DLP System Description

Peter F. Patel-Schneider
Bell Labs Research, Murray Hill, NJ, U.S.A.

Abstract

DLP (Description Logic Prover) is an experimental description logic knowledge representation system. DLP currently implements a superset of propositional dynamic logic as well as $\mathbf{K_{(m)}}$ and $\mathbf{KT_{(m)}}$. Although DLP is an experimental system, it nonetheless provides a fast satisfiability checker for the above propositional modal logics as well as a fast reasonder for knowledge bases.

Introduction

There is an effort underway to build a next-generation description logic system. This system will perform complete reasoning over a very expressive description logic, a logic that contains converse propositional dynamic logic. As part of this effort, I have built an experimental description logic knowledge representation system called DLP (for Description Logic Prover). DLP implements an expressive description logic, one that includes propositional dynamic logic as a subset. DLP provides a simple interface allowing users to build knowledge bases of descriptions in this description logic, but, as an experimental system, DLP does not have a full user interface.

Because of the correspondence between description logics and propositional modal logics, the description logic reasoner in DLP can serve as a reasoner for several propositional modal logics. As well as propositional dynamic logic, the logic underlying DLP contains fragments that are in direct correspondence to the propositional modal logics $\mathbf{K_{(m)}}$ and $\mathbf{K4_{(m)}}$. DLP provides an interface that allows direct satisfiability checking of formulae in $\mathbf{K_{(m)}}$ and $\mathbf{K4_{(m)}}$. Using a standard encoding, the interface also allows satisfiability checking of formulae in $\mathbf{KT_{(m)}}$ and $\mathbf{S4_{(m)}}$.

One of the purposes in building DLP was to investigate various optimisations for description logic systems. A number of these optimisations have appeared in various description logic systems [Baader *et al.*, 1992; Bresciani *et al.*, 1995; Horrocks, 1997]. In fact, much of DLP is a reimplementation of the ideas underlying FaCT [Horrocks, 1997]. As there is still need to investigate optimisations further and to develop new optimisation techniques, DLP has a number of compile-time options to select various description logic optimisations.

Most of the optimisations in DLP have to do with optimising subsumption checking, and thus correspond to potential optimisations for propositional modal logic satisfiability checking. DLP concentrates on this sort of optimisation and does not, as of yet, have a complete suite of optimisations that have to do with avoiding subsumption checking whenever possible, such as the taxonomy optimisations in Kris.

Logic

DLP implements the following description logic.

	Syntax	Semantics
Concepts	A	$\mathsf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
	$\top$	$\Delta^{\mathcal{I}}$
	$\bot$	$\emptyset$
	$\neg C$	$\Delta^{\mathcal{I}} - C^{\mathcal{I}}$
	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
	$\exists R.C$	$\{d \in \Delta^{\mathcal{I}} : R^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \neq \emptyset\}$
	$\forall R.C$	$\{d \in \Delta^{\mathcal{I}} : R^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$
	$\geqslant n\mathsf{P}$	$\{d \in \Delta^{\mathcal{I}} : \mid R^{\mathcal{I}}(d) \mid \ \geq n\}$
	$\leqslant n\mathsf{P}$	$\{d \in \Delta^{\mathcal{I}} : \mid R^{\mathcal{I}}(d) \mid \ \leq n\}$
Roles	P	$\mathsf{P}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
	$R \sqcup S$	$R^{\mathcal{I}} \cup S^{\mathcal{I}}$
	R/C	$R^{\mathcal{I}} \cap (\Delta^{\mathcal{I}} \times C^{\mathcal{I}})$
	$R \circ S$	$R^{\mathcal{I}} \circ S^{\mathcal{I}}$
	R^+	$\bigcup_{n>1} R^{\mathcal{I}^n}$

In the syntax chart A is an atomic concept; C and D are arbitrary concepts; P is an atomic role; R and S are arbitrary roles.

The semantics for DLP is a normal semantics for description logics, with a domain $\Delta^{\mathcal{I}}$ and an interpretation function $^{\mathcal{I}}$ for concepts and roles.

DLP's reasoner is sound and complete for this logic, modulo bugs in the implementation. It uses the now-standard method for subsumption testing in description logics, namely translating subsumption tests into satisfiability tests and checking for satisfiability using an optimised tableaux method.

Implementation

DLP was designed from the beginning to be an experimental system. As a result, much more attention has been paid to making the internal algorithms correct and efficient in the worst-case than to reducing constant factors. Similarly, the internal data structures have been chosen for their flexibility rather than having the absolute best modification and access speeds. Some care has been taken to make the internal data structures reasonably fast, however—there is considerable use of binary maps and hash tables instead of lists to store sets, for example.

This has extended even to the choice of implementation language and implementation philosophy of DLP. DLP is implemented in ML instead of a language like C so that it can be more-easily changed. There is some price to be paid for this, as ML does not allow some of the low-level optimisations possible in languages like C. Further, DLP is implemented in a mostly-functional fashion. The only non-functional portions of the satisfiability checker in DLP have to do with unique storage of formulae, and caching of several kinds of information. All this caching is monotone, i.e., it does not have be undone during a proof, or even between proofs.

Nonetheless, DLP is quite fast on several problem sets, including the Tableau'98 propositional modal logic comparison benchmark and several collections of hard random formulae in **K**. This was somewhat surprising to me, but probably just serves to confirm that in satisfiability checking is it more important to avoid work than to work fast.

Optimisation Techniques

Most of the optimisation techniques in DLP have already appeared in various description logic systems. The most complete description of these optimisations can be found in Ian Horrocks' thesis [Horrocks, 1997].

The basic algorithm in DLP is a simple tableau algorithm that searches for a model that demonstrates the satisfiability of a description logic description or, equivalently, a propositional modal logic formula. The algorithm incorporates the usual control mechanism to guarantee termination, including a check for equality of nodes to guarantee termination for transitive roles (modalities).

The optimisations in DLP include lexical normalisation, semantic branching, boolean constraint propagation, dependency-directed backtracking, caching, and heuristic guidance search.

Lexical Normalisation: Before the subsumption algorithm in DLP starts, incoming formulae are converted into a normal form, and common sub-formulae are uniquely stored. This conversion detects analytically satisfiable sub-formulae. It also allows values to be efficiently given to any sub-formula in the formula, not just propositionally atomic formulae. This can result in clashes being detected much earlier than would otherwise be the case.

Semantic Branching Search: DLP performs semantic branching search. This means that when DLP decides to branch on a formula, it picks an element of the formula and assigns that formula to true and false in turn instead of picking the elements of the formula and assigning them to true in turn. For example, semantic branching on $D_1 \sqcup D_2$ will result in branches containing $D_1, D_1 \sqcup D_2$ and $\neg D_1, D_1 \sqcup D_2$. Under syntactic branching the two branches would contain D_1 and D_2.

Semantic branching is guaranteed to explore each section of the search space at most once, as opposed to syntactic branching, and this is important in propositional modal logics as the generation and analysis of successors can result in large overlap in the search space when using syntactic branching.

Boolean Constraint Propagation: DLP performs simple boolean constraint propagation, looking for disjuncts in unexpanded disjunctions whose value is constrained due to values being known for the other disjuncts in the disjunction. For example, boolean constraint propagation deduces D_2 from $D_1 \sqcup D_2$ and $\neg D_1$, without branching on the disjunction. This technique can result in dramatic reductions in the search space, particularly in the presence of semantic branching.

Dependency Directed Backtracking: For every sub-formula, including clashes, DLP keeps track of which choice points lead to the deduction of that sub-formula. When backtracking to a choice point, DLP checks to see if the clash depends on that choice; if it does not, the alternative branch need not be considered, as it would just lead to the same clash. This technique, often called backjumping [Baker, 1995], can dramatically reduce the search space, but does have some overhead.

Caching: During a satisfiability check many sucessor nodes are generally created. These nodes can look considerably alike, so DLP caches and reuses their status. The caching in DLP is more complete than in other description logics systems, including FaCT.

Care has to be taken to ensure that caching does not interfere with the rest of the algorithm, particularly the determination of dependencies and loop analysis. Caching does require that information about each node generated be retained for a longer period of time than required for a basic depth-first implementation of the satisfiability checker. However, caching can produce dramatic gains in speed, particular for non-random formulae.

Heuristic Guided Search: There are many heuristic techniques that can be used to determine which sub-formula to branch on first. However, these techniques require considerable information to be kept or computed for each sub-formula of the unexpanded disjunctions. As DLP is implemented in a functional manner, it has to recompute this information whenever it wants to determine the best branch formula. Further, the heuristic techniques available have mostly been devised for non-modal logics and are not necessarily suitable for modal logics.

Nonetheless, DLP includes some simple heuristics to guide its search. These heuristics include the heuristics present in earlier description logic systems, but also include some new heuristics designed to interact better with DLP's other optimisations.

Performance

DLP has not been used in any actual applications, and as an experimental system, it is unlikely to receive any such use. DLP has been used to classify the Galen medical knowledge base [Rector *et al.*, 1997] with the portions it cannot represent removed. DLP performed capably on this knowledge base, creating the subsumption partial order in 210 seconds on a Sparc Ultra 1-class machine. FaCT also takes about 200 seconds for this task on a comparable machine.

DLP has also been tested on two sets of benchmarks, the Tableaux'98 comparison benchmarks [Heuerding and Schwendimann, 1996] and a collection of hard random modal formulae due to Hustadt and Schmidt [Hustadt and Schmidt, 1997]. DLP outperformed the existing description logic systems on the Tableaux'98 benchmarks and was competitive with FaCT, the fastest description logic system on the random formulae. More detail on DLP's performance is available in a paper discussing the performance of the various optimisations in DLP [Horrocks and Patel-Schneider, 1998].

Summary

As it is an experimental system, I did not expect DLP to be particularly fast on hard problems. It was gratifying to me that it is competitive with existing propositional modal reasoners including FaCT and KSAT [Giunchiglia and Sebastiani, 1996].

DLP is under continuing development. It currently handles propositional dynamic logic and may be extended to handle converse propositional dynamic logic. The ideas underlying DLP will be incorporated into a new description logic system that is currently being designed by a group of researchers including Enrico Franconi, Ian Horrocks, and myself. DLP is available via the WWW under
http://d----ut-p--tr.sal-atore.restsearch.bell-labs.com/user/pfps.

References

[Baader *et al.*, 1992] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)*, pages 270–281. Morgan-Kaufmann Publishers, San Francisco, CA, 1992. Also available as DFKI RR-93-03.

[Baker, 1995] A. B. Baker. *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results.* PhD thesis, University of Oregon, 1995.

[Bresciani *et al.*, 1995] P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive description logics: a preliminary report. In Gerard Ellis, Robert A. Levinson, Andrew Fall, and Veronica Dahl, editors, *Knowledge Retrieval, Use and Storage for Efficiency: Proceedings of the First International KRUSE Symposium*, pages 28–39, 1995.

[Giunchiglia and Sebastiani, 1996] F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for $\mathcal{ALC}$. In L. C. Aiello, J. Doyle, and S. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR'96)*, pages 304–314. Morgan Kaufmann Publishers, San Francisco, CA, November 1996.

[Heuerding and Schwendimann, 1996] A. Heuerding and S. Schwendimann. A benchmark method for the propositional modal logics k, kt, s4. Technical report IAM-96-015, University of Bern, Switzerland, October 1996.

[Horrocks and Patel-Schneider, 1998] I. Horrocks and P. F. Patel-Schneider. Comparing subsumption optimisations. In E. Franconi, G. De Giacomo, R. M. MacGregor, W. Nutt, C. A. Welty, and F. Sebastiani, editors, *Collected Papers from the International Description Logics Workshop (DL'98)*, 1998. To appear.

[Horrocks, 1997] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics.* PhD thesis, University of Manchester, 1997.

[Hustadt and Schmidt, 1997] U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. Technical Report MPI-I-97-2-003, Max-Planck-Institut Für Informatik, Im Stadtwald, D 66123 Saarbrücken, Germany, February 1997.

[Rector *et al.*, 1997] A. Rector, S. Bechhofer, C. A. Goble, I. Horrocks, W. A. Nowlan, and W. D. Solomon. The GRAIL concept modelling language for medical terminology. *Artificial Intelligence in Medicine*, 9:139–171, 1997.