}$. Since $C^{\mathcal{J}} = \Delta^{\mathcal{J}} \setminus D^{\mathcal{J}}$, $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}} \cup \Phi$ and $D^{\mathcal{J}} = D^{\mathcal{I}}$, then $C^{\mathcal{J}} = C^{\mathcal{I}} \cup \Phi$ and thus $C \notin \mathsf{local}(\mathbf{S})$. If $D \notin \mathsf{local}(\mathbf{S})$, by induction, $D^{\mathcal{J}} = D^{\mathcal{I}} \cup \Phi$; since $C^{\mathcal{J}} = \Delta^{\mathcal{J}} \setminus D^{\mathcal{J}}$, $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}} \cup \Phi$ and $D^{\mathcal{J}} = D^{\mathcal{I}} \cup \Phi$, then $C^{\mathcal{J}} = C^{\mathcal{I}}$ and thus $C \in \mathsf{local}(\mathbf{S})$.
- Let C be of the form $C_1 \sqcap C_2$. If $C_1, C_2 \notin \mathsf{local}(\mathbf{S})$, then $C^{\mathcal{J}} = C_1^{\mathcal{J}} \cap C_2^{\mathcal{J}} = (C_1^{\mathcal{I}} \cup \Phi) \cap (C_2^{\mathcal{I}} \cup \Phi) = C^{\mathcal{I}} \cup \Phi$ and thus $C \notin \mathsf{local}(\mathbf{S})$. If any $C_i \in \mathsf{local}(\mathbf{S})$, then it is immediate to verify that $C^{\mathcal{J}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} = C^{\mathcal{I}}$ and thus $C \in \mathsf{local}(\mathbf{S})$.
- If C is of the form $\exists R.D$ or $\geq nR.D$, it is easy to see that the induction hypothesis holds, since $R^{\mathcal{J}} = R^{\mathcal{I}}$, $C^{\mathcal{J}} = C^{\mathcal{I}}$ independently of whether D is local or not.

Proof for Theorem 2

($\Rightarrow$)

Suppose that $\mathcal{T}$ is unsafe, then there is a model $\mathcal{I} \models \mathcal{T}$ and a set Φ s.t. $\mathcal{J} \not\models \mathcal{T}$, with $\mathcal{J}$ being the expansion of $\mathcal{I}$ with Φ. We also have that $\mathcal{T}$ is composed of a set of GCIs $C \sqsubseteq D$ and each of these GCIs can be of one of the following forms (only):

1. Both C, D local concepts. By definition, $C^{\mathcal{J}} = C^{\mathcal{I}}$ and $D^{\mathcal{J}} = D^{\mathcal{I}}$. Therefore, $\mathcal{J} \models C \sqsubseteq D$
2. C local and D non-local. Now, $C^{\mathcal{J}} = C^{\mathcal{I}}$ and $D^{\mathcal{J}} = D^{\mathcal{I}} \cup \Phi$. Again, since $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, we have that $J \models C \sqsubseteq D$.
3. C and D non-local. Now, $C^{\mathcal{J}} = C^{\mathcal{I}} \cup \Phi$ and $D^{\mathcal{J}} = D^{\mathcal{I}} \cup \Phi$. Again, since $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, we have that $J \models C \sqsubseteq D$.
4. C non-local and D local. In this case, $C^{\mathcal{J}} = C^{\mathcal{I}} \cup \Phi$ and $D^{\mathcal{J}} = D^{\mathcal{I}}$. However, since $\Phi \cap D^{\mathcal{I}} = \emptyset$, $J \not\models C \sqsubseteq D$.

Note that in order not to be satisfied by $\mathcal{J}$, the GCI must be of the form 4), which is an unsafe GCI.

($\Leftarrow$)

Suppose that $\mathcal{T}$ contains explicitly an unsafe GCI $C \sqsubseteq D$. Since such a GCI is unsafe, C is non-local and D is local. Let $\mathcal{I}$ be a model of $\mathcal{T}$ and $\mathcal{J}$ an extension of $\mathcal{I}$ with some set Φ, then $D^{\mathcal{J}} = D^{\mathcal{I}}$ and $C^{\mathcal{J}} = C^{\mathcal{I}} \cup \Phi$ and, since $D^{\mathcal{J}} = D^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and $\Phi \cap \Delta^{\mathcal{I}} = \emptyset$, $\mathcal{J} \not\models C \sqsubseteq D$ and consequently $\mathcal{J} \not\models \mathcal{T}$; thus $\mathcal{T}$ is not safe.

Proof for Theorem 3

Since $\mathcal{T}$ is consistent, there exists an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ s.t. $\mathcal{I} \models \mathcal{T}$. We show that we can construct from $\mathcal{I}$ an interpretation $\mathcal{J}$ of the desired form s.t. $\mathcal{J} \models \mathcal{T}$. First, we define the domain $\Delta^{\mathcal{J}}$ of $\mathcal{J}$ using the following steps:

1. $\Delta^{\mathcal{J}} \leftarrow \emptyset$
2. For every $x \in \Delta^{\mathcal{I}}$, generate n new objects $x_1, ..., x_n$ and do $\Delta^{\mathcal{J}} \leftarrow \Delta^{\mathcal{J}} \cup \{x_1, ..., x_n\}$.

Now, we define the interpretation function $.^{\mathcal{J}}$ as well as the n sets $(\Delta_i^{\mathcal{J}})_{1 \leq i \leq n}$ as follows:

1. Initialize $\Delta_i^{\mathcal{J}} \leftarrow \emptyset$ for all $1 \leq i \leq n$; initialize $A^{\mathcal{J}}, R^{\mathcal{J}} \leftarrow \emptyset$ for each concept name A and role name R.
2. For every concept name A with $\mathcal{V}(A) = v_i$ and every $x \in A^{\mathcal{I}}$, do $A^{\mathcal{J}} \leftarrow A^{\mathcal{J}} \cup \{x_i\}$ and $\Delta_i^{\mathcal{J}} \leftarrow \Delta_i^{\mathcal{J}} \cup \{x_i\}$.
3. For every role name R s.t. $\mathcal{V}(R) = \langle v_i, v_j \rangle$ and every pair $\langle x, y \rangle \in R^{\mathcal{I}}$, do $R^{\mathcal{J}} \leftarrow R^{\mathcal{J}} \cup \langle x_i, y_j \rangle$, $\Delta_i^{\mathcal{J}} \leftarrow \Delta_i^{\mathcal{J}} \cup \{x_i\}$; $\Delta_j^{\mathcal{J}} \leftarrow \Delta_j^{\mathcal{J}} \cup \{y_j\}$.

By construction, it is easy to see that:

- $\Delta^{\mathcal{J}} = \bigcup_{i=1,...,n} \Delta_i^{\mathcal{J}}$, with $\Delta_i^{\mathcal{J}} \neq \emptyset$ for $1 \leq i \leq n$
- $\Delta_i^{\mathcal{J}} \cap \Delta_j^{\mathcal{J}} = \emptyset$ for $i \neq j$
- $A^{\mathcal{J}} \subseteq \Delta_i^{\mathcal{J}}$, for each A with $\mathcal{V}(A) = v_i$
- $R^{\mathcal{J}} \subseteq \Delta_i^{\mathcal{J}} \times \Delta_j^{\mathcal{J}}$, with $\mathcal{V}(R) = \langle v_i, v_j \rangle$

Note also that, by construction of $\mathcal{J}$, $\langle x_i, y_j \rangle \in R^{\mathcal{J}} \Leftrightarrow \langle x, y \rangle \in R^{\mathcal{I}}$, for every role.

We show that $\mathcal{J} \models \mathcal{T}$. For such a purpose, we use the following result (♣):

CLAIM(♣): Let C be a concept s.t $\mathcal{V}(C) = v_i$, then:

1. If C is local, then $C^{\mathcal{J}} = \{x_i | x \in C^{\mathcal{I}}\}$
2. If C is not local, then $C^{\mathcal{J}} = \bigcup_{k \neq i} \Delta_k^{\mathcal{J}} \cup \{x_i | x \in C^{\mathcal{I}}\}$

Using the definition of $\mathcal{J}$ and the properties of the partitioning graph, the claim is easily shown by induction on the structure of C; the induction uses similar arguments as the ones employed in the proof for Theorem 1. We just include here a sample case of the induction step in order to illustrate the arguments employed along the proof:

- If C of the form $\exists R.D$, then C is local by Theorem 1. By construction of the partitioning graph G, $\mathcal{V}(R) = \langle v_i, v_j \rangle$ for some $j \in \{1, ..., n\}$ and $\mathcal{V}(D) = v_j$. By definition of $\mathcal{J}$, $R^{\mathcal{J}} \subseteq \Delta_i^{\mathcal{J}} \times \Delta_j^{\mathcal{J}}$ and $\Delta_k^{\mathcal{J}} \cap \Delta_m^{\mathcal{J}} = \emptyset$ for $k \neq m$. Using the semantics of $\mathcal{SHOIQ}$ it is not hard to see that $C^{\mathcal{J}} \subseteq \Delta_i^{\mathcal{J}}$. It only remains to be shown that $x_i \in C^{\mathcal{J}}$ iff $x \in C^{\mathcal{I}}$, with $x_i \in \Delta_i^{\mathcal{J}}$:
  - ($\Rightarrow$) If $x_i \in C^{\mathcal{J}}$, then there exists an element $y_j \in \Delta_j^{\mathcal{J}}$ s.t. $\langle x_i, y_j \rangle \in R^{\mathcal{J}}$ and $y_j \in D^{\mathcal{J}}$. We have that $\langle x_i, y_j \rangle \in R^{\mathcal{J}} \Leftrightarrow \langle x, y \rangle \in R^{\mathcal{I}}$ and hence $\langle x, y \rangle \in R^{\mathcal{I}}$. By induction hypothesis, $y \in D^{\mathcal{I}}$. Therefore $x \in C^{\mathcal{I}}$.
  - ($\Leftarrow$) If $x \in C^{\mathcal{I}}$, then there exists an element $y \in \Delta_i^{\mathcal{J}}$ s.t. $\langle x, y \rangle \in R^{\mathcal{I}}$ and $y \in D^{\mathcal{I}}$. We have that $\langle x_i, y_j \rangle \in R^{\mathcal{J}} \Leftrightarrow \langle x, y \rangle \in R^{\mathcal{I}}$ and hence $\langle x_i, y_j \rangle \in R^{\mathcal{J}}$. By induction hypothesis, $y_j \in D^{\mathcal{J}}$ and thus $x_i \in C^{\mathcal{J}}$.

Using ♣ it is easy to show that $\mathcal{J} \models \mathcal{T}$. By safety of $\mathcal{T}$ and the properties of the partitioning graph, $\mathcal{T}$ can only contain GCIs $C \sqsubseteq D$ such that $\mathcal{V}(C) = \mathcal{V}(D) = v_i$ for some $i \in \{1, ..., n\}$ and either of the following:

1. Both C, D local concepts. By ♣, $C^{\mathcal{J}}$ only contains elements in $\Delta_i^{\mathcal{J}}$. Also by ♣, if $x_i \in C^{\mathcal{J}}$, then $x \in C^{\mathcal{I}}$. Since $\mathcal{I}$ satisfies the GCI, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and thus $x \in D^{\mathcal{I}}$. By ♣, $x \in D^{\mathcal{J}}$ and hence $\mathcal{J}$ satisfies the GCI.
2. For C local and D non-local, the argument is identical to the previous case
3. Both C, D non-local. By ♣, both $C^{\mathcal{J}}$ and $D^{\mathcal{J}}$ contain all the elements in $\Delta^{\mathcal{J}} \setminus \Delta_i^{\mathcal{J}}$; thus we focus only on elements of $\Delta_i^{\mathcal{J}}$. Again by ♣, if $x_i \in C^{\mathcal{J}}$, then $x \in C^{\mathcal{I}}$. Since $\mathcal{I}$ satisfies the GCI, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and thus $x \in D^{\mathcal{I}}$. By ♣, $x \in D^{\mathcal{J}}$ and hence $\mathcal{J}$ satisfies the GCI.

The fact that $\mathcal{J}$ satisfies the role inclusion and transitivity axioms in $\mathcal{T}$ is straightforward to verify.

Proof for Theorem 5

Lemma 1 *Let $\mathcal{T}' = \mathsf{GenerateModule}(\mathcal{T}, C)$ with $\mathcal{T}' \neq \mathcal{T}$ and suppose that $\mathsf{Sig}(\mathcal{T}') \cap \mathsf{Sig}(\mathcal{T} \setminus \mathcal{T}') = \emptyset$. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ be a model of $\mathcal{T}'$ and $\mathcal{J} = (\Delta^{\mathcal{J}}, .^{\mathcal{J}})$ be a model of $\mathcal{T} \setminus \mathcal{T}'$ s.t. $\Delta^{\mathcal{I}} \cap \Delta^{\mathcal{J}} = \emptyset$, then the interpretation $\mathcal{M} = (\Delta^{\mathcal{M}}, .^{\mathcal{M}})$ defined as follows:*

- $\Delta^{\mathcal{M}} = \Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}$.
- $A^{\mathcal{M}} = A^{\mathcal{I}}$ *if* $A \in \mathsf{Sig}(\mathcal{T}')$ *and* $A^{\mathcal{M}} = A^{\mathcal{J}}$ *otherwise.*
- $R^{\mathcal{M}} = R^{\mathcal{I}}$ *if* $R \in \mathsf{Sig}(\mathcal{T}')$ *and* $R^{\mathcal{J}}$ *otherwise.*

is a model of $\mathcal{T}$.

Proof:

$\mathcal{M} \models \mathcal{T}'$ as a consequence of the definition of safety, since:

1. $\mathcal{T}'$ is safe, and safe ontologies are invariant under domain expansions.
2. For the signature of $\mathcal{T}'$, $\mathcal{M}$ can be seen as the expansion of $\mathcal{I}$ with set $\Delta^{\mathcal{J}}$, since the signatures of $\mathcal{T}'$ and $\mathcal{T} \setminus \mathcal{T}'$ are disjoint.

Analogously, $\mathcal{T} \setminus \mathcal{T}'$ is safe and $\mathcal{M}$ can be seen, for the signature of $\mathcal{T} \setminus \mathcal{T}'$, as the expansion of $\mathcal{J}$ with set $\Delta^{\mathcal{I}}$. Thus $\mathcal{M} \models \mathcal{T} \setminus \mathcal{T}'$.

Lemma 2 *Let $\mathcal{T}' = \mathsf{GenerateModule}(\mathcal{T}, C)$ and suppose that: **1)** the signatures of $\mathcal{T}$ and $\mathcal{T} \setminus \mathcal{T}'$ share at least one symbol; **2)** $\mathsf{Sig}(\mathcal{T}')$ does not contain nominals.*

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ be a model of $\mathcal{T}'$ and $\mathcal{J} = (\Delta^{\mathcal{J}}, .^{\mathcal{J}})$ be a model of $\mathcal{T}$ s.t. $\Delta^{\mathcal{I}} \cap \Delta^{\mathcal{J}} = \emptyset$, then the interpretation $\mathcal{M} = (\Delta^{\mathcal{M}}, .^{\mathcal{M}})$ defined as follows:

- $\Delta^{\mathcal{M}} = \Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}$.
- $A^{\mathcal{M}} = A^{\mathcal{I}} \cup A^{\mathcal{J}}$, *if* $A \in \mathsf{Sig}(\mathcal{T}')$ *and* $A^{\mathcal{M}} = A^{\mathcal{J}}$ *otherwise.*

- $R^{\mathcal{M}} = R^{\mathcal{I}} \cup R^{\mathcal{J}}$ *if* $R \in \mathsf{Sig}(\mathcal{T}')$ *and* $R^{\mathcal{J}}$ *otherwise.*

is a model of $\mathcal{T}$.

Proof:

We first show the following claim ($\diamondsuit$):

1. For every concept C in $\mathcal{T}$ s.t. $C \in \mathsf{Con}(\mathsf{Sig}(\mathcal{T}'))$, $C^{\mathcal{M}} = C^{\mathcal{J}} \cup C^{\mathcal{I}}$.
2. For every other concept C occurring in $\mathcal{T}$, $C^{\mathcal{M}} = C^{\mathcal{J}}$, if C is local and and $C^{\mathcal{M}} = C^{\mathcal{J}} \cup \Delta^{\mathcal{I}}$ otherwise.

Proof for $\diamondsuit$:

Let $C \in \mathsf{Con}(\mathsf{Sig}(\mathcal{T}'))$. The proof goes by induction on the structure of C; the base of the induction is straightforward to verify, using the definition of $\mathcal{M}$. For the induction step, we include here, as a sample, the cases of negation and existential restriction; the remaining cases can be easily checked using similar arguments:

- Let C be of the form $\neg D$. By induction hypothesis, $D^{\mathcal{M}} = D^{\mathcal{J}} \cup D^{\mathcal{I}}$. By the semantics, $C^{\mathcal{M}} = \Delta^{\mathcal{M}} \setminus D^{\mathcal{M}}$. Hence, $C^{\mathcal{M}} = (\Delta^{\mathcal{J}} \cup \Delta^{\mathcal{I}}) \setminus (D^{\mathcal{J}} \cup D^{\mathcal{I}})$. Since $D^{\mathcal{J}} \subseteq \Delta^{\mathcal{J}}$, $D^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and $\Delta^{\mathcal{I}} \cap \Delta^{\mathcal{J}} = \emptyset$, we have that $C^{\mathcal{M}} = (\Delta^{\mathcal{I}} \setminus D^{\mathcal{I}}) \cup (\Delta^{\mathcal{J}} \setminus D^{\mathcal{J}}) = (\neg D)^{\mathcal{I}} \cup (\neg D)^{\mathcal{J}}$, which verifies the induction hypothesis.
- Let C be of the form $\exists R.D$. We have that $R^{\mathcal{M}} = R^{\mathcal{I}} \cup R^{\mathcal{J}}$ and by induction hypothesis $D^{\mathcal{M}} = D^{\mathcal{I}} \cup D^{\mathcal{J}}$, with $\Delta^{\mathcal{I}} \cap \Delta^{\mathcal{J}} = \emptyset$, $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and $R^{\mathcal{J}} \subseteq \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}}$. By the semantics of $(\exists R.D)$ it is easy to see that the induction hypothesis holds.

Let now $C \in \mathsf{Con}(\mathsf{Sig}(\mathcal{T} \setminus \mathcal{T}'))$ s.t. $C \notin \mathsf{Con}(\mathsf{Sig}(\mathcal{T}'))$. First note that if R is a role occurring in $\mathcal{T} \setminus \mathcal{T}'$, then $R^{\mathcal{M}} = R^{\mathcal{J}}$, for any role. The proof again goes by induction on the structure of C. The base of the induction is easy to verify; for the induction step, we include here the cases of negation and existential restriction:

- Let C be of the form $\neg D$. Two possibilities:
  - D is local, in which case C is non-local. Since D is local, by induction hypothesis $D^{\mathcal{M}} = D^{\mathcal{J}}$. By the semantics of concept negation, $C^{\mathcal{M}} = \Delta^{\mathcal{M}} \setminus D^{\mathcal{M}}$. Hence, $C^{\mathcal{M}} = (\Delta^{\mathcal{J}} \cup \Delta^{\mathcal{I}}) \setminus D^{\mathcal{J}}$. Since $D^{\mathcal{J}} \subseteq \Delta^{\mathcal{J}}$ and $\Delta^{\mathcal{I}} \cap \Delta^{\mathcal{J}} = \emptyset$, we have that $C^{\mathcal{M}} = \Delta^{\mathcal{I}} \cup (\Delta^{\mathcal{J}} \setminus D^{\mathcal{J}}) = \Delta^{\mathcal{I}} \cup (\neg D)^{\mathcal{J}} = \Delta^{\mathcal{I}} \cup C^{\mathcal{J}}$.
  - D is non-local, in which case C is local. Since D is non-local, by induction hypothesis $D^{\mathcal{M}} = D^{\mathcal{J}} \cup \Delta^{\mathcal{I}}$. By the semantics, $C^{\mathcal{M}} = \Delta^{\mathcal{M}} \setminus D^{\mathcal{M}}$. Hence, $C^{\mathcal{M}} = (\Delta^{\mathcal{J}} \cup \Delta^{\mathcal{I}}) \setminus (\Delta^{\mathcal{I}} \cup D^{\mathcal{J}}$. Since $D^{\mathcal{J}} \subseteq \Delta^{\mathcal{J}}$ and $\Delta^{\mathcal{I}} \cap \Delta^{\mathcal{J}} = \emptyset$, we have that $C^{\mathcal{M}} = \Delta^{\mathcal{J}} \setminus D^{\mathcal{J}} = (\neg D)^{\mathcal{J}}$.
- If C is of the form $\exists R.D$, then C is local. Given the way modules are generated, we have two possibilities
  - $D \in \mathsf{Con}(\mathsf{Sig}(\mathcal{T} \setminus \mathcal{T}'))$ but not in $\mathsf{Con}(\mathsf{Sig}(\mathcal{T}'))$. We have $R^{\mathcal{M}} = R^{\mathcal{J}}$. Again two possibilities:
    - D is local, which implies that $D^{\mathcal{M}} = D^{\mathcal{J}}$. It is immediate to see that $C^{\mathcal{M}} = C^{\mathcal{J}}$ using the semantics of existential restrictions.
    - D is non-local, in which case $D^{\mathcal{M}} = D^{\mathcal{J}} \cup \Delta^{\mathcal{I}}$. Since $R^{\mathcal{M}} = R^{\mathcal{J}}$, there is no element $y \in \Delta^{\mathcal{M}}$ s.t. $\langle x, y \rangle \in R^{\mathcal{M}}$ and $y \in \Delta^{\mathcal{I}}$, since y must be in $\Delta^{\mathcal{J}}$. Therefore $C^{\mathcal{M}} = C^{\mathcal{J}}$ and the induction hypothesis holds.
  - $D \in \mathsf{Con}(\mathsf{Sig}(\mathcal{T}'))$. In this case, again $R^{\mathcal{M}} = R^{\mathcal{J}}$ and $D^{\mathcal{M}} = D^{\mathcal{J}} \cup D^{\mathcal{I}}$. We have that $D^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and the proof reduces to the case above.

Using the safety of $\mathcal{T}$ and $\diamondsuit$, it is not hard to see that $\mathcal{M} \models \mathcal{T}$. In particular,

- if $C \sqsubseteq D \in \mathcal{T}'$ then, by $\diamondsuit$, $C^{\mathcal{M}} = C^{\mathcal{J}} \cup C^{\mathcal{I}}$ and $D^{\mathcal{M}} = D^{\mathcal{J}} \cup D^{\mathcal{I}}$. Since $\mathcal{I}, \mathcal{J} \models \mathcal{T}'$, then $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $C^{\mathcal{J}} \subseteq D^{\mathcal{J}}$; therefore, $C^{\mathcal{M}} \subseteq D^{\mathcal{M}}$ and $\mathcal{M}$ satisfies the axiom.
- if $C \sqsubseteq D \in \mathcal{T} \setminus \mathcal{T}'$ then, by $\diamondsuit$, $C^{\mathcal{M}} = C^{\mathcal{J}}$, if C is local and $C^{\mathcal{M}} = C^{\mathcal{J}} \cup \Phi$, if C is non-local; analogously for D. Since $\mathcal{T}$ is safe, we can only have safe GCIs in $\mathcal{T} \setminus \mathcal{T}'$; it is easy to verify that, if $C \sqsubseteq D$ is safe and $\mathcal{J}$ satisfies it, then also does $\mathcal{M}$.

Proof for Theorem 5

We now prove Theorem 5 using Lemma 1 and Lemma 2.

In order for $\mathcal{T}'$ to be locally complete w.r.t. $\mathcal{T}$ it must verify the following condition: for every axiom $C \sqsubseteq D$ s.t. $C, D \in \mathsf{Con}(\mathsf{Sig}(\mathcal{T}'))$, if $\mathcal{T} \models C \sqsubseteq D$, then $\mathcal{T}' \models C \sqsubseteq D$.

By the way modules are generated, we have three possibilities:

1. $\mathcal{T}' = \mathcal{T}$.
2. $\mathsf{Sig}(\mathcal{T} \setminus \mathcal{T}')$ and $\mathsf{Sig}(\mathcal{T}')$ are disjoint.
3. $\mathsf{Sig}(\mathcal{T} \setminus \mathcal{T}')$ and $\mathsf{Sig}(\mathcal{T}')$ are not disjoint and $\mathsf{Sig}(\mathcal{T}')$ does not contain nominals.

Case **1)** is obvious; we prove **2)** and **3)**

Case **2):** Suppose that $\mathcal{T} \models C \sqsubseteq D$ but $\mathcal{T}' \not\models C \sqsubseteq D$. Then, there is a model $\mathcal{I}$ of $\mathcal{T}'$ s.t. $\mathcal{I} \not\models C \sqsubseteq D$. Since $\mathsf{Sig}(\mathcal{T}')$ and $\mathsf{Sig}(\mathcal{T} \setminus \mathcal{T}')$ are disjoint, we can always find a model $\mathcal{J} \models \mathcal{T} \setminus \mathcal{T}'$ s.t. $\Delta^{\mathcal{I}} \cap \Delta^{\mathcal{J}} = \emptyset$. Then, the interpretation $\mathcal{M}$ as defined in Lemma 1 is a model of $\mathcal{T}$. By Theorem 2, the GCI $C \sqsubseteq D$ must be safe. It is not hard to see that $\mathcal{M} \not\models C \sqsubseteq D$, which yields a contradiction.

Case **3):** Suppose that $\mathcal{T} \models C \sqsubseteq D$ but $\mathcal{T}' \not\models C \sqsubseteq D$. Then, there is a model $\mathcal{I}$ of $\mathcal{T}'$ s.t. $\mathcal{I} \not\models C \sqsubseteq D$. Since $\mathcal{T}'$ does not contain nominals, we can always find a model $\mathcal{J} \models \mathcal{T}$ s.t. $\Delta^{\mathcal{I}} \cap \Delta^{\mathcal{J}} = \emptyset$. Then, the interpretation $\mathcal{M}$ as defined in Lemma 2 is a model of $\mathcal{T}$. Since $\mathcal{I} \not\models C \sqsubseteq D$, $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}}$. Since, by $\diamondsuit$ in Lemma 2 $C^{\mathcal{M}} = C^{\mathcal{I}} \cup C^{\mathcal{J}}$ and $D^{\mathcal{M}} = D^{\mathcal{I}} \cup D^{\mathcal{J}}$ and by definition of $\mathcal{I}, \mathcal{J}$, $C^{\mathcal{I}} \cap \Delta^{\mathcal{J}} = \emptyset$ it follows that $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}} \cup D^{\mathcal{J}}$ and therefore $\mathcal{M} \not\models C \sqsubseteq D$, which yields a contradiction.

Proof for Theorem 6

We show that $\mathcal{T}'$ is a module for A w.r.t. $\mathcal{T}$. First, $\mathcal{T}'$ is a logical module, as shown in Theorem 5. Second, condition 3) in Definition 3 holds as a direct consequence of Theorem 4. We now verify condition 2) in Definition 3.Let B a concept name in $\mathsf{Sig}(\mathcal{T})$. Two possibilities:

- $B \in \mathsf{Sig}(\mathcal{T}')$. In this case Properties 2a), 2b) in Definition 2 hold as a straightforward consequence of Theorem 5.
- $B \notin \mathsf{Sig}(\mathcal{T}')$. In this case, by Theorem 4, $\mathcal{T} \not\models A \sqsubseteq B$ and $\mathcal{T} \not\models B \sqsubseteq A$. By monotonicity, these entailments also do not hold in $\mathcal{T}'$ and thus 2a), 2b) also hold.