Towards Novel Techniques for Reasoning in Expressive Description Logics based on Binary Decision Diagrams *

Uwe Keller

Digital Enterprise Research Institute (DERI), University of Innsbruck, Austria
uwe.keller@deri.org

Abstract. We propose to design and study new techniques for description logic (DL) reasoning based on a prominent data structure that has been applied very successfully in various domains in computer science where one has to face the efficient representation and processing of large scale problems: *Binary Decision Diagrams* (BDDs). BDDs have been used very successfully for reasoning in propositional logics, and have been lifted to the level of first-order logics, too. In both cases, they provide a rich semantic structure to guide proof search. Therefore, we believe that (i) BDDs are interesting to study in the context of reasoning for a logic of intermediate expressivity (such as DLs) and (ii) that they provide a fertile ground for the design of novel efficient methods for reasoning in particular expressive DLs. The project will help to enrich the available machinery of DL reasoning techniques.

1 Introduction

Description Logics (DLs) [1] are a family of class-based knowledge representation formalisms characterised by the use of various constructors to build complex classes from simpler ones, and by an emphasis on the provision of sound, complete and (empirically) tractable reasoning services. They have a wide range of applications, but are most widely known as the basis for ontology languages such as OWL. Recently, [14] pointed out that the increasing use of DL-based ontologies in areas such as e-Science and the Semantic Web however is already stretching the capabilities of existing DL systems, and brings with it a range of challenges for future research on reasoning methods for DL. Key issues here are the provision of efficient algorithms that allow (advanced) applications (i) to scale up to knowledge bases of practical relevance and (ii) to leverage expressive languages for capturing domain knowledge. However, expressiveness of DLs comes at a price: the theoretically high (worst-case) complexity of relevant reasoning tasks. Hence, it is unlikely, that there is a *single* method, that performs well in all possible cases. Rather, one can expect that specific techniques perform well one particular classes of problems.
So far, research in practical DL reasoning methods has centered around structural subsumption algorithms [2] and tableau methods [13], and have recently been extended by the application of the resolution principle [16,18] (and optimized evaluation techniques from the area of deductive databases) to expressive DLs. Automata-based approaches (e.g.(e.g. [24]) (although possible in theory) have had nearly no impact on the development of practical reasoning algorithms for DLs.
Based on the observation of recent trends in the area of DL reasoning and the research challenge identified in [14] for this field, we propose to design and research novel techniques for Description Logic reasoning that are based on a well-known principles of reasoning that (a) has been studied for other (especially more expressive) logics, (b) proved itself to be a successful method of reasoning for these logics and (c) work significantly different from current state-of-the-art techniques in DL reasoning. More specifically, we propose to investigate the use of *Binary Decision Diagrams* (BDDs) [3] and their manifold variants and extensions as a fundamental framework for realizing well-known reasoning tasks, in particular for expressive DLs.

2 Binary Decision Diagrams and their Variants

A Binary Decision Diagram (BDD) [3] is a simple data structure for representing an (n-ary) boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$. A boolean function $f(x_1, \ldots, x_n)$ can be represented as a rooted, directed, acyclic graph, which consists of decision nodes and two terminal nodes called 0-terminal and 1-terminal. Each decision node is labeled by a Boolean variable x_i and has two child nodes called low child and high child. The edge from a node to a low (high) child represents an assignment of the variable to 0 (1). Such a BDD is called *ordered* if different variables appear in the same order on all paths from the root. It is called *reduced* if the graph is reduced according to two rules: (i) merge any isomorphic subgraphs, and (ii) eliminate any node whose two children are isomorphic. Consequently, reduced BDDs reuse structures in the BDD representation to a maximum extent and therefore shrink the size of the representation. Most

* This work has been funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the project *Semantic Engineering Support Environment* (SEnSE, FIT-IT contract FFG 810807).

often, the term BDD refers actually to Reduced Ordered Binary Decision Diagram (ROBDD), i.e. BDDs that are reduced in regard of a specific (given) order. The advantage of an ROBDD is that it is canonical (unique) for a particular boolean function: Although the boolean function might have various (equivalent) descriptions, the respective ROBDD is unique. A path from the root node to the 1-terminal represents a (possibly partial) variable assignment for which the represented Boolean function has the value true. As the path descends to a low child (high child) from a node, then that node's variable is assigned to 0 (1).

The fundamental and most important characteristic of ROBDDs hereby is (i) an extreme compression in many practical cases (after the application of reduction rules to remove eliminate redundancies) and (ii) very fast implementations of standard operations on boolean functions. Although the (naive) representation of a boolean function in a BDD might be very large (and require exponential space wrt. the number of boolean input variables), given some fixed ordering on input variables, BDDs can most often be reduced to an OBDDs (representing the same function) such that the resulting representation actually is comparably small (e.g. polynomial wrt. the BDD representation). In particular, for a given ordering the reduced form is unique, and the OBDD for the $n-ary$ boolean function which returns always 0 consists only of the 0-terminal node. The achievable compression crucially depends on the chosen variable ordering, i.e for many boolean functions there exists an ordering such that the corresponding reduced OBDD has a minimal size. On the other hand, there are functions (e.g. the multiplication function) that are inherently difficult, i.e. no variable ordering exists such that the reduced OBDD has small size. Practical experience shows, that such functions are rare in many industrial applications. Finding an optimal variable ordering is known to be intractable [26], however heuristics often work well in practice [23].

BDDs have been applied in various domains, most prominently hardware verification [17], in Computer Aided Design (CAD), and in software to synthesize circuits (logic synthesis). Very often, they superseded previously known methods. Various variations and generalizations of BDDs have been developed over time to overcome limitations for particular domains, e.g. Zero Suppressed Decision Diagrams (ZDDs), Binary Moment Diagrams (BMDs), Free Binary Decision Diagrams (FBDDs), (reduced ordered) Multi-valued Decision Diagrams ((RO)MDDs).

3 How to Reason with BDDs

BDDs can be used for reasoning in propositional logics straightforwardly: A propositional formula $\phi(x_1,\ldots,x_n)$ containing n propositional variables x_i can be seen as an n-ary boolean function. To construct a BDD for $\phi(x_1,\ldots,x_n)$ one can apply *Shannon's decomposition principle*: for all boolean variable assignments $x_1,\ldots,x_n \in \{0,1\}$ it holds that $\phi \Leftrightarrow (x_i \Rightarrow \phi\{x_i/1\}) \wedge (\neg x_i \Rightarrow \phi\{x_i/0\})$, where $\phi\{x/\phi'\}$ denotes the formulae which is constructed from ϕ by replacing all occurrences of x by ϕ'. The principle can be recursively applied, potentially in regard of a given ordering $\prec$ on the propositional variables x_i in ϕ. Reduction and simplification operations can be applied after each step to construct a ROBDD. Since the ROBDD for a boolean function is unique, one can read off immediately from the constructed BDD, if the respective input formula is unsatisfiable (or valid): ϕ is unsatisfiable (valid) if the respective ROBDD contains only the node 0 (1). This shows immediately that the construction of an ROBDD in the worst-case is expensive (unless $P = NP$). However, in practice (especially when applying a suitable variable ordering $\prec$) the construction of ROBDDs can be done efficiently. Alternatively, BDDs can be constructed bottom up, too, starting from atomic subformulae stepwise to increasingly complex sub-formulae of ϕ, since the application of logical operators (e.g. $\wedge$,$\vee$,$\neg$,$\Rightarrow$, $\Leftrightarrow$) to combine formulae to more complex ones can be implemented very efficiently (i.e. in linear or quadratic time in the size of the BDDs to be combined) as standard BDD graph operations on the the corresponding BDDs.

Interestingly, BDDs are very rich structures for storing semantic information about the input formulae ϕ: in the propositional case, paths from the root to the 1 leaf node compactly represent all *models* of ϕ (wrt. to the given propositional signature). Analogously, all pathes from the root to the 0 leaf node capture compactly all *counter models* for ϕ, i.e. interpretation for which ϕ is not satisfied. Further, if one considers a 1-path as a conjunction of literals and the set of 1-pathes disjunctively combined, then the BDD contains a *disjunctive normal form* of ϕ. At the same time, one can directly interpret the set of 0-pathes in the BDD as a *conjunctive normal form* for ϕ. This is promising since proof search strategies can be implemented on top of BDDs that use either normal form representation. Since tableau methods can be seen as processes that derive a disjunctive normal form for an input formula ϕ and resolution methods as processes that iteratively extend conjunctive normal forms of ϕ, we expect that techniques from both fields can be considered for the design of efficient proof search strategies. Further, we believe that BDDs are able to provide a uniform structure that can be used to realize a variety of reasoning tasks for logics (beyond satisfiability), because they are inherently encode compactly a lot of semantic (i.e. syntax-independent) information about ϕ: in the propositional case this is the whole truth table of ϕ.

The link to First-order Logics (FOLs) is as well rather straightforward: Let Σ be a first-order signature (including two 0-ary predicates 1 and 0 denoting the respective truth values and a set $\mathcal{V}$ of variable names). The set of terms $Term(\Sigma)$ is defined as usual as the smallest set containing all variables $x \in \mathcal{V}$ and is closed under the application of n-ary function symbols $f \in \Sigma$ to any combination of n terms. The set of atomic formulae $\mathcal{L}_0(\Sigma)$ is defined as the set of expressions that can be generated from terms by applying any n-ary predicate symbol $p \in \Sigma$ to any combination of n terms. The First-order Logic $\mathcal{L}(\Sigma)$ over signature Σ is then defined as the smallest set of expression that contains all atomic formulae

$\phi \in \mathcal{L}_0(\Sigma)$ and is closed under the application of the usual logical junctors $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ as well as the application of any quantor $Q \in \{\exists, \forall\}$ to any pair of variable $x \in \mathcal{V}$ and formula $\phi \in \mathcal{L}(\Sigma)$.
For the sake of simplicity (and without loss of generality[1]), we consider here only formulae $\phi \in \mathcal{L}(\Sigma)$ in universal prenex form, i.e. have the form $\phi = \forall x_1, \ldots, x_n.M_\phi$ with M_ϕ a quantifier free formulae. The described techniques can be extended to the full language $\mathcal{L}(\Sigma)$ as shown e.g. in [22,9]. For such a ϕ we construct the Binary Decision Diagram $BDD_\phi = bdd(M_\phi)$, i.e. a graph (V, E) with vertices $v \in V$ and l-labeled edges $e = (v, l, v') \in E$ recursively as follows:

$$bdd(\psi) = \begin{cases} (\{l\}, \emptyset) & \text{if } \psi = l \in \{0,1\} \\ (\{a, 0, 1\}, \{(a, +, 1), (a, -, 0)\}) & \text{if } \psi = a \in \mathcal{L}_0(\Sigma) \setminus \{0,1\} \\ negbdd(bdd(\psi')) & \text{if } \psi = \neg\psi' \\ conjunctbdd(bdd(\psi'), bdd(\psi'')) & \text{if } \psi = \psi' \wedge \psi'' \\ disjunctbdd(bdd(\psi'), bdd(\psi'')) & \text{if } \psi = \psi' \vee \psi'' \\ implbdd(bdd(\psi'), bdd(\psi'')) & \text{if } \psi = \psi' \Rightarrow \psi'' \\ biimplbdd(bdd(\psi'), bdd(\psi'')) & \text{if } \psi = \psi' \Leftrightarrow \psi'' \end{cases}$$

whereby $negbdd, conjunctbdd, disjunctbdd, implbdd, biimplbdd$ represent standard operations to negate BDDs and to combine BDDs conjunctively, disjunctively, by implication, and biimplication. The resulting BDD_ϕ therefore contains only nodes that represent atomic subformulae occurring in ϕ, 1, or 0; atomic subformulae are considered as (unstructured) propositional letters. For any given ϕ, BDD_ϕ can be constructed in finite time (and usually very fast). Reduction (wrt. a fixed order $\prec$ on atomic formulae in $\mathcal{L}_0(\Sigma)$ can be applied as in the propositional case. If BDD_ϕ is considered as a formula (in the so-called *if-then-else* or Shannon normal form), then BDD_ϕ is logically equivalent to M_ϕ.
Clearly, since BDD_ϕ in general contains atomic formulaes with variables, we can not determine the unsatisfiability of ϕ directly from the graph structure. However, it is a compact, logically equivalent representation of M_ϕ that allows to check (in many practical cases) efficiently for unsatisfiability if no variable were present in ϕ, or if M_ϕ contains variables but is already propositionally unsatisfiable. Hence, the question is how to deal with the variables (and therefore the quantifiers) in $\phi = \forall x_1, \ldots, x_n.M_\phi$ which is equivalent to $\forall x_1, \ldots, x_n.BDD_\phi$. Here, Herbrand's theorem [4] provides the theoretical means to identify the missing piece to devise a proof procedure for FOL, since it allows to reduce FOL unsatisfiability to unsatisfiability on propositional logic: A formulae of the form $\phi = \forall x_1, \ldots, x_n.M_\phi$ is unsatisfiable iff. there exists a $k \in \mathbb{N}$ and a substitution σ such that $(M^1_\phi \wedge M^2_\phi \wedge \ldots \wedge M^k_\phi)\sigma$ is a propositionally unsatisfiable formulae, whereby M^i_ϕ denotes a „new" copy of M_ϕ where the variables $x_1, \ldots, x_n$ in M_ϕ have been renamed uniquely to $x^i_1, \ldots, x^i_n$ such that they do not occur by any other copy M^j_ϕ and $x^i_k \neq x^i_l$ if $x_k \neq x_l$.
Therefore, we can devise a FOL proof procedure by enriching the data structure BDD_ϕ with a search procedure that attempts to find suitable number of extension step k and ground substitution σ, such that $\Pi_{k,\sigma}(\phi) = (M^1_\phi \wedge M^2_\phi \wedge \ldots \wedge M^k_\phi)\sigma$ can be demonstrated as being unsatisfiable. Since $\Pi_{k,\sigma}(\phi)$ is propositional and can be efficiently constructed (for any k) from BDD_ϕ (essentially by application of the standard $conjunctbdd$ operation), the compact representation of M_ϕ and the „built-in" unsatisfiability check are promising features of BDDs as the basis of a FOL proof procedure. For a given k (starting with $k = 1$), the search procedure can try to find a suitable substitution σ that „falsifies" (or refutes) the BDD and iteratively increase the number of required copies k if all possibilities have been explored but turned out to be unsuccessful. Clearly, blind guessing of candidates σ is absolutely undesirable. As in Semantic Tableau and Resolution, the proof procedure should take the formulae (and its structure) itself into account and use well-known tools such as unification and the computation of most general unifiers. Here, BDDs provide again a rich structure and various options for this specific purpose (even if orderings are not used), such as analysis and elimination of 1-paths or strategies that work with 0-paths instead. Clearly, the proof procedure is only guaranteed to terminate in the case of an unsatisfiable formula. For FOL, this can not be changed since the set of satisfiable formulas in FOL is not recursively enumerable.
A straightforward way to apply BDDs to DL reasoning could then be as follows: many DLs can be considered as very restricted subsets of FOLs, where the syntactic restrictions lead to decidability of fundamental reasoning tasks such as unsatisfiability of a knowledge base. This even works for very expressive DLs as long as they can be „embedded" to FOL. The main question here is how to achieve the termination of the FOL proof procedure in these cases. Clearly, there are two parameters to play with: (a) the translation function that embeds a given DL knowledge base into a set of first-order formulae, and (b) suitable refinements (or restrictions) of the proof search process based on the specific characteristics of the underlying DL (such as the finite tree model property) or the syntactic structure of the generated set of FOL formulae. In regard of (b), we are very optimistic, since BDDs can be used to generate some tableau-like as well as some resolution-like (micro) inference steps (when simplifying the BDD that represent the current state of the proof search), we expect that certain well-studied techniques can be rebuilt in the BDD framework. At the same time we can exploit in the BDD framework that it possible to do both at the same time: checking unsatisfiability of a formula (non-existence of consistent and deductively complete 1-paths) as well as its satisfiability (the presence of a consistent deductively complete 1-path). Therefore, novel techniques for reasoning (even for very expressive DLs), potentially interweaving both processes can be designed and investigated thoroughly.

[1] Every formulae $\phi \in \mathcal{L}(\Sigma)$ can be transformed into an equi-satisfiable formula in universal prenex form in polynomial time [19]

In the past, a few approaches [21,8,22,12] on how to generalize the principles underlying BDDs and OBDDs from the propositional level to the first-order level have been studied. Each of them could serve as a distinct starting point for our purposes and will be investigated closely. A brief overview of essential underlying principles is given in [10]. Interestingly, [11] shows theoretically that BDDs and Resolution are fundamentally different techniques for Propositional Logic, whereby the argument carries over to FOL. Further, [22] discusses the relation of their specific approach to First-order Semantic Tableaux, and points out the specifically important advantage of the BDD-based method over Semantic Tableaux which is the property of very compact representations during proof search.

All these approaches target at First-order Logics and therefore at *checking for unsatisfiability* of an input formula. For reasoning in DLs, a possible and slightly different approach would be the following: one exploits the rich structure of BDDs to search for models of an input formula, i.e. to build a *model generation procedure* based on BDDs. This is essentially the basic idea underlying the DL tableau procedures (that work on a different representation than BDDs) and can be expected to simplify termination proofs for all input formulas (e.g. for DLs with the finite model property).

The resulting model generation algorithm will be different from the proposed unsatisfiability checking algorithms for FOL, since it uses the information that is represented in the BDD in a different way and applies different modifications. Still, both algorithms can be represented and performed on top of the BDD representation of the input formulae (or knowledge base). This suggests to study the possibility and efficiency of deductive process that interweave both activities (i.e. theorem proving and disproving). In consequence, a resulting model generation procedure would be applicable (when dropping certain DL-specific assumptions) to First-order Logics, too, and potentially result in novel techniques for model generation for FOLs.

4 Related Work

In the following we briefly discuss approaches that are relevant for DL reasoning and make major use of Binary Decision Diagrams during the proof search.

The Knowledge Cartographer Approach. The work reported in [5,6,7] takes a purely set-theoretic perspective on DL reasoning, especially on TBoxes. The underlying idea is simple, yet elegant: Given a DL signature Σ for any interpretation over Σ the universe under consideration is partitioned into a number of non-overlapping sets (so-called *atomic regions*). Given any interpretation, the extension (or interpretation) of any concept expression over Σ can be composed by atomic regions (via set-theoretic union) only. If we consider a given TBox $\mathcal{T}$ (as it is common in practical applications), then the possible partitions of the universe of models of $\mathcal{T}$ are often restricted severely (in comparison to the partitions for arbitrary interpretations) and the number of atomic regions decreases drastically. Hence, if n is the number of atomic regions, then any concept expression can be identified with an n-dimensional bit vector in $\mathbb{B}^n$ (the so-called *signature*). The base vectors of the canonical basis of $\mathbb{B}^n$ represent the atomic regions themselves. Since concept are constructed essentially by means of set-theoretic operations, the most important (yet simple) concept constructors (such as $\sqcap, \sqcup, \neg$) can be very efficiently mapped (and implemented) by means of bit-operations on signature. Important semantic tests between concept expressions can be check by simple comparison of the bit vectors (that are linear in the size n of the bit vectors), e.g. concept C_1 is subsumed by concept C_2 if for the corresponding signatures (or bit vectors) it holds that $sig(C_1) \leq sig(C_n)$. However, in the worst-case the required length n of the bit vectors is exponential in the size of the signature. The key problem in this approach is to determine needed atomic regions for a given signature Σ and TBox $\mathcal{T}$. Ordered BDDs are taken as an efficient means for computing the signatures that need to be considered for a given TBox $\mathcal{T}$. In this sense, $\mathcal{T}$ is compiled in a preprocessing step into a semantic data structure that later on simplifies particular semantic checks, such as concept subsumption. The approach is defined for a restricted subset of the DL $\mathcal{ALC}$ and can not deal with arbitrary concept descriptions for ABox queries. The reported evaluation results seem to indicate superior performance over state-of-the-art systems, especially in the presence of ABoxes of significant size. One has to keep in mind here, that the a system for a rather limited DL is compared against more general DL system. Further, the performed result are not well documented and do not give a clear indication of scientific significance of measured experiment.

A BDD-based calculus for Reasoning in the Modal Logic K. Very recently [20] proposed a novel satisfiability checking procedure for formulae in the basic modal logic $\mathbf{K}$. It has been reported that a corresponding implementation is competitive or even superior to existing highly-optimized modal reasoning systems for certain knowledge bases (KB). In particular, for formulae that require extensive modal reasoning, the method seems to perform very well. The authors note, that the method can be extended to the multi-modal logic $\mathbf{K}_{(\mathbf{m})}$. Since $\mathbf{K}_{(\mathbf{m})}$ can be considered as a syntactic variant of the description logic $\mathcal{ALC}$ [25] (where m corresponds to the number of role names in the underlying DL signature), the proof procedure is suitable for reasoning with the non-trivial DL $\mathcal{ALC}$, too. It is not clear to what extent it is possible to transfer the approach to other more expressive DLs than $\mathcal{ALC}$. Further, testing satisfiability of a formula wrt. to background knowledge is not covered in [20].

In contrast, to these specific related BDD-based techniques the approach that we propose addresses DL reasoning by refinement and tailoring of BDD-based calculi for a logic that is more expressive than many expressive DLs, namely First-order Logic. It has therefore inherently the advantage to be applicable to a wide range of expressive DLs, that go

beyond $\mathcal{ALC}$. Further, it has the potential to support expressive extensions of DLs that result in undecidable yet empirically still tractable knowledge representation frameworks. We expect that major elements of the Knowledge Cartographer approach naturally arise in our framework and allow to related both approaches on a more detailed technical level.

5 Conclusions & Future Work

We have proposed to investigate the use of BDDs and their manifold variants and extensions as a fundamental framework for realizing well-known reasoning tasks, in particular for expressive DLs. BDDs are very rich structures capturing a variety of useful information that can be exploited by inference calculi during proof search. We are especially interested in investigating and refining BDD-based calculi that have been proposed for First-order Logics a couple of years ago. Since there are various different ways of using BDDs to design new inference calculi and numerous variants of the BDD data structure, we expect that BDDs give us sufficiently many options for our investigation and gives enough room to come up with useful novel inference techniques. Using the approach of refining FOL techniques naturally enables us to cope with various expressive DLs (such as the ones underlying the Web Ontology Language (OWL) in version 1.0 ($\mathcal{SHOIN}(\mathbf{D})$) and version 1.1. ($\mathcal{SROIQ}$ [15], with extensions for metamodeling and n-ary datatypes). Further, the investigation of the behavior of the developed methods for certain tractable subsets of such DLs (e.g. the ones discussed in OWL 1.1 language proposal[2]) is certainly desirable.
We expect that the proposed research agenda helps to evolve the state-the-art in the field of Description Logic reasoning by (i) extending the available machinery of tools for reasoning in expressive DLs by distinctively novel methods and (ii) by provision of a deep understanding of the strengths and potential weaknesses of the developed novel methods. Our ultimate aim is to design techniques to help to increase the possible range of applications of DLs for knowledge-based and intelligent systems. As far as possible, we aim to identify potential extensions to existing DL-based knowledge representation frameworks to make them more expressive for applications while still staying in an empirically tractable framework. This is well in line with the needs of advanced applications of DLs as is has been discussed in [14]. Technical details of a BDD-based inference calculus for $\mathcal{ALC}$ are subject to an upcoming paper. In all approaches to BDD-based FOL proof procedures that we are aware of, specific means of equality reasoning or reasoning with of concrete data-types have not been studied yet. However, this is needed to deal with popular features in expressive DLs such as cardinality restrictions or concrete domains and it therefore subject of investigation after we are able to deal with $\mathcal{ALC}$. Eventually, investigations on how to deal with large ABoxes (e.g. provided and managed by a relational database system) and the integration of rules and rule-based reasoning could complete the outlined research project along dimensions that are currently observable as main lines of research in the DL field.

References

1. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
2. Alexander Borgida and Peter F. Patel-Schneider. A semantics and complete algorithm for subsumption in the classic description logic. *J. Artif. Intell. Res. (JAIR)*, 1:277–308, 1994.
3. Randal E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.
4. Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, second edition edition, 1996.
5. Krzysztof Goczyla, Teresa Grabowska, Wojciech Waloszek, and Michal Zawadzki. The cartographer algorithm for processing and querying description logics ontologies. In *Advances in Web Intelligence Third International Atlantic Web Intelligence Conference (AWIC 2005), Lodz, Poland*, pages 163–169, 2005.
6. Krzysztof Goczyla, Teresa Grabowska, Wojciech Waloszek, and Michal Zawadzki. Cartographic approach to knowledge representation and management in kasea. In *Proceedings of International Workshop on Description Logics (DL 2005), Edinburgh, Scotland, UK*, 2005.
7. Krzysztof Goczyla, Teresa Grabowska, Wojciech Waloszek, and Michal Zawadzki. The knowledge cartography - a new approach to reasoning over description logics ontologies. In *Theory and Practice of Computer Science, 32nd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2006), Merín, Czech Republic*, pages 293–302, 2006.
8. Jean Goubault. Proving with bdds and control of information. In *In Proceedings of the 12th International Conference on Automated Deduction (CADE) Nancy, France 1994*, pages 499–513, 1994.
9. Jean Goubault. A BDD-Based Simplification and Skolemization Procedure. *Logic Jnl IGPL*, 3(6):827–855, 1995.
10. Jean Goubault and Joachim Posegga. BDDs and automated deduction. In *International Syposium on Methodologies for Intelligent Systems*, pages 541–550, 1994.

[2] `http://owl1_1.cs.manchester.ac.uk/tractable.html`

11. J. F. Groote and H. Zantema. Resolution and binary decision diagrams cannot simulate each other polynomially. *Discrete Applied Mathematics*, 130(2):157–171, August 2003.
12. Jan Friso Groote and Olga Tveretina. Binary decision diagrams for first-order predicate logic. *J. Log. Algebr. Program.*, 57(1-2):1–22, 2003.
13. Ian Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.
14. Ian Horrocks. Applications of description logics: State of the art and research challenges. In Frithjof Dau, Marie-Laure Mugnier, and Gerd Stumme, editors, *Proc. of the 13th Int. Conf. on Conceptual Structures (ICCS'05)*, number 3596 in Lecture Notes in Artificial Intelligence, pages 78–90. Springer, 2005.
15. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 57–67. AAAI Press, 2006.
16. Ullrich Hustadt. *Resolution-Based Decision Procedures for Subclasses of First-Order Logic*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, November 1999.
17. J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 1–33, Washington, D.C., 1990. IEEE Computer Society Press.
18. Boris Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesität Karlsruhe (TH), Karlsruhe, Germany, January 2006.
19. A. Nonnengart and C. Weidenbach. Computing small clause normal forms. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 6, pages 335–367. Elsevier Science, 2001.
20. Guoqiang Pan, Ulrike Sattler, and Moshe Y. Vardi. Bdd-based decision procedures for the modal logic k. *Journal of Applied Non-Classical Logics*, 16(1-2):169–208, 2006.
21. Joachim Posegga. *Deduktion mit Shannongraphen für Prädikatenlogik erster Stufe.*, volume 51 of *DISKI*. Infix Verlag, St. Augustin, Germany, 1993.
22. Joachim Posegga and Peter H. Schmitt. Automated deduction with shannon graphs. *Journal of Logic and Computation*, 5(6):697–729, 1995.
23. Richard Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *ICCAD '93: Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design*, pages 42–47, Los Alamitos, CA, USA, 1993. IEEE Computer Society Press.
24. U. Sattler and M. Y. Vardi. The hybrid mu-calculus. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proceedings of the International Joint Conference on Automated Reasoning*, volume 2083 of *LNAI*, pages 76–91. Springer Verlag, 2001.
25. Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *In Proceedings of the International Joint Conference of Artificial Intelligence (IJCAI 1991)*, pages 466–471, 1991.
26. Seiichiro Tani, Kiyoharu Hamaguchi, and Shuzo Yajima. The complexity of the optimal variable ordering problems of shared binary decision diagrams. In *ISAAC '93: Proceedings of the 4th International Symposium on Algorithms and Computation*, pages 389–398, London, UK, 1993. Springer-Verlag.