Decomposing Description Logic Ontologies

Boris Konev
Liverpool University
United Kingdom

Carsten Lutz
University of Bremen
Germany

Denis Ponomaryov
Institute of Informatics Systems
Novosibirsk, Russia

Frank Wolter
Liverpool University
United Kingdom

Introduction

The purpose of an ontology in knowledge representation is to fix the vocabulary of an application domain and to formally describe the meaning of this vocabulary using a logic-based language. This simple idea has proved to be rather successful, and consequently a considerable number of ontologies have been developed for various application domains. In broad domains such as medicine, ontologies used in practice can be extremely large; as an example, take the medical ontology SNOMED CT that covers almost half a million vocabulary items. Unsurprisingly, the design and maintenance of logical theories of this size poses serious challenges and it has long been a major goal of the KR community to provide support in the form of automated reasoning techniques.

Basic reasoning support for ontology design and maintenance aims to make explicit the structure of an ontology, for example by using classification (computing the subconcept/superconcept hierarchy). This is fundamental for an ontology designer who can easily lose track of the overall structure of an ontology—especially when it is constructed by multiple designers working in parallel as in the case of SNOMED CT. Making explicit the structure is also essential when an existing ontology has to be re-engineered due to changes in the modeled application domain or to customize it for a novel application—especially when the ontology was designed by somebody else.

In this paper, we consider a way of analyzing the structure of an ontology that aims at making explicit the dependencies among vocabulary items in the ontology. Our approach is based on signature decompositions, a partition of the signature of an ontology (i.e., of the symbols used to describe vocabulary items) into parts that are independent regarding their meaning. Similar kinds of structural analysis of an ontology have already been advocated, e.g. in (d’Aquin et al. 2009). However, all existing approaches are syntax-dependent and do not participate in the decomposition. Thus, the quality of the computed signature decomposition depends on the quality of the representation of the analyzed ontology (when the goal of the analysis may actually be to improve the quality of a poorly organized ontology).

Our aim is to establish the theoretical foundations for a purely semantical approach to signature decompositions that are not syntax-dependent in the above sense. Formally, the basic notion studied in this paper is the following: a partition \( \Sigma_1, \ldots, \Sigma_n \) of the signature of an ontology \( T \) formulated in an ontology language \( \mathcal{L} \) is a signature decomposition of \( T \) in \( \mathcal{L} \) if there are ontologies \( T_1, \ldots, T_n \) formulated in \( \mathcal{L} \) such that (i) each \( T_i \) uses only symbols from \( \Sigma_i \) and (ii) the union \( T_1 \cup \cdots \cup T_n \) is logically equivalent to \( T \). This notion has first been proposed by Parikh (1999) and Makinson (2007) in the context of propositional logic and belief revision. There is a close relationship between signature decompositions and approaches to modularization of ontologies that aim at a partition of the axioms (rather than signature) of an ontology into independent parts (Cuenca Grau et al. 2006; Amir and McIlraith 2005; Stuckenschmidt, Parent, and Spaccapietra 2009). Again, however, all existing approaches are syntax-dependent and aim at partitioning the existing axiomatization. We emphasize that the ontologies \( T_1, \ldots, T_n \) used in the definition of signature decompositions need not be subsets of the original ontology \( T \). Moreover, as we are interested in decompositions of signatures, we only demand the existence of these ontologies, but do not insist they are explicitly computed.

In many cases, the initial version of signature decompositions defined above should be expected to yield signature decompositions that are too coarse to be informative. To see this, consider a description logic (DL) ontology \( T \) that consists of the axioms \( \alpha = (\text{Car} \sqsubseteq \exists \text{has_part}.Tire) \) and \( \beta = (\text{Ship} \sqsubseteq \exists \text{has_part}.\text{Deck}) \). It is not difficult to show that, due to the use of the role \( \text{has_part} \), the only decomposition of \( T \) consists of only one set that contains the whole signature. From an ontology design perspective, though, the ontology \( T \) contains cars and ships as two separate subject areas that should not be ‘merged’ due to using the general-purpose role \( \text{has_part} \) that, intuitively, does not belong to any specific subject area. From a logical viewpoint, \( \text{has_part} \) thus behaves like a logical symbol much like the equality symbol or the symbol \( \bot \) for contradiction. This example suggests to generalize the initial version of signature decomposition by adding a set of symbols \( \Delta \) that do not induce dependencies and do not participate in the decomposition.
Formally, a signature $\Delta$-decomposition is defined just like a signature decomposition, except that each ontology $T_i$ is allowed to use symbols from $\Sigma_i$ and $\Delta$. This generalization was first proposed by Ponomaryov (2008). In practice, it may not be easy to determine a suitable $\Delta$. In fact, we do not expect signature decompositions to be a push-button technique, but rather envision an iterative and interactive process of understanding and improving the structure of an ontology, where the designer repeatedly chooses sets $\Delta$ and analyzes the impact on the resulting decomposition.

It is important to observe that the notion of a signature decomposition, both with and without the set $\Delta$, depends on the language $\mathcal{L}$ used to formulate the ontologies $T_1, \ldots, T_n$ that realize the signature decomposition (henceforth called realizations). In principle, this is a point of concern as it may not be clear which language $\mathcal{L}$ is appropriate here; for example, when decomposing an ontology $T$ given in a DL, one might expect more fine-grained decompositions if $\mathcal{L}$ is a more expressive logic or even second-order logic (SO) compared to when $\mathcal{L}$ is again a DL. Therefore, the first aim of this paper is to study in how far decompositions of DL ontologies depend on the language for the realizations. Fortunately, it turns out that for many standard DLs, decompositions of TBoxes do not depend on whether one uses SO or the DL for realizations. The main tool for proving this and related results is establishing the split interpolation property, a type of interpolation that has not yet been investigated in the context of ontologies.

In general, one may expect that there can be many distinct and incomparable signature decompositions of a given ontology $T$. This is another point of concern because facing a large number of incomparable decompositions is likely to be confusing rather than helpful for an ontology designer. Therefore and since finer decompositions are clearly more informative than coarser ones, one would ideally like to have a unique finest decomposition to work with. Thus, the second aim of this paper is to investigate when unique finest decompositions exist. Fortunately, we can use split interpolation to show that this is the case for many standard DLs.

Finally, we provide a first analysis of the complexity of some computational problems related to signature decompositions in DL ontologies. We show that for many expressive DLs, these problems are not harder than standard reasoning. Given that there is a very close connection between signature decompositions on the one hand, and computationally very expensive notions such as conservative extensions and uniform interpolation on the other hand, this result is rather surprising. We also show that in the lightweight description logic DL-Lite, signature decompositions can typically be computed in polynomial time. For the lightweight DL $\mathcal{EL}$, we establish the same result for some restricted, but natural cases.

Most proofs have been moved to the appendix of the extended version of this abstract, to be found at http://www.csc.liv.ac.uk/~frank/publ/kr10.pdf

### Preliminaries

Let $\mathcal{N}_C$, $\mathcal{N}_R$, and $\mathcal{N}_I$ be countably infinite and mutually disjoint sets of concept names (unary predicates), role names (binary predicates), and individual names. We use $\mathcal{N}_C$, $\mathcal{N}_R$, and $\mathcal{N}_I$ as the vocabulary for second-order logic (SO), first-order logic (FO), and a variety of DLs. More precisely, we consider SO (and FO) with equality, the predicates from $\mathcal{N}_C \cup \mathcal{N}_R$ and constants from $\mathcal{N}_I$.1 Matching this vocabulary, second-order quantification is over set variables and binary relation variables. We use $\mathcal{T} \subseteq \mathcal{S}$ and $\mathcal{T} \subseteq_{fin} \mathcal{S}$ to denote that $\mathcal{T}$ is a set, respectively finite set, of SO-sentences; we write $\mathcal{T} \models \varphi$ if $\varphi$ is an SO-sentence that is a consequence of $\mathcal{T}$. A set $\mathcal{T} \subseteq \mathcal{S}$ is satisfiable iff $\mathcal{T}$ has a model. Two sets $\mathcal{T}_1 \subseteq \mathcal{S}$ and $\mathcal{T}_2 \subseteq \mathcal{S}$ are equivalent, in symbols $\mathcal{T}_1 \equiv \mathcal{T}_2$, if they have the same models or, equivalently, if $\mathcal{T}_1 \models \varphi$ for all $\varphi \in \mathcal{T}_2$ and vice versa. We sometimes write $\mathcal{T}_1 \models \mathcal{T}_2$ as shorthand for ‘$\mathcal{T}_1 \models \varphi$ for all $\varphi \in \mathcal{T}_2$’. The signature $\text{sig}(\varphi)$ of an SO-formula is the set of all predicate and constant symbols (except equality) used in $\varphi$. This notion is lifted to sets of sentences in the obvious way. A fragment of second-order logic is simply a subset $\mathcal{L} \subseteq \mathcal{S}$.

Description logics can be viewed as fragments of FO. DL concepts are formed starting from concept names by inductively applying concept constructors such as those shown in the upper part of Figure 1. The choice of different constructors gives rise to different DLs. In the figure, we have marked the constructors of the basic DLs $\mathcal{EL}$ and $\mathcal{ALC}$ and assigned to each additional constructor a letter that allows the systematic appellation of extended DLs. The extension $\mathcal{I}$ is with a role constructor for inverse roles, not a concept constructor. When $\mathcal{I}$ is present, inverse roles can be used inside existential restrictions, number restrictions $\exists \$ and role hierarchies $\mathcal{H}$. For details we refer the reader to (Baader et al. 2003).

The semantics of SO (and, therefore, of FO and DLs) is based on interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty set and $\cdot^\mathcal{I}$ is a function that assigns a subset $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ to each $A \in \mathcal{N}_C$, a relation $r^\mathcal{I}$ over $\Delta^\mathcal{I}$ to each $r \in \mathcal{N}_R$, and an element $a^\mathcal{I} \in \Delta^\mathcal{I}$ to each $a \in \mathcal{N}_I$. This also allows us to define the extension $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$ of a DL concept $C$ by the standard inductive translation of $C$ into an FO-formula with one free variable $x$ as shown in Figure 1.

![Figure 1: Standard translation](image-url)
a TBox or ontology, which is a finite set of concept inclusions (CL) \( C \subseteq D \), where \( C, D \) are concepts. An interpretation satisfies a CL \( C \subseteq D \) (written \( I \models C \subseteq D \)) iff \( C^I \subseteq D^I \) and a TBox \( I \models T \) (written \( I \models T \)) if \( I \models C \subseteq D \) for all \( C \subseteq D \in T \). In the presence of role hierarchies (indicated by the letter \( H \)), TBoxes can also include role inclusions \( r \subseteq s \) whose semantics can be found in Figure 1. We will typically not distinguish between DL concepts (resp. TBoxes) and their FO translations. In particular, we often regard DL-TBoxes as finite sets of FO-sentences (and thus SO-sentences).

**Signature Decomposition**

We introduce and illustrate the basic notion of this paper and identify some of its fundamental properties.

**Definition 1 (Signature Decomposition)** Let \( T \subseteq_{fin} SO \) and \( \Delta \subseteq \text{sig}(T) \) and \( \mathcal{L} \subseteq SO \). A partition \( \Sigma_1, \ldots, \Sigma_n \) of \( \text{sig}(T) \setminus \Delta \) is called a signature-\( \Delta \)-decomposition of \( T \) in \( \mathcal{L} \) if there are \( T_1, \ldots, T_n \subseteq \mathcal{L} \) such that

- \( \text{sig}(T_i) \subseteq \Sigma_i \cup \Delta \) for \( 1 \leq i \leq n \);
- \( T_1 \cup \cdots \cup T_n \equiv T \).

In this case, we say that \( T_1, \ldots, T_n \) realize the signature-\( \Delta \)-decomposition \( \Sigma_1, \ldots, \Sigma_n \) in \( \mathcal{L} \).

For simplicity, we will often speak only of decompositions instead of signature decompositions. When \( \Delta = \emptyset \), we simply drop it and speak only of signature decompositions. For any \( T \) and \( \Delta \), there exists at least one \( \Delta \)-decomposition, namely the trivial decomposition consisting only of the single set \( \text{sig}(T) \setminus \Delta \). We call a partition \( \Sigma_1, \ldots, \Sigma_n \) finer than the partition \( \Pi_1, \ldots, \Pi_m \) if they are distinct and for every \( i \leq m \) there exist \( j_1, \ldots, j_k \leq n \) such that \( \Pi_j = \bigcup_{i \leq k} \Sigma_i \).

**Example 2** Let \( T \) be the TBox consisting of

- \( \alpha_1 = (\text{Ball} \sqsubseteq \text{Physical Object}), \alpha_2 = (\text{Table} \sqsubseteq \text{Object}), \alpha_3 = (\text{Ball} \sqsubseteq \exists \text{has colour}, \text{T}), \alpha_4 = (\text{Table} \sqsubseteq \exists \text{has colour}, \text{T}), \alpha_5 = (\text{Orange Ball} \sqsubseteq \text{Ball}). \)

For any \( \Delta = \emptyset \), \( \Delta = \{\text{Physical object}\} \) and \( \Delta = \{\exists \text{has colour}\} \), there are no non-trivial \( \Delta \)-decompositions of \( T \) because, intuitively, Ball and Table are connected independently via both Physical Object and has colour. In many contexts, one would not regard this as a relevant dependency between the two terms. In fact, for \( \Delta = \{\text{Physical object}, \exists \text{has colour}\} \) the finest \( \Delta \)-decomposition of \( T \) is \( \{\text{Ball}, \text{Orange Ball}\}, \{\text{Table}\}, \) realized by \( \{\alpha_1, \alpha_3, \alpha_5\} \) and \( \{\alpha_2, \alpha_4\} \).

One way to extend \( T \) such that Ball and Table are separated already when choosing \( \Delta = \{\exists \text{has colour}\} \) is to add \( \alpha_6 = (\exists \text{has colour}, \text{T} \sqsubseteq \text{Object}). \)

In the resulting \( T' \), the axioms \( \alpha_1, \alpha_2 \) become redundant and the finest \( \Delta \)-decomposition is \( \{\text{Physical object}, \text{Table}, \{\text{Ball}, \text{Orange Ball} \}\} \), realized by \( \{\alpha_3, \alpha_5\} \) and \( \{\alpha_4\} \).

Finally, note that Orange Ball and Ball cannot be separated in a non-trivial way because one would have to extend \( \Delta \) by at least one of the two concepts.

Signature decompositions that can be obtained by analyzing the syntactic form of axioms are a special case of signature decompositions in the sense of Definition 1. The following example shows how such syntactic decompositions can be computed.

**Example 3 (Syntactic decomposition)** Let \( T \subseteq_{fin} SO \) and \( \Delta \subseteq \text{sig}(T) \). There always exists a (unique) finest \( \Delta \)-decomposition \( \Sigma_1, \ldots, \Sigma_n \) that is realized by subsets \( T_1, \ldots, T_n \) of \( T \). We denote this \( \Delta \)-decomposition by \( \text{sd}e\text{co}_\Delta(T) \) and call it the syntactic \( \Delta \)-decomposition of \( T \). To compute \( \text{sd}e\text{co}_\Delta(T) \), let \( T_\Delta \) be the set of \( \alpha \in T \) with \( \text{sig}(\alpha) \subseteq \Delta \). Define an equivalence relation \( \sim_\Delta \) on \( T \setminus T_\Delta \) by setting \( \alpha \sim_\Delta \beta \) if there exist \( \alpha_0, \ldots, \alpha_n \) such that \( \alpha = \alpha_0, \beta = \alpha_n \) and \( \text{sig}(\alpha_i) \cap \text{sig}(\alpha_{i+1}) \subseteq \Delta \) for \( i < n \). Denote by \( T_\Delta \) the \( \sim_\Delta \)-equivalence class of \( \alpha \). Then it is easy to see that \( \{\text{sig}(T_\Delta) \setminus \Delta | \alpha \in T \setminus T_\Delta \} \) coincides with \( \text{sd}e\text{co}_\Delta(T) \) and is realized by \( \{T_\alpha \cup T_\Delta \mid \alpha \in T \setminus T_\Delta \} \). It follows, in particular, that \( \text{sd}e\text{co}_\Delta(T) \) can be computed in poly-time.

We now establish some basic properties of decompositions in SO used throughout the paper. As announced in the introduction, decompositions in SO play a special role in this paper as they are easy to work with and turn out to be equivalent to decompositions in many standard DLs. To formulate SO decompositions succinctly, we write \( \exists \alpha \varphi \) to denote \( \exists P.\varphi[P/\alpha] \), where either \( \alpha \) is a predicate and \( P \) a fresh predicate variable of the same arity as \( \alpha \), or \( \alpha \) is an individual constant and \( P \) a fresh individual variable. Clearly, \( \exists \exists \alpha \varphi \) is shorthand for \( \exists \exists \alpha \exists \varphi \). \( \exists \Sigma \varphi \) is a predicate and \( \Sigma \) is a signature.

**Theorem 4 (Characterization)** Let \( T \subseteq_{fin} SO \) and \( \Delta \subseteq \text{sig}(T) \). A partition \( \Sigma_1, \ldots, \Sigma_n \) of \( \text{sig}(T) \setminus \Delta \) is a signature-\( \Delta \)-decomposition of \( T \) in \( SO \) iff

\[
\{ \exists \Sigma_1, \bigwedge_{\varphi \in T} \varphi, \cdots, \exists \Sigma_n, \bigwedge_{\varphi \in T} \varphi \} \models T
\]

where \( \Sigma_i := \bigcup_{1 \leq j \leq n, j \neq i} \Sigma_j \).

**Proof.** \("\Rightarrow\). Assume that the partition \( \Sigma_1, \ldots, \Sigma_n \) of \( \text{sig}(T) \setminus \Delta \) is a \( \Delta \)-decomposition of \( T \) in \( SO \) realized by \( T_1, \ldots, T_n \). To show that \((*)\) holds, let \( T \) be a model of the left-hand side of \((*)\). Then \( T \) is a model of \( T_\alpha \) for \( 1 \leq i \leq n \): since \( T \models \exists \Sigma_i \bigwedge_{\varphi \in T} \varphi \), there is a model \( J \) of \( T \) that agrees with \( T \) on the interpretation of all symbols from \( \Sigma_i \cup \Delta \): since \( T \models T_1 \cup \cdots \cup T_n \), we have \( J \models T_\alpha \) and due to \( \text{sig}(T_\alpha) \subseteq \Sigma_i \cup \Delta \), it follows that \( T \models J \), as stated. Thus \( T \models T_1 \cup \cdots \cup T_n \) and \( T \models T_1 \cup \cdots \cup T_n \) yields that \( T \) is a model of \( T \), as required.

\("\Leftarrow\). If \((*)\) holds, then \( T \models \{ \exists \Sigma_i, \bigwedge_{\varphi \in T} \varphi \}, 1 \leq i \leq n \), clearly realize \( \Sigma_1, \ldots, \Sigma_n \).

As a consequence of the proof of Theorem 4, for each decomposition \( \Sigma_1, \ldots, \Sigma_n \) in \( SO \), there exists a realization of a canonical (though rather uninformative) form.\(^2\) Clearly, this

\(^2\)We note that a similar proof shows the following purely model-theoretic characterization of \( \Delta \)-decompositions in SO: a partition \( \Sigma_1, \ldots, \Sigma_n \) of \( \text{sig}(T) \setminus \Delta \) is a \( \Delta \)-decomposition of \( T \subseteq_{fin} SO \) iff
canonical form relies on second-order quantifiers and does not exist in (fragments of) FO. As a first application of this result, one can show that there always exists a unique finest $\Delta$-decomposition in SO.

**Theorem 5 (Unique Finest Decomposition)** Let $T \subseteq_{fin} SO$, $\Delta \subseteq \text{sig}(T)$, and let $\Sigma_1, \ldots, \Sigma_n$ and $\Pi_1, \ldots, \Pi_m$ be $\Delta$-decompositions of $T$ in SO. Then the partition $\Sigma_i \cap \Pi_j$ for all $i, j$ with $\Sigma_i \cap \Pi_j \neq \emptyset$ of $\text{sig}(T) \setminus \Delta$ is a $\Delta$-decomposition of $T$ in SO. Thus, there exists a unique finest $\Delta$-decomposition of $T$ in SO.

In the following example, we compute the finest $\Delta$-decomposition in SO of concept hierarchies.

**Example 6** Let $T$ be a finite set of inclusions $A \subseteq B$ between concept names $A, B$. A realization of the unique finest $\Delta$-decomposition in SO of $T$ is obtained by first adding to $T$ all CIs $A \subseteq B$ with $T \models A \subseteq B$ that contain at most one non-$\Delta$ symbol. Now remove from the resulting TBox all $A \subseteq B$ with two non-$\Delta$ symbols for which there exists $D \in T$ with $A \subseteq D, D \subseteq B \in T$, and denote by $T'$ the resulting TBox. It can be shown that $\Delta \text{dec}_{SO}(T')$ is the unique finest $\Delta$-decomposition of $T$ which, in this case, is realized using again a concept hierarchy and no second-order quantifiers.

Although there always is a unique finest decomposition in SO, the theories that realize this (finest) decomposition are generally not uniquely determined. To see this, consider the TBox $T = \{ T \subseteq A \cap B_1 \cap B_2 \}$ and let $\Delta = \{ A \}$. Then the unique finest $\Delta$-decomposition $\{ B_1, B_2 \}$ is realized by both $\{ T \subseteq A \cap B_1 \cap B_2 \}$ and $\{ T \subseteq A \cap B_1 \cap B_2 \}$. Clearly, there are no two sets in these two realizations that are logically equivalent. For many fragments $L$ of SO, uniqueness of realizations $T_1, \ldots, T_n$ can be restored by demanding that all $T_i, T_j$ are $\Delta$-inseparable in $L$, i.e., that $T_i \models \varphi$ iff $T_j \models \varphi$, for all $\varphi \in L$ and with $\text{sig}(\varphi) \subseteq \Delta$. Note that the canonical realization in Theorem 4 has this property.

**Definition 7 (Unique Realizations)** Let $L$ be a fragment of SO. We say that $L$ has unique decomposition realizations (UDR) if for all satisfiable $T \subseteq_{fin} L$ and all finite $L$-realizations $T_1, \ldots, T_n$ and $T'_1, \ldots, T'_n$ of a $\Delta$-decomposition of $T$ such that:  
- $T_i, T_j$ are $\Delta$-inseparable w.r.t. $L$ for $i, j \leq n$ and  
- $T'_i, T'_j$ are $\Delta$-inseparable w.r.t. $L$ for $i, j \leq n$,  
we have $T_i \equiv T'_i$ for all $i \leq n$.

UDR has interesting consequences. For example, if $T_1, \ldots, T_n$ satisfy the conditions of Definition 7, and $L$ has UDR, then one can show that $T$ is a conservative extension of each $T_i$ (i.e., $T_i \models \varphi$ iff $T \models \varphi$ for all $\varphi$ with $\text{sig}(\varphi) \subseteq \text{sig}(T_i)$). Thus, realizations satisfy the basic conditions for logic-based ontology modules as proposed and discussed in (Cuenca Grau et al. 2006; 2008; Konev et al. 2009).

**Theorem 8** SO has UDR.

In this section, we have seen that decompositions in SO have a variety of desirable properties. The aim of the next section is to investigate in how far these are also enjoyed by decompositions in DLs.

**Signature decompositions and split interpolation in DLs**

By definition, if $\mathcal{L}_1$ is a fragment of $\mathcal{L}_2$, then every $\Delta$-decomposition of some $T \in \mathcal{L}_1$ is a $\Delta$-decomposition of $T$ in $\mathcal{L}_2$. In particular, every $\Delta$-decomposition of $T$ in some fragment of SO is a $\Delta$-decomposition of $T$ in SO. In this section, we show that for many DLs the converse implication holds as well and that, therefore, DLs inherit many of the desirable properties of decompositions in SO.

**Definition 9 ($\mathcal{L}$-decompositions = SO-decompositions)** Let $\mathcal{L}$ be a fragment of SO. We say that $\mathcal{L}$-decompositions coincide with SO-decompositions if for every $T \subseteq_{fin} \mathcal{L}$ and every signature $\Delta \subseteq \text{sig}(T)$, the $\Delta$-decompositions of $T$ in $\mathcal{L}$ coincide with the $\Delta$-decompositions of $T$ in SO.

Before we provide methodologies for proving this property for a wide range of DLs, we provide a counterexample showing that ACCO-decompositions do not coincide with SO-decompositions.

**Example 10** Let $\Delta = \emptyset$ and $T$ consist of the ACCO-inclusions

$$\{ a \} \subseteq (\exists r, \neg\{a\}) \cap (\forall r, \neg\{a\}), \quad T \subseteq \{ b \} \sqcup \{ b' \},$$

$$\neg\{a\} \subseteq (\exists r, \{a\}) \cap (\forall r, \{a\}), \quad \{a'\} \subseteq \{a'\}.$$

By the CI $T \subseteq \{ b \} \sqcup \{ b' \}$ each model of $T$ has at most two domain elements. Using the two CIs involving $a$ it is, therefore, easy to see that $T$ axiomatizes the class of two-element interpretations in which $b, b'$ denote distinct elements and $r$ is a symmetric and irreflexive relation that connects the two domain elements. In particular, $T$ “says nothing” about $a$ and $a'$. Thus, the finest $\Delta$-decomposition in SO (and FO) of $T$ is $\{a\}, \{a'\}, \{r\}, \{b, b'\}$. In contrast, one can show that there is no finer $\Delta$-decomposition of $T$ in ACCO than $\Delta \text{dec}_{\mathcal{L}}(T)$ which coincides with $\{a, r\}, \{a'\}, \{b, b'\}$. Another $\Delta$-decomposition of $T$ in ACCO, which is incompatible with $\Delta \text{dec}_{\mathcal{L}}(T)$, is given by $\{a', r\}, \{a\}, \{b, b'\}$. It follows that ACCO TBoxes do not always have a unique finest $\Delta$-decomposition in ACCO.

We now introduce an interpolation property that is not only sufficient to prove that SO-decompositions coincide with $\mathcal{L}$-decompositions but also implies UDR.

**Definition 11 (Split Interpolation)** Let $L$ be a fragment of SO. $(T_1, T_2)$ be two sets of SO-sentences, $\alpha$ an SO-sentence, and $\Delta$ a signature. A pair $(T'_1, T'_2)$ with $T'_i \subseteq L$ for $i = 1, 2$ is called a $\Delta$-split interpolant of $(T_1, T_2)$ and $\alpha$ in $L$ if the following conditions hold:

- $T_i \models T'_i$ for $i = 1, 2$;
\( \text{sig}(T_i) \setminus \Delta \subseteq \text{sig}(T_i) \cap \text{sig}(\alpha) \) for \( i = 1, 2 \);
\( T_1 \cup T_2 \models \alpha \).
\( L \) has the split interpolation property (SIP) if for all \( T_1, T_2 \subseteq L \), all \( \alpha \in L \), and all signatures \( \Delta \) such that
1. \( \text{sig}(T_1) \cap \text{sig}(T_2) \subseteq \Delta \).
2. \( T_1 \cup T_2 \models \alpha \).
3. \( T_1 \) and \( T_2 \) are \( \Delta \)-inseparable w.r.t. \( L \),
there exists a \( \Delta \)-split interpolant of \((T_1, T_2)\) and \( \alpha \) in \( L \).

The main reason for studying split interpolation is the following result.

**Theorem 12** Let \( L \) be a fragment of SO with the SIP. Then
1. \( L \)-decompositions coincide with SO-decompositions.
2. \( L \) has UDR.

In particular, every \( T \subseteq_{fin} L \) has a unique finest \( \Delta \)-decomposition in \( L \).

**Proof.** (Sketch for Point 1) Assume that \( \Sigma_1, \Sigma_2 \) is a \( \Delta \)-decomposition in SO of \( T \). It follows from Theorem 4 that \( \{ \exists \Sigma_2, \exists \Sigma_1, \exists \Sigma_2 \exists \Sigma_1 \Theta \} \models T \). Let \( S_1 \) and \( S_2 \) be the subsets of \( \Sigma_2 \) obtained from \( \Sigma_1 \) by replacing all predicates in \( \Sigma_2 \) and \( \Sigma_1 \), respectively, by fresh predicates. Then \( S_1 \cup S_2 \models T \) and the componentwise union of the \( \Delta \)-split interpolants of \((S_1, S_2)\) and \( \alpha \) in \( L \), \( \alpha \in T \), realizes \( \Sigma_1, \Sigma_2 \) in \( L \).

The proof shows that an algorithm computing \( \Delta \)-split interpolants in \( L \) can be directly employed to construct a realization in \( L \) of a given \( \Delta \)-decomposition. It is beyond the scope of this paper to develop such algorithms. As the focus of this paper is on signature decompositions rather than realizations, we concentrate on proving the SIP and leave the computation of \( \Delta \)-split interpolants for future work.

In FO, it is easy to prove the equivalences of the SIP and the standard Craig Interpolation Property (Parikh 1999; Kourousias and Makinson 2007). Unfortunately, this is not the case for DLs because the proof uses the fact that FO-sentences are closed under Boolean operations and this typically does not hold for DLs (e.g., there does not exist a TBox \( T \) in ALC that is equivalent to \( \neg (T \subseteq A) \)). This also implies that recent results on the existence and computation of Craig interpolants in DL using tableaux are not directly applicable (Seylan, Franconi, and de Bruijn 2009). Nevertheless, in the full paper we prove the following interpolation theorem by extending techniques, and using results from, (Konev et al. 2009; Lubotz and Wolter 2010).

**Theorem 13** The following DLs have SIP: \( \mathcal{E}L \), \( \mathcal{E}LH \), \( \mathcal{A}LC \), \( \mathcal{A}LCQ \), \( \mathcal{A}LCQI \).

We note that the proofs depend on the fact that for the DLs from Theorem 13, the class of models of a TBox is closed under forming disjoint unions. \( \mathcal{A}LCQ \), on the other hand, does not have this property. It is interesting to observe that the addition of role inclusions to \( \mathcal{E}L \) preserves the SIP. This is true for DL-Lite (see the analysis below) as well, but expressive DLs with role inclusions typically do not have the SIP.

Example 14 \( \mathcal{ALCH} \) does not have the SIP. Let \( \Delta = \{ r_1, r_2 \} \), \( \alpha = \forall r_1.A \subseteq \exists r_2.A \), \( T_1 = \{ T \subseteq \exists r_1.T \cap \exists r_2.T \} \), and \( T_2 = \{ s \subseteq \{ r_1 \cup r_2 \} \subseteq \exists s.T \} \). Then \( T_1 \cup T_2 \models \alpha \) but there does not exist a \( \Delta \)-split interpolant of \((T_1, T_2)\) and \( \alpha \) in \( \mathcal{ALCH} \). We note that it remains an open problem whether \( \mathcal{ALCH} \)-decompositions coincide with SO-decompositions.

We now show how the SIP can be restored for expressive DLs with role inclusions and/or nominals by including into \( \Delta \) all role and individual names. To obtain the SIP in the presence of nominals we take, in addition, the \( \Theta \)-operator from hybrid logic (Arenes and ten Cate 2006) (an alternative approach restoring SIP is to admit Boolean TBoxes or, equivalently, the universal role). Given a DL \( L \), we denote by \( L^0 \) the DL obtained from \( L \) by adding the \( \Theta \)-operator as a new concept constructor: if \( a \) is an individual name and \( C \) is an \( L \)-concept, then \( \Theta a.C \) is an \( L^0 \)-concept. In every interpretation \( I \), \( (\Theta a.C)^I = \Delta^I \) if \( a \in C^I \) and \( (\Theta a.C)^I = \emptyset \) otherwise. The following theorem can now be proved by extending results and techniques introduced in (ten Cate 2005; ten Cate et al. 2006).

**Theorem 15** Assume \( L \in \{ \mathcal{ALCH}, \mathcal{ALCHI}, \mathcal{ALCQ}, \mathcal{ALCHO}, \mathcal{ALCHO} \} \). Then \( \Delta \)-split interpolants exist in \( L \) for every \((T_1, T_2)\) in \( L \) and \( \Delta \)-inclusion \( \alpha \) such that
1.–3. from Definition 11 hold and \( \Delta \) contains all role and individual names in \( T_1, T_2, \alpha \).

In particular, for every \( T \in L \) and \( \Delta \) containing all role and individual names in \( T \), \( \Delta \)-decompositions of \( T \) in \( L \) coincide with \( \Delta \)-decompositions of \( T \) in \( L \).

Although in the following section we concentrate on languages in which SO-decompositions coincide with \( \mathcal{L} \)-decompositions, it is straightforward to formulate and prove finer grained results for languages such as those in Theorem 15 in which certain restrictions on \( \Delta \) are required to ensure that SO-decompositions coincide with \( \mathcal{L} \)-decompositions.

**Computing decompositions in expressive DLs**

We now exploit the results of the previous two sections to analyze the computational complexity of the problem of computing, for a given \( T \subseteq_{fin} L \) and \( \Delta \subseteq \text{sig}(T) \), the finest \( \Delta \)-decomposition of \( T \) in \( L \). We confine ourselves to languages \( L \) in which SO-decompositions coincide with \( \mathcal{L} \)-decompositions and, therefore, can assume that unique finest decompositions always exist and coincide with the finest \( \Delta \)-decomposition in SO. In this section, we prove tight complexity bounds for a range of expressive DLs; in the next section we consider lightweight DLs.

It will be convenient to reformulate the problem of computing the finest \( \Delta \)-decomposition as a decision problem. Say that two symbols \( \{ \sigma_1, \sigma_2 \} \subseteq \text{sig}(T) \) are \( \Delta \)-decomposable w.r.t. \( T \) if, and only if, there exists a \( \Delta \)-decomposition \( \Sigma_1, \Sigma_2 \) of \( T \) such that \( \sigma_1 \in \Sigma_1 \) and \( \sigma_2 \in \Sigma_2 \). The problem of deciding \( \Delta \)-decomposability of two symbols w.r.t. \( T \) may be viewed as the decision problem associated with computing the finest \( \Delta \)-decomposition of \( T \): indeed, it
is easy to see that the finest \(\Delta\)-decomposition of \(T\) coincides with the partition of \(\text{sig}(T) \setminus \Delta\) induced by the equivalence relation \(\sim\) defined by setting \(\sigma_1 \sim \sigma_2\) iff \(\{\sigma_1, \sigma_2\}\) are not \(\Delta\)-decomposable w.r.t. \(T\).

**Theorem 16 (Complexity of \(\Delta\)-decomposability)**

Let \(\mathcal{L}\) be \(\mathcal{AECL}, \mathcal{AECL}, \mathcal{AECLQ},\) or \(\mathcal{AECLT}\). The problem of deciding whether two symbols are \(\Delta\)-decomposable w.r.t. \(T\)-Boxes in \(\mathcal{L}\) is \(\text{ExpTIME}\)-complete.

**Proof.** We start with the upper bound. Assume a \(T\)Box \(T\) in \(\mathcal{L}\), a signature \(\Delta \subseteq \text{sig}(T)\), and \(\sigma_1, \sigma_2 \in \text{sig}(T) \setminus \Delta\) are given. Enumerate all (exponentially many) partitions \(\Sigma_1, \Sigma_2\) of \(\text{sig}(T) \setminus \Delta\) such that \(\sigma_1 \in \Sigma_1\) and \(\sigma_2 \in \Sigma_2\). Then \(\sigma_1, \sigma_2\) are \(\Delta\)-decomposable w.r.t. \(T\) if, and only if, at least one these partitions is a \(\Delta\)-decomposition of \(T\). It is thus sufficient to show that the latter problem can be decided in \(\text{ExpTIME}\). Assume \(\Sigma_1, \Sigma_2\) is given. By Theorem 4, \(\Sigma_1, \Sigma_2\) is a \(\Delta\)-decomposition of \(T\) in \(\text{SO}\) (and, therefore, by the \(\text{SIP}\), in \(\mathcal{L}\)) if, and only if,

\[
\left\{ \exists \Sigma_2, \bigwedge_{C \subseteq D \in \mathcal{T}} C \subseteq D, \exists \Sigma_1, \bigwedge_{C \subseteq D \in \mathcal{T}} C \subseteq D \right\} \models T.
\]

This condition can be checked by standard subsumption checking w.r.t. \(\mathcal{L}\)-TBoxes. This problem is in \(\text{ExpTIME}\) (Baader et al. 2003). For the \(\text{ExpTIME}\)-lower bound, observe that a \(T\)Box \(T\) is unsatisfiable iff \(\mathcal{L}\)-Box \(T\) is \(\Delta\)-decomposable w.r.t. \(T\) if, and only if, at least one these partitions is a \(\Delta\)-decomposition of \(T\). Checking unsatisfiability of \(T\)Boxes in \(\mathcal{L}\) is \(\text{ExpTIME}\)-hard (Baader et al. 2003).

Clearly, this proof does not provide a practical method for computing finest decompositions and for expressive DLs we leave this problem for future work. Theorem 16 can be generalized in various directions. In the proof, we did not use any specific properties of \(\mathcal{L}\), except that SO-decompositions coincide with \(\mathcal{L}\)-decompositions and that subsumption and satisfiability are \(\text{ExpTIME}\)-complete. Thus the same proof can be used to show that for any language \(\mathcal{L}\) in which subsumption/satisfiability are at least \(\text{ExpTIME}\)-hard, checking \(\Delta\)-decomposability of two symbols is of the same complexity as subsumption/satisfiability. For languages \(\mathcal{L}\) in which reasoning is strictly less complex than \(\text{ExpTIME}\), the proof does not necessarily work because the enumeration step for the signature partitions requires exponential time already. In particular, we cannot use the proof to establish tractability of \(\Delta\)-decomposability for DLs such as DL-Lite and \(\mathcal{EL}\) in which subsumption/satisfiability is tractable.

**DL-Lite and \(\mathcal{EL}\)**

Our aim is to establish the SIP and investigate the computational properties of decomposition for lightweight DLs. We start with DL-Lite (Calvanese et al. 2009), which is relatively easy, and then consider the much harder case of \(\mathcal{EL}\) (Baader, Brandt, and Lutz 2005).

For simplicity, we present our results for DL-Lite\(_{core}\) and DL-Lite\(_{horn}\), but corresponding results hold as well for more expressive DL-Lite dialects such as DL-Lite\(_{R}\), DL-Lite\(_{\pi}\), and DL-Lite\(_{\pi,horn}\) (Calvanese et al. 2006; Artale et al. 2009).

Recall that basic DL-Lite concepts \(B\) are defined as

\[ B ::= \top \mid \bot \mid A \mid \exists r \mid \exists r^\neg, \]

where \(A\) ranges over \(\mathbb{N}_C\) and \(r\) over \(\mathbb{N}_R\). Denote by \(NB\) the set of basic DL-Lite concepts and their negations. A DL-Lite\(_{core}\)-inclusion has the form \(B_1 \subseteq B_2\), where \(B_1, B_2 \in \mathbb{N}_B\). A DL-Lite\(_{horn}\)-inclusion has the form \(\bigwedge \{ \bigvee B_{i} \subseteq B \}\), where the \(B_i\) and \(B\) are basic DL-Lite concepts.

**Theorem 17**

(1) DL-Lite\(_{core}\) and DL-Lite\(_{horn}\) have the SIP.

(2) For TBoxes in DL-Lite\(_{core}\) and DL-Lite\(_{horn}\) and signatures \(\Delta\), one can compute in polynomial time a realization (in the respective language) of its finest \(\Delta\)-decomposition. Thus, the finest \(\Delta\)-decomposition can be computed in polynomial time as well.

We sketch the algorithms required to prove (2). For DL-Lite dialects without conjunctions on the left hand side of CIs (such as DL-Lite\(_{core}\)) the algorithm is straightforward and almost identical to the algorithm for concept hierarchies in Example 6. Namely, let \(T\) be a DL-Lite\(_{core}\)-TBox. First add to \(T\) all DL-Lite\(_{core}\) CIs \(B_1 \subseteq B_2\) with \(T \models B_1 = B_2\) and containing not more than one non-\(\Delta\)-symbol. Now remove from the resulting TBox all \(B_1 \subseteq B_2\) containing two non-\(\Delta\)-symbols for which there exists \(D \in \mathbb{N}_B\) with \(\text{sig}(D) \subseteq \Delta\) and \(T \models B_1 = D\) and \(T \models D = B_2\). For the resulting TBox \(T'\) one can show that \(\text{sdceo}_\Delta(T')\) coincides with the finest \(\Delta\)-decomposition of \(T\).

For DL-Lite\(_{horn}\) the algorithm is more involved because of conjunctions on the left hand side of CIs. We illustrate the additional step for the role-free fragment of DL-Lite\(_{horn}\) (i.e., for propositional Horn Logic).

**Example 18**

Let \(\Delta = \{D_1, D_2\}\) and

\[ T = \{A_1 \cap A_2 \subseteq B, A_1 \subseteq D_1, A_2 \subseteq D_2, D_1 \cap D_2 \subseteq A_1\}. \]

The interesting CI is \(\alpha_0 = (A_1 \cap A_2)\). Intuitively, \(\{A_1, A_2\}\) is \(\Delta\)-decomposable w.r.t. \(T\) because \(A_1 \cap A_2\) is equivalent to a concept not using \(A_1\) in \(T \setminus \{\alpha_0\}\).

More precisely,

\[ T \setminus \{\alpha_0\} \models (A_1 \cap A_2) \equiv (D_1 \cap A_2). \]

Thus we can replace in \(T\) the CI \(\alpha_0\) by \(D_1 \cap A_2 \subseteq B\). The resulting TBox realizes the partition \(\{A_2, B\}, \{A_1\}\) which can be shown to be the finest \(\Delta\)-decomposition of \(T\). Note that we could have used \(D_1 \cap D_2 \cap A_2\) instead of \(D_1 \cap A_2\).

The idea from Example 18 can be generalised: for each inclusion \(\alpha = (C_0 \subseteq B_0)\) of the DL-Lite\(_{horn}\) TBox \(T\) under consideration one can compute, in polynomial time, some minimal \(\Sigma \subseteq \text{sig}(C_0) \setminus \Delta\) such that

\[ T \setminus \{\alpha\} \models \text{Cons}_T \setminus \{\alpha\}, \Sigma \cup \Delta (C_0) \subseteq C_0, \]

where \(\text{Cons}_T \setminus \{\alpha\}, \Sigma \cup \Delta (C_0)\) is the conjunction of all basic DL-Lite concepts \(B\) constructed from \(\Sigma \cup \Delta \cup \{T, \bot\}\) such that \(T \setminus \{\alpha\} \models C_0 \subseteq B\), and then replaces \(\alpha\) by \(\text{Cons}_T \setminus \{\alpha\}, \Sigma \cup \Delta (C_0) \subseteq B_0\). (In Example 18, one obtains...
Finally, we discuss $\mathcal{EL}$. We have seen already (Theorem 13) that $\mathcal{EL}$ and $\mathcal{ELH}$ have the SIP, thus we focus on its computational properties. Call an $\mathcal{EL}$-TBox $T$ role-acyclic if there does not exist an $\mathcal{EL}$-concept $C$ and role names $r_1, \ldots, r_n$ with $n \geq 1$ such that $T \models C \sqsubseteq \exists r_1, \ldots, \exists r_n, C$. Note that acyclic terminologies such as SNOMED CT satisfy this condition.

**Theorem 19** Let

- $\Delta = \emptyset$ and $T$ be an arbitrary $\mathcal{EL}$-TBox; or
- $\Delta$ arbitrary and $T$ be a role-acyclic TBox.

Then the finest $\Delta$-decomposition of $T$ can be computed in polynomial time.

The proof for arbitrary $\Delta$ and role acyclic $T$ is rather involved and will be sketched in the full paper. Whether this result holds for arbitrary $\mathcal{EL}$-TBoxes remains an open problem. We note that the polytime algorithm we give does not compute a realization of the finest $\Delta$-decomposition in $\mathcal{EL}$ but only a realization in an extension of $\mathcal{EL}$ with simulation quantifiers (a variant of bisimulation quantifiers (Visser 1996) appropriate for $\mathcal{EL}$) and additional concepts and names. The reason is that, in contrast to DL-Lite, minimal $\mathcal{EL}$-TBoxes realizing finest $\Delta$-decompositions in $\mathcal{EL}$ are not always of polynomial size. Let, for example, $T$ consist of $A_i \equiv \exists r_i, A_{i+1} \sqcap \exists s_i, A_{i+1}$, for $0 \leq i < n$, and $A_n \equiv \top$. For $\Delta = \{r_0, \ldots, r_{n-1}, s_0, s_1, \ldots, s_{n-1}\}$ the finest $\Delta$-decomposition of $T$ is $\{A_0\}, \ldots, \{A_n\}$ but any $\mathcal{EL}$-TBox $T_0, \ldots, T_n$ realizing it is of exponential size.

**Conclusion**

We have established the theoretical foundations for a syntax-independent approach to signature decomposition in ontologies. Our investigation has been inspired by previous work in propositional logic, belief revision, and abstract logical calculi (Parikh 1999; Kourousias and Makinson 2007; Ponomaryov 2008). Of course, a semantic approach leads to reasoning services of higher complexity than purely syntactic approaches. Still, the results are quite promising: for lightweight DLs the main reasoning problem is (mostly) still tractable and for expressive DLs it is not harder than subsumption checking. This shows that signature decomposition is computationally much simpler than semantically complete approaches to other modularization tasks such as module extraction, conservative extensions, and forgetting/uniform interpolation (Konev et al. 2009; Cuenca Grau et al. 2008). Future work will include decomposition experiments with existing ontologies and the development of guidelines to determine meaningful $\Delta$'s.

**References**


