Approximation of Ontologies in CASL

Klaus Lüttich
SFB/TR8 – FB3, Universität Bremen, P.O.B. 33 04 40, 28334 Bremen, Germany
e-mail: luettich@informatik.uni-bremen.de

Abstract. In this paper we present methods to generate a Description Logic (DL) theory from a given First Order Logic (FOL) theory, such that each DL axiom is entailed by the given FOL theory. This is obtained by transforming the given FOL formulas. If this method is applied to an ontology specification in FOL, the resulting DL specification is still grounded on the same semantics but clearly weaker than the FOL specification. The benefit of specification in DL is that efficient reasoning procedures are available as implemented in tools such as Racer, Fact++ or Pellet. Such ontologies in DL could be used for knowledge representation systems and the semantic web where efficient reasoning plays a major role. This method can be used to compile a foundational ontology formalized in FOL, like DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering), into DL for use with domain ontologies formalized in DL, or for the development of domain ontologies based on the compiled foundational ontology. These weakening strategies are described using CASL (Common Algebraic Specification Language) and CASL-DL, and will be integrated into HETS (Heterogeneous Tool Set). Furthermore, this paper includes examples from DOLCE.

Keywords. Theory Approximation, Knowledge Compilation, CASL, Description Logic

1. Introduction

Traditionally, Description Logics (DL) [1] or other less expressive formalisms have been used to describe and reason about knowledge in an efficient way. This is clearly needed for applications such as the Semantic Web [2] and natural language processing. However, on the other hand DLs are too weak to formalize a rich axiomatized foundational ontology such as DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [3,4]. Therefore CASL [5] (Common Algebraic Specification Language) and even ModalCASL (an extension of CASL with multi-modalities) will be used for the formalization of DOLCE. That this offers great potential has been previously presented in [6]. One advantage of CASL is the associated tool HETS [7] (Heterogeneous Tool Set). It allows syntax and type checking of CASL and is connected to the semi-automatic theorem prover Isabelle [8] as well as the automatic first-order reasoner SPASS [9].

This paper offers a bridge between rich axiomatizations in FOL and tractable theories in DL. It describes an approximation of FOL theories to DL preserving the semantics. This idea is also called knowledge compilation as the knowledge or semantics of a theory should be kept while the tractability is improved [10,11].

This paper is structured as follows: First we introduce description logic, CASL and CASL-DL, the languages used in the examples. Then we present the idea of knowledge compilation in general, and further we present our approach to approximate FOL with DL. This section is accompanied by an informative example illustrating the method. The paper ends with a perspective on future work.

2. Logical Foundations

In this section we introduce the logical foundations for our approach: First we briefly present Description Logics, then we describe the formal languages CASL and CASL-DL.

2.1. Description Logic

Description Logic (DL) has been developed for efficient knowledge representation and reasoning. Its principle is the distinction between so-called TBoxes and ABoxes. TBoxes provide the terminology of the knowledge base such as hierarchies of concepts and roles, and axioms describing which individuals belong to a concept based on relations to other individuals. These descriptions are formulas with a restricted flow of variables. Furthermore, some predefined datatypes like strings and numbers are available to attach data to individuals with data-valued roles. ABoxes, on the other hand, represent facts about the world and the individuals, by axioms with constants (individuals) [1].

DLs have the benefit that the tractability, decidability, and the complexity of the reasoning systems has been studied very well by now [1]. Another advantage is the great number of automatic reasoners available, such as Racer [12], FaCT++ [13] or Pellet [14].

2.2. CASL

CASL, the *Common Algebraic Specification Language* [5], has been designed by CoFI, the *Common Framework Initiative* for algebraic specification and development. It has been designed by a large number of experts from different groups, and serves as a defacto standard. The design of CASL has been approved by the IFIP WG 1.3 "Foundations of System Specification". Originally CASL was designed for specifying software requirements and design, but this paper goes further to explore the use of CASL for the specification of ontologies [6,15].

CASL consists of two major *levels*, which are quite independent: *basic specifications* and *structured specifications*. This paper focuses on the basic specification level where theories are defined in terms of FOL axioms. These axioms are based on signatures introduced by the user. A signature is the declaration of symbols for sorts, predicates, total and partial functions. Sorts are to be interpreted as non-empty sets and are used for the types of predicates and functions. Subsort hierarchies are available and axioms can be formed by the usual first order logical connectives. A sort can be declared to consist only of values reachable by terms (generated types) or to be isomorphic to a term algebra (free types). CASL offers loose specification and design specification, where loose specification covers only requirements, but leaves representation issues unspecified. For design specifications it is possible to describe representations of datatypes in great detail.

The present approach uses CASL in a limited way; only predicates with one or two arguments, functions (operations) with one argument and constants are used. The specification of subsorting is not limited. CASL datatype definitions must not contain constructors with arguments; hence only datatypes consisting of subsort embeddings or constant constructors are allowed. We refer to this sublanguage of CASL as *2-FOL* in the rest of this paper.

2.3. CASL-DL

CASL-DL is a sublanguage of CASL, which is equivalent to the description logic $\mathcal{SHOIN}(\mathbf{D})$ [15,16,1]. Basically, this is a DL with concepts (unary predicates) and roles (binary predicates), which allows to specify a hierarchy of concepts and a hierarchy of roles. For roles, only specific axioms are allowed: they can be specified as functional, inverse functional, transitive or symmetric, and they can be related to other roles as inverse, equivalent or as subrole. Also, the range and domain of a role can be specified. Concepts can be fully defined by, or just imply, certain descriptions. Descriptions either allow us to describe the negation, union and intersection of descriptions, or they are just another concept. Further descriptions are role restrictions, which allow limited introduction of one new variable and either demand the existence of certain relations between members of two descriptions or restrict the second argument of a role to be in the specified description. The following paragraph gives a detailed list of CASL constructs allowed in CASL-DL which can be used for the specification of a $\mathcal{SHOIN}(\mathbf{D})$ theory.

Explicitly, the following CASL constructs are allowed in CASL-DL:

- sorts and subsorts, but limited to those having a common maximal supersort called *Thing*, used for classes / concepts;
- free types with only the subsort alternative used to define the disjoint union of concepts;
- subsort definitions;
- a free or generated type with constant constructors is used to define enumerated concepts where all members are known; a free type implies that all constants have distinct values;
- predefined datatypes are allowed which have *DATA* as maximal supersort and form a hierarchy separate from the concept hierarchy below *Thing*;
- predicates with an arity of one (possibly empty concepts / classes) or two (roles / properties) arguments;
- partial functions restricted to one argument (functional roles / properties) and total constants (individuals);
- types for predicates and functions are only *Thing* or subsorts of it as subject (first or only argument position); the object position (second argument or result) is either typed with *Thing* or *DATA* or a subsort of one of these; except for constants which are typed with *Thing* or a subsort of it;
- formulas defining predicates and functions with types *Thing* $\times$ *Thing* and *Thing* $\rightarrow$ *Thing*, which are restricted to implication, equivalence, symmetry, transitivity, functional and inverse functional axioms with the further restriction that functional predicates cannot be transitive; for predicates and functions which relate to *DATA* only equivalence and implication axioms are allowed; additionally, so-called argument restriction axioms are allowed which allow the implication of a conjunction of two descriptions (see below) each restricting the arguments of the role which forms the premise;

- description axioms characterize either named concepts (sorts or unary predicates) or relate two descriptions via implication (partial definition) or equivalence (complete definition);
- descriptions are the logical constants *true* and *false*, concept membership axioms, negation, union and intersection of descriptions ($\neg$, $\wedge$, $\vee$), existential quantifications stating the existence of a relation to some object (or data value) described by a description, all-value restrictions stating that all fillers (second argument of predicate or result of partial function) fall into the given description, and has-value restrictions stating the relation to a particular individual or data value and cardinality restrictions;
- facts are axioms involving only constants and stating the relation among them (the ABox).

Further details on the expressiveness of CASL-DL and its relation to OWL DL and $\mathcal{SHOIN}(\mathbf{D})$ can be obtained from [15].

3. Approximating FOL Theories

In [17] a method is described to derive tractable Horn clauses from arbitrary propositional theories. This method (called *knowledge compilation*) uses Horn approximation to obtain a tractable form of the theory for automated reasoning. The approximation produces a stronger and a weaker set of Horn clauses, which is used in the reasoning process. Furthermore, [17] shows a sketch for the approximation of arbitrary FOL theories by Horn clauses. Alvaro del Val studies in [18,11] a general algorithm for FOL Knowledge Compilation into various sublanguages of FOL, but he mainly focuses on variants of Horn-languages. Del Val shows that only a weaker theory (which is maximally strong, s. Sect. 3.2) is needed if only a sublanguage of the target language is allowed for query answering from the compiled theory [18].

3.1. Approximating 2-FOL by DL Formulas

Inspired by this Horn approximation we have developed a DL approximation of 2-FOL. Our method tries to find a weaker DL theory that is entailed by the 2-FOL theory. In terms of CASL structured specifications this is a *view*:

view ENTAILMENT_OF_DL : EXMPLTHY_DL **to** EXMPLTHY_FOL

that gives a proof obligation that all axioms in EXMPLTHY_DL are entailed by EXMPLTHY_FOL. The model theoretic semantics of this view is

$$\mathit{Mod}(\text{EXMPLTHY\_FOL}) \subseteq \mathit{Mod}(\text{EXMPLTHY\_DL}).$$

Hence we generate a DL theory that is logically weaker than the original theory and that has a larger set of models.

The signature of the 2-FOL theory is translated by newly introducing one maximal sort which subsumes all sorts which are concepts and not datatypes in the DL sense if such is a sort is not available before. This sort is then mapped to the sort *Thing*. All sorts

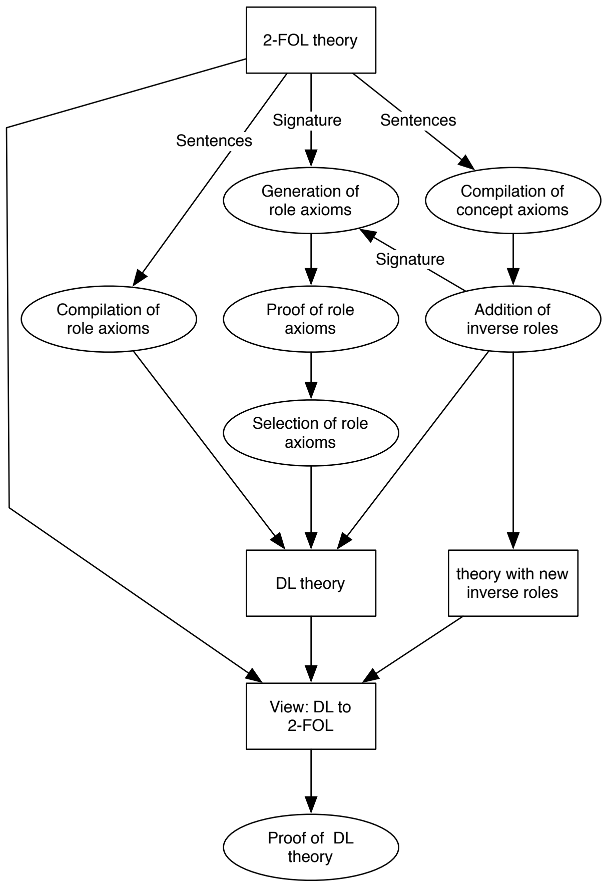

Figure 1. Flow of data during the generation of DL sentences

not subsumed by this topsort have to be mapped to one of the predefined datatypes in CASL-DL or must be hidden.

For the generation of the sentences of the approximated theory we distinguish two cases: (i) generation of axioms for each role (binary predicate) (ii) selection of formulas by patterns for argument restrictions, implications, equivalence and inverse axioms of roles and for descriptions of concepts (sorts and unary predicates). These patterns are derived from the allowed constructs in $\mathcal{SHIF}$ with reflexive relations, which is a sublogic of CASL-DL ($\mathcal{SHOIN}(\mathbf{D})$). $\mathcal{SHIF}$ has EXPTIME complexity but there exist DL reasoners that are highly optimized for this DL, e.g. Racer [12] and FaCT++ [13]. Although the DL variant we use is theoretically intractable, it is still tractable for practical purposes [19,20].

Figure 1 shows the schema of the following algorithm. Unlabeled edges transport a whole theory consisting of signature and sentences to the next node, labeled edges transport only signatures or sentences, respectively.

3.1.1. Generating role axioms

First we generate for each binary relation symbol in the theory transitive, symmetric, functional and inverse functional axioms and we try to prove all these axioms within the original theory. All proved axioms are included in the new theory except for those which are simultaneously prohibited by $\mathcal{SHIF}$. Here the user has to decide for each role which of the conflicting axioms should be kept.

3.1.2. Selection of role axioms by patterns

Here only those axioms are considered which involve binary predicates (roles) as premise or on one side of an equivalence axiom and are quantified over two variables. For the implications the consequent should either describe a concept for one or both arguments which yields an argument restriction or is a role application to the same argument variables (with the same order) as in the premise or is a conjunction of such constructs. Schematic template of these implications:

$\forall x,y\colon \mathrm{s} \bullet Rel_1(x,y) \Rightarrow \phi(x,y)$ and
$\phi(x,y) \equiv C_1(x) \mid C_2(y) \mid Rel_2(x,y) \mid \phi_1(x,y) \wedge \phi_2(x,y)$
where C_i are descriptions of concepts and ϕ_i are constructed by the same rules as ϕ

Equivalences where role applications occur on both sides either state that the two roles are equivalent or that one role is the inverse of the other one. These axioms are allowed in CASL-DL, hence they are just kept. Other equivalences are checked for conjunctions with binary predicates (having the same argument order) where each conjunct with a role yields a consequence of an implication.

Here are some examples of axioms concerning roles:

$\forall x,y\colon \mathrm{s} \bullet PP(x,y) \Leftrightarrow P(x,y) \wedge \neg P(y,x)$ %(Dd1_Proper_Part)%

where *s* is either the maximal topsort or a subsort of it. Here $PP(x,y)$ implies the conjunction and the conjunction has one conjunct that is allowed in DL. So the weak semantics that is compiled to DL is this

$\forall x,y\colon \mathrm{s} \bullet PP(x,y) \Rightarrow P(x,y)$ %(Dd1_Proper_Part_Impl)%

Clearly more than one implication could be derived from such a defining equivalence if more positive conjuncts are available. If *P* is transitive then *PP* is transitive, which can be proved. Further the symmetry of *O* can be proved directly from

$\forall x,y\colon \mathrm{s} \bullet O(x,y) \Leftrightarrow (\exists y\colon \mathrm{s} \bullet P(z,x) \wedge P(z,y))$ %(Dd2_Overlap)%

All the above proof obligations can be discharged automatically with SPASS [9].

3.1.3. Selection of concept axioms by patterns

The abstraction of formulas describing sorts or unary predicates is slightly more difficult than for binary predicates as there are recursive constructs of descriptions in CASL-DL. We focus on the cases where a formula is given as an implication or equivalence and respectively the premise or one of the equivalent formulas is a sort membership or unary predicate application. The formula must have only one outer free variable used in the left and right hand side of the logical connective. The consequent or other side of the equivalence must match a so-called role restriction: restrictions allow the introduction of a new variable only together with relation to the outer variable, where the outer variable is the first argument and the newly introduced variable is the second argument of the binary predicate. Here is the schema of such formulas where x is the outer variable and y the newly introduced one: $\phi(x) \equiv \forall y \bullet R(x, y) \Rightarrow \psi(y)$ In case that a predicate application fails to fall into this pattern, the inverse predicate is tried and introduced into the new theory if needed. Thus the axiom can be rewritten with the inverse predicate (formula %(Ad18_rw)% below in Section 3.3 is an example). Furthermore, a formula which is not universally quantified can be turned in such an implication as required above, with the sort membership as antecedent.

In summary, a view between the DL theory and the original theory plus inverse predicate definitions (introduced by the compilation) holds:

```
spec THY_FOL+ =
     THY_FOL
then %def
     <inverse-predicate-definitions>

view THEORY_IN_DL_TO_FOL : THY_DL to THY_FOL+
```

This method abstracts via approximation some 2-FOL theory into a semantically weaker but tractable DL theory such that for the model classes $Mod(\text{THY\_FOL}^+) \subseteq Mod(\text{THY\_DL})$ holds. Such a view and its symbol mapping are constructed as the last step of the approximation and its proof obligations must be discharged with SPASS (or Isabelle).

3.2. Is the resulting theory maximally strong?

One important question is not addressed in the above algorithm: Is the resulting DL-theory maximally strong with respect to the original theory? A DL-theory T_{DL} is maximally strong with respect to a FOL-theory T_{FOL} iff T_{DL} is weaker than T_{FOL} and equally strong as every DL-Theory between T_{DL} and T_{FOL}. A maximally strong theory is also called a minimal upper bound with respect to the original theory [17].

This paper will not give a complete answer to this question. But for roles (binary predicates) all possible formulas which can be generated with the given signature and are allowed (together) within a DL theory are kept if they can be proved. Thus, it is not possible to add further role axioms without leaving the restriction to DL, i.e. for formulas on roles this approach finds the maximal theory up to equivalence. For concept descriptions, only very few formulas are selected by patterns for inclusion into the DL theory. Here it is an open question, how close the approach in this paper comes to a maximal theory. Further ways of keeping more knowledge of the original theory in the DL theory and possibly finding a maximal DL theory with respect to the original theory are the subject of ongoing work.

3.3. Example: Mereology in FOL and CASL-DL

First a 2-FOL theory of mereology is presented which is the 2-FOL part of the FOL fragment of DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [3,4] as library MEREOFOL. A mereology provides the basic parthood relations for base concepts (categories) like *time interval* (*T*), *space region* (*S*) and *perdurant* (*PD*). We omit all ternary predicates (or two argument functions) like sum, product and difference which are present in the original FOL fragment of DOLCE. The specification PRIMITIVES gives a subset of the taxonomy with concepts named *particular* (*PT*), *physical endurant* (*PED*), *spatial location* (*SL*) and *temporal location* (*TL*). All subconcepts (subsorts) of *PT* are pairwise disjoint, as specified by the free type construct.

```
library MEREOFOL version 0.1

%{This library is based on “A fragment of DOLCE for CASL”
   by Stefano Borgo, Claudio Masolo
   LOA-CNR
   March 5, 2004}%

spec PRIMITIVES =
     sorts PD, PED, S, SL, T, TL
     free type PT = sorts PD, PED, S, SL, T, TL
end

spec GENPARTHOOD [sort s] =
     pred P : s × s
     ∀ x, y, z: s
     • P(x, x)                                              %(Ad11)%
     • P(x, y) ∧ P(y, x) ⇒ x = y                            %(Ad12)%
     • P(x, y) ∧ P(y, z) ⇒ P(x, z)                          %(Ad13)%
end

spec GENMEREOLOGY [sort s] =
     GENPARTHOOD [sort s]
then preds PP(x, y: s) ⇔ P(x, y) ∧ ¬ P(y, x);          %(Dd1_Proper_Part)%
           O(x, y: s) ⇔ ∃ z: s • P(z, x) ∧ P(z, y);          %(Dd2_Overlap)%
           At(x: s) ⇔ ¬ (∃ y: s • PP(y, x));                    %(Dd3_Atom)%
     ∀ x, y: s
     • ¬ P(x, y) ⇒ ∃ z: s • P(z, x) ∧ ¬ O(z, y)                  %(Ad14)%
     • ∃ z: s • At(z) ∧ P(z, x)                                  %(Ad18)%
then %implies
%% Probable Theorems (1)
     ∀ x, y, su, su′, p, p′, d, d′: s
     • (∀ z′: s • At(z′) ⇒ P(z′, x) ⇒ P(z′, y)) ⇒ P(x, y)         %(Td1)%
     • At(x) ⇔ (∀ y′: s • P(y′, x) ⇒ x = y′)                      %(Td2)%
     • (∀ z: s • O(z, x) ⇔ O(z, y)) ⇒ x = y                       %(Td3)%
end

spec MEREOLOGY =
     PRIMITIVES and GENMEREOLOGY [sort T] and
     GENMEREOLOGY [sort S] and GENMEREOLOGY [sort PD]
end
```

Library MEREODL shows the DL theory by applying the rules above to the library MEREOFOL. The inverse predicates of *PP* and *P* marked with *_i* are introduced to rewrite axioms %(Dd3_Atom)% and %(Ad18)%. Rewritten axioms are marked with _rw and axioms for inverse predicates are named either with the label of the original predicate definition or with the predicate name, and are marked with _inv. Derived implication, symmetry and transitivity axioms are named like the originating axiom and marked with respectively _impl, _sym and _trans. For the disambiguation of names they would be just numbered. Note that generic specifications are just kept in the resulting library of DL theories. So the structuring and grouping of the original theories is preserved and aids the readability of the result in this example. The real algorithm will just look at the theory of the specification to be compiled, but it can be extended to keep the structuring.

```
library MEREODL version 0.1

%{This library is based on “A fragment of DOLCE for CASL”
   by Stefano Borgo, Claudio Masolo
   LOA-CNR
   March 5, 2004
   further it is translated to CASL-DL}%

spec PRIMITIVES_DL =
     sorts PD, PED, S, SL, T, TL
     free type Thing = sorts PD, PED, S, SL, T, TL
end

spec GENPARTHOOD_DL [sort s] =
     pred P : s × s
     ∀ x, y, z: s
     • P(x, y) ∧ P(y, z) ⇒ P(x, z)                                   %(Ad13)%
end

spec GENMEREOLOGY_DL [sort s] =
     GENPARTHOOD_DL [sort s]
then preds PP : s × s;
          PP_i(x, y: s) ⇔ PP(y, x);                     %(Dd1_Proper_Part_inv)%
          O(x, y: s) ⇔ O(y, x);                            %(Dd2_Overlap_sym)%
          At(x: s) ⇔ ¬ (∃ y: s • PP_i(x, y));                %(Dd3_Atom_rw)%
          P_i(x, y: s) ⇔ P(y, x)                                     %(P_inv)%
     ∀ x, y, z: s
     • ∃ z′: s • P_i(x, z′) ∧ At(z′)                                %(Ad18_rw)%
     • PP(x, y) ⇒ P(x, y)                              %(Dd1_Proper_Part_impl)%
     • PP(x, y) ∧ PP(y, z) ⇒ PP(x, z)                 %(Dd1_Proper_Part_trans)%
end

spec MEREOLOGY_DL =
     PRIMITIVES_DL and GENMEREOLOGY_DL [sort T] and
     GENMEREOLOGY_DL [sort S] and GENMEREOLOGY_DL [sort PD]
end
```

The view from Mereology_DL to Mereology that we proved to show the correctness of the approximation is presented below. For the inverse predicate definitions again a generic specification InvPP_P is used as the original axioms were defined in a generic specification. The instantiations are also derived from the instantiations of GenMereology used in Mereology.

```
from MereoFOL get Mereology
from MereoDL get Mereology_DL

spec InvPP_P [sort s
              preds PP, P : s × s] =
     preds PP_i(x, y: s) ⇔ PP(y, x);
           P_i(x, y: s) ⇔ P(y, x)
end

spec Mereology_Inv =
     Mereology
and  InvPP_P [sort T
              preds PP, P : T × T]
and  InvPP_P [sort S
              preds PP, P : S × S]
and  InvPP_P [sort PD
              preds PP, P : PD × PD]
end

view Mereology_DL_to_FOL :
     Mereology_DL to Mereology_Inv =
     sorts Thing ↦ PT, T ↦ T, S ↦ S, PD ↦ PD, TL ↦ TL,
     SL ↦ SL, PED ↦ PED
end
```

4. Conclusion

We have presented a method for compiling the knowledge of a 2-FOL theory into a tractable DL theory. Further, we have given a method for deriving proof obligations to ensure that the approximated theory subsumes the FOL theory. This is done by constructing a view that shows the refinement from the approximated DL theory to the original theory. Also the derived proof obligations written as views can be structured along the original theory, so that the user easily finds the originating specifications. This method of approximation generates only a weaker approximation (rather than also a stronger one). This is sufficient if the query language is a sublanguage of the target language [18]. Thus, with this method a formal underpinning for applied ontologies can be generated that is based on a foundational ontology and approximates its semantics. Such a foundational ontology can be formalized in a very expressive language without losing the connection to tractable application ontologies.

4.1. Future Work

Further extensions to this approach should be investigated such as the encoding of n-ary predicates into binary predicates while not cluttering the whole theory with complicated formulas. It should be investigated how to preserve the structuring of the original theory such that the resulting DL theory will be more readable.

In addition we can examine the quality of the compiled theory, i.e. check whether it is maximally strong with respect to the original (or as close as possible, but stronger and in DL). The generation of the DL theory and the proof of the generated view will be implemented within the HETS framework, using SPASS (or if necessary Isabelle). We expect that the approximated theory can in any case be proved to be a logical consequence of the original automatically with SPASS.

Moreover it is worth studying an application to the entire theory of DOLCE and to develop rich domain ontologies based on this, e.g. for Spatial Cognition. Furthermore, one should have a look at keeping some concrete domains such as integers which could be expressed in CASL (with axiomatization) and CASL-DL (only as literals). This would compile into $\mathcal{SHIF}(\mathbf{D})$, which is a DL that is still practically tractable and a sublogic of CASL-DL.

Acknowledgments

This paper has been supported by Deutsche Forschungsgemeinschaft (DFG) in the SFB/TR8 “Spatial Cognition”. We thank Stefano Borgo, Nino Trainito, Till Mossakowski, Claudio Masolo, John Bateman, Stefan Wölfl, Scott Farrar, Lutz Schröder, Robert Ross, Wencke Lüttich and Nicola Guarino for discussions and ideas. Finally we thank two anonymous referees for their advice and comments.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[3] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening Ontologies with DOLCE. In A. Gómez-Pérez and V. R. Benjamins, editors, *EKAW*, volume 2473 of *LNCS*, pages 166–181. Springer Verlag; Berlin; `http://www.springer.de`, 2002.

[4] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, and L. Schneider. WonderWeb deliverable D17. The wonderWeb Library of Foundational Ontologies and the DOLCE ontology. Preliminary Report (ver. 2.0, 15-08-2002).

[5] CoFI (The Common Framework Initiative). CASL *Reference Manual*. LNCS 2960 (IFIP Series). Springer Verlag; Berlin; `http://www.springer.de`, 2004.

[6] K. Lüttich and T. Mossakowski. Specification of Ontologies in CASL. In A. C. Varci and L. Vieu, editors, *Formal Ontology in Information Systems – Proceedings of the Third International Conference (FOIS-2004)*, volume 114 of *Frontiers in Artificial Intelligence and Applications*, pages 140–150. IOS Press; Amsterdam; `http://www.iospress.nl`, 2004.

[7] T. Mossakowski. The Heterogeneous Tool Set. Available at `www.tzi.de/cofi/hets`, University of Bremen.

[8] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Springer Verlag; Berlin; `http://www.springer.de`, 2002.

[9] C. Weidenbach, U. Brahm, T. Hillenbrand, E. Keen, C. Theobalt, and D. Topic. SPASS version 2.0. In A. Voronkov, editor, *Automated Deduction – CADE-18*, volume 2392 of *Lecture Notes in Computer Science*, pages 275–279. Springer Verlag; Berlin; `http://www.springer.de`, July 27-30 2002.

[10] B. Selman and H. Kautz. Knowledge Compilation Using Horn Approximation. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 904–909, 1991.

[11] A. del Val. First order LUB approximations: characterization and algorithms. *Artif. Intell.*, 162(1-2):7–48, 2005.

[12] V. Haarslev and R. Möller. Racer System Description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *IJCAR*, volume 2083 of *Lecture Notes in Computer Science*, pages 701–706. Springer Verlag; Berlin; `http://www.springer.de`, 2001.

[13] I. Horrocks. FaCT++. Available at `http://owl.man.ac.uk/factplusplus/`.

[14] E. Sirin, M. Grove, B. Parsia, and R. Alford. Pellet OWL reasoner. `http://www.mindswap.org/2003/pellet/index.shtml`, May 2004.

[15] K. Lüttich, T. Mossakowski, and B. Krieg-Brückner. Ontologies for the Semantic Web in CASL. In J. Fiadeiro, editor, *Recent Trends in Algebraic Development Techniques, 17th International Workshop (WADT 2004)*, volume 3423 of *Lecture Notes in Computer Science*, pages 106–125. Springer Verlag; Berlin; `http://www.springer.de`, 2005.

[16] I. Horrocks and P. F. Patel-Schneider. Reducing OWL Entailment to Description Logic Satisfiability. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 17–29. Springer Verlag; Berlin; `http://www.springer.de`, 2003.

[17] B. Selman and H. A. Kautz. Knowledge Compilation and Theory Approximation. *J. ACM*, 43(2):193–224, 1996.

[18] A. del Val. An Analysis of Approximate Knowledge Compilation. In *IJCAI (1)*, pages 830–836, 1995.

[19] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. Log. Comput.*, 9(3):385–410, 1999.

[20] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001.