Approximation of Ontologies in CASL

Klaus Lüttich

FB3, Universität Bremen, P.O.B. 33 04 40, 28334 Bremen, Germany
luettich@tzi.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 rewriting 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 it is decidable, and that efficient reasoning procedures are also 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 and decidable reasoning plays a major role. These weakening strategies are described with CASL (Common Algebraic Specification Language), and one of its sublogics CASL-DL, and will be integrated into Hets (Heterogeneous Tool Set).

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) [4, 10]. Therefore CASL [3] (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 [8]. One advantage of CASL is its tool Hets [11] (Heterogeneous Tool Set). It allows syntax and type checking of CASL and it is connected to the semi-automatic theorem prover Isabelle [12] and to the automatic first-order reasoner SPASS [16].

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 [13].

This paper is structured as follows: First we introduce description logics, CASL and CASL-DL, the languages used for 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 now [1]. Another advantage is the great number of available automatic reasoners, such as Racer [5], FaCT++ [6] or Pellet [15].

2.2 CASL

CASL, the *Common Algebraic Specification Language* [3], 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 de-facto 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 [8, 9].

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 symbols introduced by the user in terms of signatures. A signature is the declaration of symbols for sorts (types), predicates, total and partial functions. Subsort hierarchies are available and axioms can be formed by the usual first order logical connectives. CASL offers loose specification and design specification, where loose specification only cover the important requirements, but leave representation issues underspecified. For design specifications it is possible to describe representations of datatypes in great detail.

This 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, except that there must be a maximal topsort available that applies to all other sorts in the theory. 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 $\mathcal{SHOIN}(\mathbf{D})$ [9, 7, 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 demand either the existence of a certain number of 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.

The following constructs of CASL 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;
- predicates (roles / properties) with an arity of one or two arguments;
- partial functions (functional roles / properties) restricted to one argument 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 $Thing \to Thing$, 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 quantification stating the existence of a relation to some object (or data value) described by a description, a value restriction that all fillers (second argument of predicate or result of partial function) fall into the given description, has value restriction 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 [9].

3 Approximating FOL Theories

In [14] 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. It is an approximation because a stronger and a weaker set of Horn clauses is used in the reasoning process. These Horn clauses are derived from the original set of propositional clauses. Furthermore, [14] shows a sketch for the approximation of arbitrary FOL theories into Horn clauses.

3.1 Approximating 2-FOL to 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_FOL **to** EXMPLTHY_DL that gives a proof obligation that all axioms in EXMPLTHY_DL are entailed by EXMPLTHY_FOL. The model theoretic semantics of this view is
Mod(EXMPLTHY_FOL) $\subseteq$ *Mod*(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 theory is translated by selecting one maximal topsort which should subsume all sorts which are concepts and not datatypes in the DL sense. This sort is then mapped to the sort *Thing*. All sorts 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 CASL-DL. Figure 1 shows the schema of the following algorithm. Unlabeled edges transport a whole theory consisting of signature and sentences to the next node and for the other cases the edges are labeled.

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 conflicting. Here the user has to decide for each role which of the conflicting axioms should be kept.

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 consequence 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. Equivalences where on both sides occur role applications 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:

$$\forall x,y\text{: s} \bullet PP(x,y) \Leftrightarrow P(x,y) \wedge \neg P(y,x) \qquad \%(\text{Dd1\_Proper\_Part})\%$$

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

$$\forall x,y\text{: s} \bullet PP(x,y) \Rightarrow P(x,y) \qquad \%(\text{Dd1\_Proper\_Part\_Impl})\%$$

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

$$\forall x,y\text{: s} \bullet O(x,y) \Leftrightarrow (\exists y\text{: s} \bullet P(z,x) \wedge P(z,y)) \qquad \%(\text{Dd2\_Overlap})\%$$

And we proved all the above proof obligations automatically with SPASS.

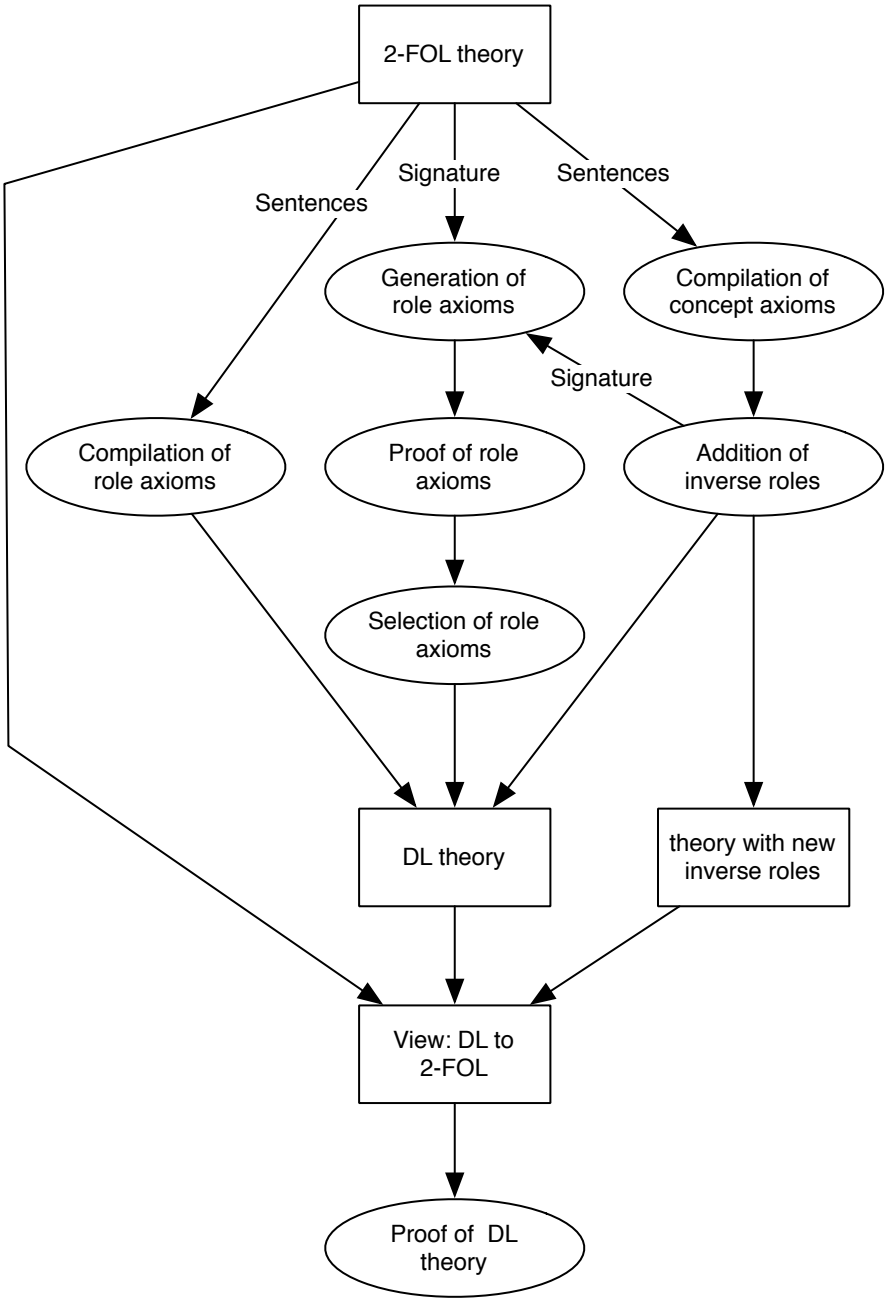

Fig. 1. Flow of data during the generation of DL sentences

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 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 the case that a predicate application fails to fall into this pattern, the inverse predicate is tried and introduced into the new theory. Thus the axiom can be rewritten with the inverse predicate (formula **%(**Ad18_rw**)%** below in Section 3.3 is an example).

So, as our DL theory is weaker than the original theory, a view between the DL theory and the original theory plus inverse predicate definitions (introduced by the compilation) holds:

spec EXMPLTHY_FOL$^+$ =
EXMPLTHY_FOL
then %def
<inverse-predicate-definitions>

view THEORY_IN_DL_TO_FOL :
EXMPLTHY_DL **to** EXMPLTHY_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*(FOL_THEORY$^+$) $\subseteq$ *Mod*(DL_THEORY) holds. Such a view and its symbol mapping is constructed as the last step of the approximation and must be proved either 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 weaker theory is maximally strong with respect to the original theory iff it is not possible to find a theory that is stronger than the already calculated theory and at the same time weaker as the FOL theory, or if the original FOL-theory is already a valid DL theory, than this is the maximally strong theory itself. A maximally strong theory is also called a minimal upper bound with respect to the original theory [14].

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 proven. So, here it is not possible to add further axioms without leaving the restriction to DL. So, for formulas on roles this approach finds the maximal theory up to equivalence. Certainly this has to be proved, which will be addressed in the future. 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 leads to a maximal theory. But, it is ongoing work to find further ways to keep more knowledge of the original theory within the DL theory. While this is investigated, also the question of finding a maximal DL theory with respect to the original theory will be answered.

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) [4, 10] 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) and all subconcepts (subsorts) of PT are pairwise disjoint which is specified by the free type. T Nowadays, the term RCC5 (Region Connection Calculus with 5 basic relations) is applied to such a theory of mereology, but the term mereology is much older and more appropriate.

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 $P : s \times s$
$\forall\ x, y, z : s$
• $P(x, x)$ %(Ad11)%
• $P(x, y) \wedge P(y, x) \Rightarrow x = y$ %(Ad12)%
• $P(x, y) \wedge P(y, z) \Rightarrow P(x, z)$ %(Ad13)%
end

spec GenMereology [**sort** s] =
GenParthood [**sort** s]
then preds $PP(x, y: s) \Leftrightarrow P(x, y) \wedge \neg P(y, x)$; %(Dd1_Proper_Part)%
$O(x, y: s) \Leftrightarrow \exists z: s \bullet P(z, x) \wedge P(z, y)$; %(Dd2_Overlap)%
$At(x: s) \Leftrightarrow \neg (\exists y: s \bullet PP(y, x))$; %(Dd3_Atom)%
$\forall x, y: s$
$\bullet \neg P(x, y) \Rightarrow \exists z: s \bullet P(z, x) \wedge \neg O(z, y)$ %(Ad14)%
$\bullet \exists z: s \bullet At(z) \wedge P(z, x)$ %(Ad18)%
then %implies
%% Probable Theorems (1)
$\forall x, y, su, su', p, p', d, d': s$
$\bullet (\forall z': s \bullet At(z') \Rightarrow P(z', x) \Rightarrow P(z', y)) \Rightarrow P(x, y)$ %(Td1)%
$\bullet At(x) \Leftrightarrow (\forall y': s \bullet P(y', x) \Rightarrow x = y')$ %(Td2)%
$\bullet (\forall z: s \bullet O(z, x) \Leftrightarrow O(z, y)) \Rightarrow x = y$ %(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 maybe 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 $P : s \times s$
  $\forall\ x, y, z: s$
  • $P(x, y) \wedge P(y, z) \Rightarrow P(x, z)$ %(Ad13)%
end

spec GENMEREOLOGY_DL [**sort** *s*] =
  GENPARTHOOD_DL [**sort** *s*]
then preds $PP : s \times s$;
    $PP\_i(x, y: s) \Leftrightarrow PP(y, x)$; %(Dd1_Proper_Part_inv)%
    $O(x, y: s) \Leftrightarrow O(y, x)$; %(Dd2_Overlap_sym)%
    $At(x: s) \Leftrightarrow \neg\ (\exists\ y: s \bullet PP\_i(x, y))$; %(Dd3_Atom_rw)%
    $P\_i(x, y: s) \Leftrightarrow P(y, x)$ %(P_inv)%
  $\forall\ x, y, z: s$
  • $\exists\ z': s \bullet P\_i(x, z') \wedge At(z')$ %(Ad18_rw)%
  • $PP(x, y) \Rightarrow P(x, y)$ %(Dd1_Proper_Part_impl)%
  • $PP(x, y) \wedge PP(y, z) \Rightarrow PP(x, z)$ %(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*, $P : s \times s$] =
  preds $PP\_i(x, y: s) \Leftrightarrow PP(y, x)$;
    $P\_i(x, y: s) \Leftrightarrow P(y, x)$
end

view MEREOLOGY_DL_TO_FOL :
MEREOLOGY_DL **to**
{MEREOLOGY
and INVPP_P [**sort** T
preds PP, $P : T \times T$]
and INVPP_P [**sort** S
preds PP, $P : S \times S$]
and INVPP_P [**sort** PD
preds PP, $P : PD \times PD$] } =
sorts $Thing \mapsto PT$, $T \mapsto T$, $S \mapsto S$, $PD \mapsto PD$, $TL \mapsto TL$,
$SL \mapsto SL$, $PED \mapsto PED$
end

4 Conclusion

In this paper, we have presented a method to compile the knowledge of a 2-FOL theory into a tractable DL theory. Further we gave a method to derive 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 down as views could be structured along the original theory such that the user easily finds the originating specifications. This method of approximation only generates a weaker theory; so, 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 connection to tractable application ontologies.

Future Work. Further extensions to this approach should be investigated such as the encoding of n-ary predicates into binary predicates while not scattering 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. One could think of developing also a stronger approximation, such that the two compiled theories can be used for efficient reasoning analogously to [14]. In addition we can look at the quality of the compiled theories, i.e. are they maximally weak or minimally strong with respect to the original. The generation of the DL theory and the proof of the generated view should be implemented within Hets and SPASS (or Isabelle). Best would be a proof 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).

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, Robert Ross and Nicola Guarino for discussions and ideas. Finally we thank three anonymous referees for their advices 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. CoFI (The Common Framework Initiative). CASL *Reference Manual*. LNCS 2960 (IFIP Series). Springer Verlag; Berlin; `http://www.springer.de`, 2004.
4. 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.
5. Volker Haarslev and Ralf Möller. Racer System Description. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *IJCAR*, volume 2083 of *Lecture Notes in Computer Science*, pages 701–706. Springer Verlag; Berlin; `http://www.springer.de`, 2001.
6. I. Horrocks. FaCT++. Available at `http://owl.man.ac.uk/factplusplus/`.
7. 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.
8. K. Lüttich and T. Mossakowski. Specification of Ontologies in CASL. In Achille C. Varci and Laure Vieu, editors, *Formal Ontology in Information Systems – Proceedings of the Third International Conference (FOIS-2004)*, volume 114 of *Monographs of the Bremen Institute of Safe Systems (BISS)*, pages 140–150. IOS Press; Amsterdam; `http://www.iospress.nl`, 2004.
9. K. Lüttich, T. Mossakowski, and B. Krieg-Brückner. Ontologies for the Semantic Web in CASL. In José 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.
10. 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).
11. T. Mossakowski. The Heterogeneous Tool Set. Available at `www.tzi.de/cofi/hets`, University of Bremen.
12. 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.
13. 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.

14. Bart Selman and Henry A. Kautz. Knowledge Compilation and Theory Approximation. *J. ACM*, 43(2):193–224, 1996.
15. E. Sirin, M. Grove, B. Parsia, and R. Alford. Pellet OWL reasoner. `http://www.mindswap.org/2003/pellet/index.shtml`, May 2004.
16. C. Weidenbach, U. Brahm, T. Hillenbrand, E. Keen, C. Theobalt, and D. Topic. SPASS version 2.0. In Andrei 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.