Ontology Partitioning Using $\mathcal{E}$-Connections Revisited

Sascha Jongebloed and Thomas Schneider*

Department of Computer Science, University of Bremen, Germany
{sasjonge,ts}@cs.uni-bremen.de

Abstract. We revisit the technique for partitioning ontologies using $\mathcal{E}$-connections introduced by Cuenca Grau et al. (2006). We extend the underlying notions of $\mathcal{E}$-connections and partitionings to the latest OWL 2 ontology language, with the exception of the universal role, which cannot be accommodated for reasons that we explain. Besides presenting a simplified notation for the theoretical foundations, we devise a new algorithm for computing partitions which, in contrast to the original quadratic-time algorithm, can be implemented in linear time, is deterministic, and admits simple rigorous correctness and maximality proofs. We further discuss possible extensions beyond OWL.

1 Introduction

Modularity is an important paradigm for ontology development and use, which has received much attention in the past decade. Modular ontologies are usually easier to maintain, comprehend, and reason over, and extracting a module of a given ontology is an important task in scenarios where only a part of the knowledge in that ontology is to be reused efficiently. If an ontology is given as a monolithic entity, the task of *decomposing* it into modules is the first step towards making it modular and being able to enjoy the previously mentioned benefits.

Partitionings based on $\mathcal{E}$-connections (henceforth: $\mathcal{E}$-partitions) have been developed for the purpose of automatically and efficiently decomposing an ontology [6,9,10]. The result is a graph whose nodes are (pairwise disjoint and mutually covering) *components* (i.e., subsets) of the ontology, and whose edges represent "semantic links" between the components in the style of $\mathcal{E}$-connections [20]. The adoption of the $\mathcal{E}$-connection framework ensures that the resulting partitions provide strong logical guarantees; thus (certain combinations of) the components are encapsulating modules of the input ontology [10].

$\mathcal{E}$-Connections were defined for *abstract description systems (ADSs)*, a notion that generalises many DLs, modal logics, and further formalisms [3]. An $\mathcal{E}$-connection is a combination of heterogeneous logical theories via semantic links established by a designated set of relations, called *link relations*. Their semantics is given by interpretations that consist of pairwise disjoint components and which interpret each component locally and the link relations as relations between the respective components. In contrast to the general nature of $\mathcal{E}$-connections, $\mathcal{E}$-partitions and the underlying framework have been defined specifically for the DL $\mathcal{SHOIQ}(\mathcal{D})$, a fragment of the latest OWL 2 ontology language. The partitioning procedure starts from a monolithic ontology $\mathcal{O}$ and attempts

* Supported by DFG project SCHN 1234/3.

to turn it into an (as fine as possible) $\mathcal{E}$-connection by identifying link relations among the roles *in* $\mathcal{O}$. An efficient algorithm for computing $\mathcal{E}$-partitions is given in [6,9,10]: is traverses the subconcepts of the input ontology, and assigns to the current concept either a fresh component (if this is witnessed by some freshly identified link relation) or an existing component (when necessary for conformance with the definition of $\mathcal{E}$-connections). When all concepts are dealt with, the assignment of components induces a distribution of the ontology's axioms over the components, constituting the partition. To ensure that the resulting $\mathcal{E}$-partition and the input ontology are equivalent under the $\mathcal{E}$-connection semantics, an additional condition has to be imposed on the input ontology, which was called safety in [6,9,10] and coincides with *domain-independence (DI)* known from first-order logic and database theory [1]. Contrary to DI in first-order logic, DI for $\mathcal{SHOIQ}(\mathcal{D})$ is decidable via a neat syntactic criterion called *locality* [10].

The partitioning algorithm was implemented as an experimental feature of the (discontinued) ontology editor Swoop.[1] In initial experiments [10], some ontologies admitted useful $\mathcal{E}$-partitions, and some of those revealed modelling deficiencies. However, some modelling patterns – e.g., the use of few top-level concepts or exhaustive declaration of disjoint concepts or distinct individuals – are notoriously problematic: they admit only coarse $\mathcal{E}$-partitions, which follows directly from the definitions underlying $\mathcal{E}$-connections. This may be one of the reasons for the limited success of $\mathcal{E}$-partitions. Another reason certainly lies in the preliminary nature of the existing implementation: it is incomplete (not all input axioms are preserved), incorrect (sometimes axioms are spuriously assigned to distinct components), and nondeterministic (we sometimes obtained different results, re-running the algorithm on the same input). Hence some observed "poor" $\mathcal{E}$-partitions might be due to a bug in the code rather than a "feature" of the general approach.

When trying to separate bugs from "features", we found a considerably simpler way of computing $\mathcal{E}$-partitions, based on the idea of creating an undirected graph G whose edges connect concepts and/or roles that must be part of the same component, and reading the minimal $\mathcal{E}$-partition off G's connected components. We were able to extend the underlying framework to most of OWL 2, with the only exception of the universal role u. This exception is unavoidable: with u, locality no longer characterises domain-independence, thus destroying the guarantee that the resulting $\mathcal{E}$-partition and the input ontology are equivalent. We consider this exception insignificant because u "typically plays a minor role in modelling" [19], confirmed by a large corpus of ontologies.[2]

Contributions and overview. Our new approach offers the following advantages over the original one:

- Ready applicability to the latest OWL 2 standard, under two restrictions ensuring equivalence (domain-independence, absence of the universal role)
- A simplified notation of the theoretical foundations
- A new, simpler partitioning algorithm which is deterministic – i.e., does not depend on the traversal order of concepts and axioms – and can be implemented to run in linear time (the original one is quadratic [10])
- Simple rigorous proofs that the algorithm is correct and outputs the maximal $\mathcal{E}$-connection that is equivalent to the input ontology under the $\mathcal{E}$-connection semantics

[1] `https://github.com/ronwalf/swoop`

[2] In last year's BioPortal corpus [21], u occurs in only 322 of all 16.3 million axioms.

Given these advantages, we expect that the algorithm is easy to implement and to evaluate on an up-to-date ontology corpus. Hence, the work reported here is in progress. We briefly introduce OWL (§2), define our extension of $\mathcal{E}$-connections and $\mathcal{E}$-partitions to $\mathcal{SROIQ}$ (§3), present the new algorithm (§4), discuss extensions to OWL and beyond (§5), and conclude in §6. Implementation and tests are subject to future work.

Due to space restrictions, we provide most proofs and further examples in a technical report: `http://www.informatik.uni-bremen.de/tdki/research/papers.html`

Related work. Numerous module extraction and modularisation approaches are known [18,30,7,8,11,29,12], as well as ontology languages supporting modular development [5,27]. Several decomposition approaches have been developed, some of which provide strong logical guarantees (such as encapsulation), e.g., signature decomposition [17], partitionings based on $\mathcal{E}$-connections [10,8], and atomic decomposition [11]; and others are rather based on statistical parameters, such as structure-based partitioning [29,2].

2 Preliminaries

The ontology language OWL is based on the DL $\mathcal{SROIQ}$ [15]. OWL additionally provides datatypes [23] and keys [25], whose treatment we defer to Section 5. Here and in Section 3, we consider $\mathcal{SROIQ}$ without the universal role (see above). We give only a short overview of the syntax and refer to [19] for semantics and further details. We denote concept names with $A, B, \ldots$, complex concepts with $C, D, \ldots$, role names with $r, s, \ldots$, complex roles (role names r or inverses r^-) with $R, S, \ldots$, individual names with $a, b, \ldots$, and axioms with $\alpha, \beta, \ldots$. Concepts and axioms are built according to the following grammars, where $m \in \mathbb{N}$ and the remaining letters are as explained above.

$$
\begin{aligned}
C &::= A \mid \neg C \mid C \sqcap D \mid \geqslant m\, R.C \mid \exists R.\mathsf{Self} \mid \{a\} \\
\alpha &::= C \sqsubseteq D \mid C \equiv D \mid R \sqsubseteq S \mid R \equiv S \mid \mathsf{Disjoint}(R, S) \mid R_1 \circ R_2 \sqsubseteq S \mid \\
&\quad\; C(a) \mid R(a, b) \mid a \approx b \mid a \not\approx b,
\end{aligned}
$$

The remaining operators $\top, \bot, \sqcup, \exists R, \forall R, \leqslant n\, R$ and axiom types (role transitivity, symmetry, etc.) are "syntactic sugar"; hence we do not treat them explicitly, thus simplifying the presentation without losing generality. We furthermore ignore OWL's global restrictions (regularity, restricted use of non-simple roles) as $\mathcal{E}$-connections and partitionings do not rely on them. An *ontology* is a set of axioms.

3 $\mathcal{E}$-Connections and Partitionings for $\mathcal{SROIQ}$

The definitions in this section largely follow those in [6,9]; however, we simplify notation to allow, as we believe, a more concise presentation. The differences and their motivation are explained at the end of this section. In the following, we work with a fixed *signature*, which is a finite set Σ of concept names, role names, and individual names (together: *terms*), i.e., $\Sigma = \Sigma_\mathsf{C} \uplus \Sigma_\mathsf{R} \uplus \Sigma_\mathsf{I}$. Here and in the following, $\uplus$ denotes disjoint union.

Given a natural number $n \geq 1$, an *n-numbering of Σ* is a function ν that assigns to each concept and individual name a number $\nu(A), \nu(a) \in \{1, \ldots, n\}$ and to each role name

a pair $\nu(r) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$ of numbers. For what follows, the order between the numbers is irrelevant; an arbitrary finite index set I instead of $\{1, \ldots, n\}$ would suffice.

Numberings are extended to arbitrary concepts and axioms inductively:

Definition 1. Let ν be an n-numbering of Σ.

- ν is extended to arbitrary roles R and concepts C as follows.
  1. If $R = r$, then $\nu(R) = \nu(r)$.
  2. If $R = r^-$, and $\nu(r) = (i, j)$, then $\nu(R) = (j, i)$.
  3. If $C = \neg D$, then $\nu(C) = \nu(D)$.
  4. If $C = D \sqcap E$ and $\nu(D) = \nu(E)$, then $\nu(C) = \nu(D)$.
  5. If $C = \geqslant m\, R.D$ and $\nu(R) = (i, j)$ and $\nu(D) = j$, then $\nu(C) = i$.
  6. If $C = \exists R.\mathsf{Self}$ and $\nu(R) = (i, i)$, then $\nu(C) = i$.
  7. If $C = \{a\}$, then $\nu(C) = \nu(a)$.

  A concept C is called an *i-concept* if $\nu(C)$ is defined[3] and $\nu(C) = i$.
- ν is further extended to axioms as follows. An *i-axiom* is of the form
  8. $C \sqsubseteq D$ or $C \equiv D$ where $\nu(C) = \nu(D) = i$;
  9. $R \sqsubseteq S$, $R \equiv S$, or $\mathsf{Disjoint}(R, S)$ where $\nu(R) = \nu(S) = (i, j)$;
  10. $R_1 \circ R_2 \sqsubseteq S$ where $\nu(R_1) = (i, j)$, $\nu(R_2) = (j, k)$, and $\nu(S) = (i, k)$;
  11. $C(a)$ where $\nu(C) = \nu(a) = i$;
  12. $R(a, b)$ where $\nu(a) = i$[4] and $\nu(R) = (\nu(a), \nu(b))$;
  13. $a \approx b$ or $a \not\approx b$ where $\nu(a) = \nu(b) = i$.

The central notion of an ontology consisting of several parts depends on ν:

Definition 2. Given an n-numbering ν, a *ν-ontology* is a tuple $\mathbb{O} = (\mathcal{O}_1, \ldots, \mathcal{O}_n)$, each of whose *components* $\mathcal{O}_i$ is a nonempty set of i-axioms.

To distinguish (monolithic) ontologies $\mathcal{O}$ from (combined) ν-ontologies $\mathbb{O}$, we sometimes call the former *simple ontologies*.

The semantics of ν-ontologies is given by ν-interpretations, which are partitioned according to the numbering of the signature given by ν:

Definition 3. A *ν-interpretation* is an interpretation $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I},\nu})$ such that

- $\Delta^{\mathcal{I}} = \biguplus_{i \leqslant n} \Delta_i^{\mathcal{I}}$ with $\Delta_i \neq \emptyset$ for each *component* Δ_i
- $A^{\mathcal{I},\nu} \subseteq \Delta_i^{\mathcal{I}}$ for all $A \in \Sigma_{\mathsf{C}}$ with $\nu(A) = i$
- $r^{\mathcal{I},\nu} \subseteq \Delta_i^{\mathcal{I}} \times \Delta_j^{\mathcal{I}}$ for all $r \in \Sigma_{\mathsf{R}}$ with $\nu(r) = (i, j)$
- $a^{\mathcal{I},\nu} \in \Delta_i^{\mathcal{I}}$ for all $a \in \Sigma_{\mathsf{I}}$ with $\nu(a) = i$

ν-interpretations are used for two purposes: (a) for defining the semantics of simple ontologies given ν, and (b) for defining the semantics of ν-ontologies, which is the usual semantics of $\mathcal{E}$-connections [20]. In case (a), we use the standard DL semantics and the usual satisfaction symbol $\models$. We say that $\mathcal{O}$ has a *ν-model* if there is a ν-interpretation $\mathcal{I} \models \mathcal{O}$. The class of ν-models of $\mathcal{O}$ is contained in the class of its (unrestricted) models. For example, if $\mathcal{O}$ contains $\top \sqsubseteq A$, then all ν-models have a single component; thus, if ν is an n-numbering with $n \geq 2$, then $\mathcal{O}$ has no ν-models but may still be consistent (i.e., have unrestricted models). For purpose (b), the interpretation function is extended ensuring that the extension of arbitrary i-concepts is a subset of $\Delta_i^{\mathcal{I}}$. For this purpose, (only) the case of negation has to differ from the standard semantics.

[3] $\nu(C)$ is undefined, e.g., for concepts $\exists r.D$ with $\nu(r) = (i, j)$ and $\nu(D) \neq j$.

[4] This choice is arbitrary; it could also be "$\nu(b) = i$". Analogously for item 10.

Definition 4. The interpretation function of a ν-interpretation $\mathcal{I}$ is extended to arbitrary ν-concepts as follows.

- $(\neg C)^{\mathcal{I},\nu} = \Delta_i^{\mathcal{I}} \setminus C^{\mathcal{I},\nu}$ for i-concepts C
- $(C \sqcap D)^{\mathcal{I},\nu} = C^{\mathcal{I},\nu} \cap D^{\mathcal{I},\nu}$
- $(\geqslant m\, R.C)^{\mathcal{I},\nu} = \{d \in \Delta_i^{\mathcal{I}} \mid \#\{e \in \Delta_j^{\mathcal{I}} \mid (d,e) \in R^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I},\nu}\} \geq m\}$ for (i,j)-roles R and j-concepts C
- $(\exists R.\mathsf{Self})^{\mathcal{I},\nu} = \{d \in \Delta_i^{\mathcal{I}} \mid (d,d) \in R^{\mathcal{I}}\}$ for (i,i)-roles R
- $\{a\}^{\mathcal{I},\nu} = a^{\mathcal{I},\nu}$

Satisfaction of i-axioms in ν-interpretations is defined as follows.

- $\mathcal{I} \models^{\nu} C \sqsubseteq D$ if $C^{\mathcal{I},\nu} \subseteq D^{\mathcal{I},\nu}$
- $\mathcal{I} \models^{\nu} C \equiv D$ if $C^{\mathcal{I},\nu} = D^{\mathcal{I},\nu}$
- $\mathcal{I} \models^{\nu} C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I},\nu}$
- For i-axioms α of all other types (role inclusion, equivalence, disjointness, assertion, or individual (in)equality), $\mathcal{I} \models^{\nu} \alpha$ if $\mathcal{I} \models \alpha$.

$\mathcal{I}$ is a *model* of a ν-ontology $\mathbb{O}$, written $\mathcal{I} \models^{\nu} \mathbb{O}$, if $\mathcal{I} \models^{\nu} \alpha$ for all axioms α in $\mathbb{O}$.
$\mathbb{O}$ is *consistent* if it has a model.

The correspondence between simple and ν-ontologies is captured by compatibility and equivalence. The former notion is syntactic and only requires that the components of $\mathbb{O}$ partition $\mathcal{O}$. The latter is semantic and relative to ν-interpretations.

Definition 5. Let $\mathcal{O}$ be an ontology, ν an n-numbering, and $\mathbb{O} = (\mathcal{O}_1, \ldots, \mathcal{O}_n)$ a ν-ontology.

1. $\mathcal{O}$ and $\mathbb{O}$ are *compatible*, written $\mathcal{O} \sim \mathbb{O}$, if $\mathcal{O} = \bigcup_{i \leq n} \mathcal{O}_i$.
2. $\mathcal{O}$ and $\mathbb{O}$ are *equivalent*, written $\mathcal{O} \approx \mathbb{O}$ if, for all ν-interpretations $\mathcal{I}$, it holds that $\mathcal{I} \models \mathcal{O}$ iff $\mathcal{I} \models^{\nu} \mathbb{O}$.

Unsurprisingly, compatibility does not imply equivalence, and neither does the converse hold. The latter is trivial: if $\mathcal{O} = \{A \sqsubseteq B\}$ and $\mathbb{O} = \{\neg B \sqsubseteq \neg A\}$, then $\mathcal{O} \approx \mathbb{O}$ but $\mathcal{O} \not\sim \mathbb{O}$. For the other direction, take $\mathcal{O} = \{\neg A \sqsubseteq A',\ B \sqsubseteq B'\}$ and $\mathbb{O} = (\{\neg A \sqsubseteq A'\},\ \{B \sqsubseteq B'\})$. Then $\mathbb{O}$ is well-formed according to Definitions 1 and 2 and $\mathcal{O} \sim \mathbb{O}$, but $\mathcal{O} \not\approx \mathbb{O}$ because $\mathbb{O}$ has ν-models $\mathcal{I}$ with 2 components but no such $\mathcal{I}$ is a model of $\mathcal{O}$. The reason is that the axiom $\neg A \sqsubseteq A'$ cannot be satisfied if the extension of both A and A' is restricted to only one component, as required by Definition 3. As a consequence of this observation, an additional assumption has to be made, which is the DL equivalent of domain-independence known from the first-order and database worlds [1]. For (most of) $\mathcal{SROIQ}$, this assumption admits an efficiently decidable syntactic characterisation. For domain-independent ontologies, compatibility implies equivalence; see Theorem 9.

Definition 6. A concept C (axiom α) is *domain-independent (DI)* if $C^{\mathcal{I}} = C^{\mathcal{J}}$ ($\mathcal{I} \models \alpha$ iff $\mathcal{J} \models \alpha$) for all interpretations $\mathcal{I}, \mathcal{J}$ with $X^{\mathcal{I}} = X^{\mathcal{J}}$ for all terms X. An ontology is DI if so are all its axioms.

The syntactic characterisation from [6,9,10] is the following.

Definition 7. The set of *local* concepts is defined inductively as follows.

- Every concept name is local.
- $\neg C$ is local if C is not.
- $C \sqcap D$ is local if C or D is local.
- $\geqslant m\, R.C$, $\exists R.\mathsf{Self}$, and $\{a\}$ are local.

An ontology $\mathcal{O}$ is *safe* if the following hold.

- For all axioms $C \sqsubseteq D$, if D is local, then so is C.
- For all axioms $C \equiv D$, C is local iff so is D.

Given an interpretation $\mathcal{I}$ and a set S, we write $\mathcal{J} = \mathcal{I} \uplus S$ if $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}} \uplus S$ and $X^{\mathcal{J}} = X^{\mathcal{I}}$ for all terms X.

Theorem 8 ([6,9,10]). *For all concepts C and ontologies $\mathcal{O}$:*

1. *C is DI iff C is local.*
2. *If C is DI, then $C^{\mathcal{J}} = C^{\mathcal{I}}$ for all $\mathcal{I}$ and $\mathcal{J} = \mathcal{I} \uplus S$.*
3. *If C is not DI, then $C^{\mathcal{J}} = C^{\mathcal{I}} \cup S$ for all $\mathcal{I}$ and $\mathcal{J} = \mathcal{I} \uplus S$.*
4. *$\mathcal{O}$ is DI iff $\mathcal{O}$ is safe.*

Not only does Theorem 8 provide a syntactic characterisation of DI; with Definition 7 it also yields a linear-time decision procedure. Here it becomes clear why the universal role u (with the semantics $u^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$) cannot be accommodated by the framework: the concept $C = \exists u.A$ is not DI but violates point 3 in Theorem 8: there are interpretations $\mathcal{I}$ and $\mathcal{J} = \mathcal{I} \uplus S$ with $C^{\mathcal{J}} = C^{\mathcal{I}} \cup S$ (e.g., $\Delta^{\mathcal{I}} = A^{\mathcal{I}} = \{d\}$ and $S = \{e\}$); but if $A^{\mathcal{I}} = \emptyset$ then $C^{\mathcal{J}} = C^{\mathcal{I}} = \emptyset$. However, point 3 is an essential ingredient not just in the syntactic characterisation witnessing decidability of DI, but also in the following theorem linking compatibility and equivalence. Its proof completes the proof of [6, Theorem 7.2].

Theorem 9. *Let $\mathcal{O}$ be an ontology, ν an n-numbering, $\mathbb{O} = (\mathcal{O}_1, \ldots, \mathcal{O}_n)$ a ν-ontology.*

1. *(a) If $\mathcal{O}$ is DI and $\mathcal{O} \sim \mathbb{O}$, then $\mathcal{O} \approx \mathbb{O}$.*
   (b) If additionally $\mathcal{O}$ is consistent, then so is $\mathbb{O}$.
2. *If $\mathcal{O}$ is consistent and not DI, and $\mathcal{O} \sim \mathbb{O}$ and $\mathcal{O} \approx \mathbb{O}$, then $n = 1$, i.e., $\mathbb{O} = \mathcal{O}$.*

Main differences in notation compared to [6,9,10]. n-numbered signatures correspond to "partitioned vocabularies" in [6,9]. We consider it easier to carry along the parameter ν instead of a partitioned vocabulary with its rather long denotation. Numberings also extend more naturally to complex concepts and axioms. Consequently, "combined knowledge bases" (a special case of $\mathcal{E}$-connections) are now called ν-ontologies.

ν-interpretations conflate "partitioned interpretations" and "combined interpretations", making "corresponding interpretations" redundant. Consequently we distinguish two semantics: $\mathcal{I} \models \mathcal{O}$ (the standard semantics for simple ontologies) and $\mathcal{I} \models^{\nu} \mathbb{O}$ (the $\mathcal{E}$-connection semantics for ν-ontologies).

Equivalence replaces "semantic compatibility" because it indeed captures equivalence under the class of ν-interpretations. As per the usual understanding of equivalence, $\mathcal{O}$ is not required to have (ν-)models. The implication "if $\mathcal{O}$ consistent, then also $\mathbb{O}$" (under certain assumptions) is now captured by Theorem 9. As a result we reduced "syntactically compatible" to "compatible".

Finally, domain-independence replaces "invariance under domain expansions". We decided to reflect the relationship with the fundamental first-order notion.

4 The New Partitioning Algorithm

We now present the partitioning algorithm, based on the preceding definitions. As in [6,9,10], our algorithm receives as input an ontology $\mathcal{O}$ and returns a ν-ontology $\mathbb{O} = (\mathcal{O}_1, \ldots, \mathcal{O}_n)$ such that $\mathcal{O} \sim \mathbb{O}$ and n is maximal with this property. If $\mathcal{O}$ is domain-independent, $\mathcal{O} \approx \mathbb{O}$ follows by Theorem 9. The numbering ν is not computed by the algorithm but is implicit in the created data structure.

Our algorithm determines n, the $\mathcal{O}_i$, and the implicit ν by "gathering" constraints between the ν-values of the concepts and roles occurring in $\mathcal{O}$ in an undirected graph G. Let $\mathsf{sub}(\mathcal{O})$ be the set of concepts (atomic or complex) occurring in $\mathcal{O}$. The graph G has one node for each concept in $\mathsf{sub}(\mathcal{O})$ or individual occurring in $\mathcal{O}$, and two nodes r_0, r_1 per role in $\mathcal{O}$. The edges represent the constraints imposed by Definition 1. For example, given the axioms $\alpha = A \sqsubseteq \exists r.B$ and $\beta = B \sqsubseteq B'$, then G has nodes $A, \exists r.B, r_0, r_1, B, B'$, and α, β induce edges $\{A, \exists r.B\}$, $\{\exists r.B, r_0\}$, $\{r_1, B\}, \{B, B'\}$, representing cases 8 and 5 of Definition 1. The edges $\{A, \exists r.B\}$ and $\{B, B'\}$ are labelled α and β, respectively. Now G has 2 connected components (CCs): G_1 with nodes $A, \exists r.B, r_0$ and label α; G_2 with r_1, B, B' and β. Hence $\mathbb{O} = (\{\alpha\}, \{\beta\})$. The 2-numbering follows from membership in the G_i: $A, \exists r.B$ are 1-concepts, r is a 12-role, and B, B' are 2-concepts.

This procedure is given by the main routine `partition(`$\mathcal{O}$`)` of Algorithm 1. It first creates the graph G and then adds all edges induced by the structure of the concepts (subroutine `addSubConceptEdges`) and axioms (`addAxiomEdges`) in $\mathcal{O}$ to G. For each role R, both subroutines use the notation R_i, which equals r_i if $R = r$ and r_{1-i} if $R = r^-$, for $i = 0, 1$. Additionally, `addAxiomEdges` labels, for each axiom α, one of the created edges with α. Next, the CCs of G are determined (e.g., via breadth-first search). Finally, the partitioning is read off the axiom labels in the CCs. Not all CCs need to be labelled with some axiom: e.g., if $B \sqsubseteq B'$ is omitted from the above example, we will get the same G_1, G_2, but G_2 will not contain any axiom label. Therefore, the partitioning of $\mathcal{O}$ is determined using only those CCs with ≥ 1 axiom label. We revisit this case below.

4.1 Correctness

Now we want to show that the algorithm returns a combined ontology $\mathbb{O}$ that is compatible with the input ontology $\mathcal{O}$. With Theorem 9 we also have that $\mathcal{O}$ and $\mathbb{O}$ are equivalent, provided that $\mathcal{O}$ is domain-independent.

Theorem 10. *If* $\mathbb{O} =$ `partition(`$\mathcal{O}$`)`*, then* $\mathbb{O}$ *is a* ν*-ontology for some* ν*, and* $\mathcal{O} \sim \mathbb{O}$.

Proof. $\mathcal{O} \sim \mathbb{O}$ follows directly from lines 4–8 of `partition`. For the existence of ν, let $G = (V, E, L)$ be the graph created in lines 1–4 of `partition`, $G_1, \ldots, G_n$ the CCs with ≥ 1 axiom label and $G_{n+1}, \ldots, G_m$ with $m \geq n$ the remaining CCs. Let ν be a function assigning a number $\leq n$ to each node in G and each axiom in $\mathcal{O}$:

- for every $i \leq n$: $\nu(x) = i$ for all nodes x in G_i
- for every $i > n$: fix some $j \leq n$ and let $\nu(x) = j$ for all nodes in x in G_i
- for all edges $\{x, y\}$ in G_i with $L(\{x, y\}) = \alpha$: $\nu(\alpha) = i$

ν contains an n-numbering of Σ if we additionally set:

- for all roles $r \in \Sigma_R$: $\nu(r) = (\nu(r_0), \nu(r_1))$

Algorithm 1: Partitioning an ontology $\mathcal{O}$

1 **Function** `partition`$(\mathcal{O})$**:**
  input : $\mathcal{O}$ with signature Σ
  output : ν-ontology $\mathbb{O}$
2 $V \leftarrow \{C \mid C \in \mathsf{sub}(\mathcal{O})\} \cup \Sigma_1 \cup \{r_0, r_1 \mid r \in \Sigma_R\}$; $E \leftarrow \emptyset$; $L \leftarrow \emptyset$; $G \leftarrow (V, E, L)$
3 **forall** $C \in \mathsf{sub}(\mathcal{O})$ **do** `addSubConceptEdges`(G, C)
4 **forall** $\alpha \in \mathcal{O}$ **do** `addAxiomEdges`(G, α)
5 $\{G_1, \ldots, G_n\} \leftarrow$ all connected components (CCs) of G with ≥ 1 axiom label
6 **forall** $i \leq n$ **do** $\mathcal{O}_i \leftarrow \{\alpha \mid L(v, v') = \alpha$ for some edge (v, v') in $G_i\}$
7 $\mathbb{O} \leftarrow (\mathcal{O}_1, \ldots, \mathcal{O}_n)$
8 **return** $(\mathbb{O})$

9 **Function** `addSubConceptEdges`(G, C)**:**
10 **switch** C **do**
11 **case** $\neg D$ **do** $E \leftarrow E \cup \{C, D\}$
12 **case** $D \sqcap F$ **do** $E \leftarrow E \cup \{\{C, D\}, \{C, F\}\}$
13 **case** $\geqslant m\, R.D$ **do** $E \leftarrow E \cup \{\{C, R_0\}, \{R_1, D\}\}$
14 **case** $\exists R.\mathsf{Self}$ **do** $E \leftarrow E \cup \{\{C, R_0\}, \{C, R_1\}\}$
15 **case** $\{a\}$ **do** $E \leftarrow E \cup \{\{C, a\}\}$

16 **Function** `addAxiomEdges`(G, α)**:**
17 **switch** α **do**
18 **case** $C \sqsubseteq D$ *or* $C \equiv D$ **do** $E \leftarrow E \cup \{C, D\}$; $L(C, D) \leftarrow \alpha$
19 **case** $R \sqsubseteq S$, $R \equiv S$ *or* $\mathsf{Disjoint}(R, S)$ **do**
20 $E \leftarrow E \cup \{\{R_0, S_0\}, \{R_1, S_1\}\}$; $L(R_0, S_0) \leftarrow \alpha$
21 **case** $R \circ S \sqsubseteq T$ **do** $E \leftarrow E \cup \{\{R_1, S_0\}, \{R_0, T_0\}, \{S_1, T_1\}\}$; $L(R_0, T_0) \leftarrow \alpha$
22 **case** $C(a)$ **do** $E \leftarrow E \cup \{C, a\}$; $L(C, a) \leftarrow \alpha$
23 **case** $R(a, b)$ **do** $E \leftarrow E \cup \{\{a, R_0\}, \{R_1, b\}\}$; $L(a, R_0) \leftarrow \alpha$
24 **case** $a \approx b$ *or* $a \not\approx b$ **do** $E \leftarrow E \cup \{\{a, b\}\}$; $L(a, b) \leftarrow \alpha$

It remains to show that ν respects Definition 1.

1. ν is a n-numbering of $\Sigma = \mathsf{sig}(\mathcal{O})$.
2. ν on arbitrary concepts is an extension of ν on Σ that respects Definition 1, i.e., every $C \in \mathsf{sub}(\mathcal{O})$ is a $\nu(C)$-concept. This is an easy induction on the structure of C.
   - $C = \neg D$: By line 11 of of the algorithm G contains the edge $\{C, D\}$. Hence C, D are in the same CC. Then $\nu(C) = \nu(D)$.
   - $C = \geqslant m\, R.D$: By line 13, G contains the edges $\{C, R_0\}$, $\{R_1, D\}$. Hence C, R_0 are in the same CC, and so are R_1, D. Then $\nu(C) = \nu(R_0) = i$, $\nu(D) = \nu(R_1) = j$ and $\nu(R) = (i, j)$.
   - All other cases are analogous.
3. $\nu(\alpha)$ respects the second part of Definition 1, i.e., every $\alpha \in \mathcal{O}$ is a $\nu(\alpha)$-axiom. This is an easy case distinction on the type of α.
   - $\alpha = C \sqsubseteq D$: By line 18, G contains the edge $\{C, D\}$ with $L(\{C, D\}) = \alpha$. Hence C, D and $\{C, D\}$ are in the same CC and $\nu(C) = \nu(D) = \nu(\alpha)$.
   - All other cases are anologous. ❏

4.2 Maximality

The result of the algorithm is the maximal $\mathbb{O}$ with $\mathcal{O} \sim \mathbb{O}$ in the following sense. Let $\mathbb{O} = (\mathcal{O}_1, \ldots, \mathcal{O}_n)$, $\mathbb{O}' = (\mathcal{O}'_1, \ldots, \mathcal{O}'_m)$ be a ν- and a ν'-ontology with $\bigcup_{i \leq n} \mathcal{O}_i = \bigcup_{i \leq m} \mathcal{O}'_i$. We write $\mathbb{O} \preceq \mathbb{O}'$ ("$\mathbb{O}$ is at least as coarse as $\mathbb{O}'$") if for every $\mathcal{O}_i$ there is an $\mathcal{O}'_i$ s.t. $\mathcal{O}_i \subseteq \mathcal{O}'_i$. In other words: if $\mathbb{O} \preceq \mathbb{O}'$ then every $\mathcal{O}'_i$ is the union of one or several $\mathcal{O}_i$.

Theorem 11. *If* $\mathbb{O} = \texttt{partition}(\mathcal{O})$ *then, for every* $\mathbb{O}'$ *with* $\mathcal{O} \sim \mathbb{O}'$*, we have* $\mathbb{O} \preceq \mathbb{O}'$.

Proof. Let $G = (V, E, L)$ the graph created in lines 1–4 of `partition` and $\mathbb{O} = (\mathcal{O}_1, \ldots, \mathcal{O}_n)$; furthermore let $\mathbb{O}' = (\mathcal{O}'_1, \ldots, \mathcal{O}'_m)$ be a ν'-ontology. It suffices to show:

$$\text{For all } i \leq n \text{ and all } \alpha, \beta \in \mathcal{O}_i\text{: } \nu'(\alpha) = \nu'(\beta) \qquad (*)$$

To prove $(*)$ we will first show an auxiliary property, which involves the obvious extension of ν' to all nodes in G; in particular, if r is a role name with $\nu'(r) = (i, j)$, then $\nu'(r_0) := i$ and $\nu'(r_1) := j$.

Claim 1. *For every* $x, y \in V$*: if* x, y *are in the same CC of* G*, then for all numberings* ν' *of* Σ *we have* $\nu'(x) = \nu'(y)$.

Claim 1 is proven via induction on the distance between x, y in their CC. The base case $x = y$ is trivial. For the induction step, suppose that the claim holds for x, x' (by IH) and there is an edge between x' and y. Now $\nu'(x') = \nu'(y)$ can be shown via a straightforward case distinction on the creation of the edge in the functions `addSubConceptEdges` and `addAxiomEdges` together with Definition 1. Hence $\nu'(x) = \nu'(y)$.

We can now prove $(*)$ proceeding by axiom types and analysing the respective cases of `addAxiomEdges`. We just show one case; the remaining ones are very similar.

- $\alpha = C_1 \sqsubseteq D_1$ and $\beta = C_2 \sqsubseteq D_2$: Since α, β are in the same CC, then so are C_1, C_2 (by construction of ν and G). Claim 1 implies $\nu'(C_1) = \nu'(C_2)$ and by Definition 1: $\nu'(\alpha) = \nu'(\beta)$. ❑

4.3 Complexity

Let $\mathcal{O}$ be the input ontology, and $G = (V, E, L)$ the graph created in lines 1–4 of `partition`. Set $k = |\mathcal{O}|$, $\ell = |\mathsf{sub}(\mathcal{O})|$ and $m = |\Sigma|$. We first consider the size of the created graph. The number of nodes is restricted by $\ell + 2m$ (2 nodes per role). It is easy to see that `addAxiomEdges` is called k-times and each call adds at most 3 edges. `addSubConceptEdges` adds at most 2 edges and is called ℓ times. Hence the number of edges is limited by $3k + 2\ell$. Each call to `addAxiomEdges` and `addSubConceptEdges` is in constant time. The connected components of G can be calculated using breadth-first search in $O(|V| + |E|) = O(k + \ell + m)$ [13]. The components with ≥ 1 axiom label can be found going through all edges in $O(|E|) = O(\ell + m)$. To collect the axioms for each component in line 5–6 every edge needs to be checked once. Therefore, this step is also in $O(|E|) = O(\ell + m)$. Altogether the `partition` algorithm takes linear time, limited by $O(k + \ell + m)$, in contrast to [6, Theorem 7.2], which is quadratic.

4.4 Discussion

We conclude this section with remarks concerning the implicit labelling of terms in the input ontology, the treatment of $\top$, and the deterministic character of our algorithm.

`partition`($\mathcal{O}$) creates a ν-ontology $\mathbb{O}$ with $\mathcal{O} \sim \mathbb{O}$. The corresponding numbering ν is induced by the CCs of the created graph G. As we have already pointed out in the description of the algorithm, some CCs may not contain any axiom label. In this case the numbering of the "unlabelled" CCs is arbitrary, as seen in the proof of Theorem 10. If a unique numbering is required, e.g., in order to assign "home components" in $\mathbb{O}$ to the terms in $\mathcal{O}$, then user intervention is required: suppose $\mathcal{O} = \{A \sqsubseteq \exists r.B,\ A' \sqsubseteq \exists r'.B\}$. Then there are 3 CCs: G_1 with nodes A, r_0; G_2 with A', r'_0; G_3 with r_1, r'_1, B. Now G_3 is not axiom-labelled, and so r_1, r'_1, B could be assigned to either G_1 or G_2. Only the user can make that decision, and they need to know the respective subdomains: if the axioms are rewritten into $\texttt{HappyChild} \sqsubseteq \exists\texttt{hasPet}.\texttt{Puppy}$ and $\texttt{HappyDog} \sqsubseteq \exists\texttt{hasChild}.\texttt{Puppy}$, then G_3 should be merged with G_1.

Algorithm 1 does not treat $\top$ explicitly. The required additions are straightforward but, from a theoretical point of view, unnecessary since $\top$ can be rewritten as $A \sqcup \neg A$, as usual. In that case, a *fresh* concept name A should be used for each occurrence of $\top$; Otherwise nodes and the corresponding subdomains might be spuriously connected, e.g., $\texttt{hasToy}_1$ and $\texttt{hasFood}_1$ if $\mathcal{O}$ contains $\texttt{HappyChild} \sqsubseteq \exists\texttt{hasToy}.\top$ and $\texttt{HappyDog} \sqsubseteq \exists\texttt{hasFood}.\top$

Compared with the previous algorithms [6,9,10], neither `partition` nor its subroutines make any nondeterministic choices. In particular, the computed partition is unique up to permutation of the components because the generated graph G does not depend on the order in which concepts and axioms are traversed. This last property is much less obvious in the previous algorithms, which contain a nondeterministic choice without a rigorous argument that the result is still deterministic (and we did observe a nondeterministic behaviour of the prototype implementation).

5 Extensions to the Language

It is straightforward to add datatypes and keys. For datatypes this is shown in [6,9,10]. In essence, ν-interpretations require an additional component where all datatypes are interpreted; data properties are link relations directed towards the datatype component. Definitions, proofs, and algorithm require straightforward additional cases for concepts and axioms involving datatypes. Keys [25] are captured by one dedicated axiom type, whose semantics guarantees domain-independence. It suffices to add the respective case to definitions and algorithm. For details, see the appendix of the technical report.

As remarked after Theorem 8, the universal role u would compromise the syntactic characterisation of domain-independence and thus the link between compatibility and equivalence. This problem does not occur with the $\top$ concept, as discussed in §4.4.

We can pinpoint even more exactly the limits of partitioning based on $\mathcal{E}$-connections: if role negation and disjunction were available, then $u = r \sqcup \neg r$ for a fresh role name r. The "culprit" for the problem with $C = \exists u.A$ as described after Theorem 8 is role negation; in fact, that argument remains valid if we instead consider $C = \exists\neg r.A$. Negated roles compromise only the syntactic characterisation of domain-independence and thus

the link between $\sim$ and $\approx$; $\mathcal{E}$-connections themselves should be extensible to first-order (even higher-order) logic.

If we forbid negation, we can even afford positive Boolean operators on roles and allow complex role expressions built using $\sqcap$, $\sqcup$, $\circ$: those are always domain-independent and can thus safely be used in number restrictions and all role axioms. Hence $\mathcal{E}$-partitions can easily be extended to encompass DLs that allow a more general use of role composition [22] or (positive) Boolean roles [26].

6 Conclusions and Future Work

We have extended the original approach underlying $\mathcal{E}$-partitions in [6,9,10] to all of OWL 2 except the universal role (which cannot be accommodated, as shown). We have presented a new linear-time algorithm for computing the maximal $\mathcal{E}$-connection that is syntactically compatible (and, assuming domain-independence, equivalent) with the input ontology. The output of our algorithm does not depend on the traversal order of axioms and concepts. En route we have introduced a simplified notation. We have further shown that theory and algorithm extend to expressive operators on roles considered in the literature, and identified role negation as not amenable to this approach.

For future work, we expect a straightforward implementation of our algorithm in the OWL API[5] [14] and as a Protégé[6] [24] plugin, as a basis for experiments on a representative up-to-date ontology corpus such as [21]. We conjecture that existing ontologies generally decompose well when allowing slight deviations from (syntactic) compatibility, to circumvent the notorious problematic modelling patterns, see §1. The implementation will also enable a comparison with other decomposition approaches, such as atomic decomposition [11]. We furthermore plan to revisit module extraction, extending the existing procedure in [10] to arbitrary $\mathcal{E}$-connections, independently of a specific partitioning algorithm. Given the linear runtime of our algorithm, module extraction might even compete in performance with syntactic locality [7,32]. Finally, the "culprit" of role negation identified in §5 suggests the hypothesis that the partitioning approach can be transferred to logics that allow only unary negation, such as frontier-one existential rules [4] or even the unary negation fragment of first-order logic, UNFO [31].

Acknowledgement We thank the anonymous reviewers for their helpful comments.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995), `http://webdam.inria.fr/Alice/`
2. Amato, F., De Santo, A., Moscato, V., Persia, F., Picariello, A., Poccia, S.R.: Partitioning of ontologies driven by a structure-based approach. In: Proc. of ICSC-15. pp. 320–323. IEEE Computer Society (2015)
3. Baader, F., Lutz, C., Sturm, H., Wolter, F.: Fusions of description logics and abstract description systems. J. Artif. Intell. Res. **16**, 1–58 (2002)

[5] `http://owlcs.github.io/owlapi/`
[6] `https://protege.stanford.edu/`

4. Baget, J., Leclère, M., Mugnier, M., Salvat, E.: Extending decidable cases for rules with existential variables. In: Proc. of IJCAI-09. pp. 677–682 (2009)
5. Bao, J., Voutsadakis, G., Slutzki, G., Honavar, V.: Package-based description logics. In: Stuckenschmidt et al. [28], pp. 349–371
6. Cuenca Grau, B.: Combination and integration of ontologies on the semantic web. Ph.D. thesis, Universidad de Valencia (2005)
7. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Extracting modules from ontologies: A logic-based approach. In: Stuckenschmidt et al. [28], pp. 159–186
8. Cuenca Grau, B., Parsia, B., Sirin, E.: Ontology integration using $\mathcal{E}$-Connections. In: Stuckenschmidt et al. [28], pp. 293–320
9. Cuenca Grau, B., Parsia, B., Sirin, E., Kalyanpur, A.: Automatic partitioning of OWL ontologies using $\mathcal{E}$-Connections. In: Proc. of DL-05. CEUR-WS.org, vol. 147 (2005)
10. Cuenca Grau, B., Parsia, B., Sirin, E., Kalyanpur, A.: Modularity and web ontologies. In: Proc. of KR-06. pp. 198–209. AAAI Press (2006)
11. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: Atomic decomposition. In: Proc. of IJCAI-11. pp. 2232–2237 (2011)
12. Gatens, W., Konev, B., Wolter, F.: Module extraction for acyclic ontologies. In: Proc. of WoMO-13. CEUR-WS.org, vol. 1081 (2013)
13. Hopcroft, J.E., Tarjan, R.E.: Efficient algorithms for graph manipulation [H] (algorithm 447). Commun. ACM **16**(6), 372–378 (1973)
14. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web **2**(1), 11–21 (2011)
15. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of KR-06. pp. 57–67. AAAI Press (2006)
16. Klinov, P., Del Vescovo, C., Schneider, T.: Incrementally updateable and persistent decomposition of OWL ontologies. In: Proc. of OWLED-12. CEUR-WS.org, vol. 849 (2012)
17. Konev, B., Lutz, C., Ponomaryov, D., Wolter, F.: Decomposing description logic ontologies. In: Proc. of KR-10. pp. 236–246. AAAI Press (2010)
18. Konev, B., Lutz, C., Walther, D., Wolter, F.: Semantic modularity and module extraction in description logics. In: Proc. of ECAI-08. FAIA, vol. 178, pp. 55–59. IOS Press (2008)
19. Krötzsch, M., Simančik, F., Horrocks, I.: A description logic primer. CoRR **abs/1201.4089** (2012), `http://arxiv.org/abs/1201.4089`
20. Kutz, O., Lutz, C., Wolter, F., Zakharyaschev, M.: $\mathcal{E}$-connections of abstract description systems. Artif. Intell. **156**(1), 1–73 (2004)
21. Matentzoglu, N., Parsia, B.: BioPortal Snapshot 30 March 2017 (data set) (2017), `http://doi.org/10.5281/zenodo.439510`
22. Mosurović, M., Krdžavac, N., Graves, H., Zakharyaschev, M.: A decidable extension of $\mathcal{SROIQ}$ with complex role chains and unions. J. Artif. Intell. Res. **47**, 809–851 (2013)
23. Motik, B., Horrocks, I.: OWL datatypes: Design and implementation. In: Proc. of ISWC-08. LNCS, vol. 5318, pp. 307–322. Springer (2008)
24. Musen, M.A.: The Protégé project: a look back and a look forward. AI Matters **1**(4), 4–12 (2015)
25. Parsia, B., Sattler, U., Schneider, T.: Easy keys for OWL. In: Proc. of OWLED-08EU. CEUR-WS.org, vol. 432 (2008)
26. Schmidt, R.A., Tishkovsky, D.: Using tableau to decide description logics with full role negation and identity. ACM Trans. Comput. Log. **15**(1), 7:1–7:31 (2014)
27. Serafini, L., Tamilin, A.: Composing modular ontologies with Distributed Description Logics. In: Stuckenschmidt et al. [28], pp. 321–347
28. Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.): Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization, LNCS, vol. 5445. Springer (2009)

29. Stuckenschmidt, H., Schlicht, A.: Structure-based partitioning of large ontologies. In: Stuckenschmidt et al. [28], pp. 187–210
30. Suntisrivaraporn, B.: Module extraction and incremental classification: A pragmatic approach for $\mathcal{EL}$ ontologies. In: Proc. of ESWC-08. LNCS, vol. 5021, pp. 230–244. Springer (2008)
31. ten Cate, B., Segoufin, L.: Unary negation. Logical Methods in Computer Science **9**(3) (2013)
32. Tsarkov, D.: Improved algorithms for module extraction and atomic decomposition. In: Proc. of DL-12. CEUR-WS.org, vol. 846 (2012)