Backtracking and Qualified Number Restrictions: Some Preliminary Results

Ian Horrocks
University of Manchester, Manchester, UK
Email: horrocks@cs.man.ac.uk

Abstract

Description Logics (DLs) are useful in the database domain, where they can be used to reason, e.g., about conceptual schemas and query containment. DL knowledge bases derived from relatively small UML diagrams have, however, proven difficult of impossible for state of the art DL systems to solve. We show that in the FaCT system this problem results from the failure of dependency directed backtracking to prune the search space caused by non-deterministic expansion of qualified number restrictions. We present an enhanced dependency directed backtracking technique that is able to deal more effectively with these constructs, along with empirical results demonstrating that, after the addition of this enhanced technique, the FaCT system is able to deal much more effectively with UML derived KBs.

1 Introduction

Description Logics (DLs) are a family of knowledge representation formalisms based on the notion of concepts (unary predicates or classes) and roles (binary relations or properties). DLs have been/are being used in wide range of applications including configuration [13] and ontological engineering (e.g., for the semantic web [4]).

More surprisingly, they have also proved useful in the database domain, where they can be used, e.g., to reason about conceptual schemas and query containment [3, 2, 10]. Their use in this context stems from the fact that most common data modelling formalisms (e.g., extended entity-relationship diagrams and UML [3]) can be captured by the n-ary DL $\mathcal{DLR}$, and the fact that an encoding of $\mathcal{DLR}$ in the $\mathcal{SHIQ}$ DL [2] allows state of the art DL systems (such as FaCT [11] and Racer [7]) to provide reasoning services for $\mathcal{DLR}$.

For expressive DLs such as $\mathcal{SHIQ}$, the worst case complexity of the basic inference problems (satisfiability and subsumption) is discouragingly high (at least EXPTIME-complete). Implementors of DL systems have, however, been very successful in overcoming this problem by developing sophisticated optimisation techniques that dramatically improve typical case performance. Systems employing such techniques have proved highly successful and have, e.g., been shown to be able to deal with large knowledge bases (KBs) derived from realistic applications [11, 6].

Recent work has, however, demonstrated these systems to be much less successful in reasoning with KBs derived from the encoding of UML models in $\mathcal{SHIQ}$ (via $\mathcal{DLR}$) [1]. In several cases, the encoding of even relatively small UML diagrams resulted in (still relatively small) KBs that could not be solved by any available DL system. In this paper we present the results of an investigation into the cause of this intractability in the FaCT system. We first show that it results from the non-deterministic expansion of *qualified number restrictions*, and in particular from the failure of FaCT's dependency directed backtracking (a key optimisation technique) to prune the resulting search space. We then present an enhanced dependency directed backtracking technique that is able to deal more effectively with these constructs, along with empirical test result demonstrating that, after the addition of this enhanced technique, the FaCT system is able to deal much more effectively with UML derived KBs.

2 Qualified number restrictions

It has already been observed in [1] that the intractability seems to derive from the use of cardinality constraints in the UML model. As a result of the encoding of $\mathcal{DLR}$ roles, these constraints lead to the occurrence of qualified number restrictions in the $\mathcal{SHIQ}$ KB, i.e., concepts of the form $\geqslant nR.C$ and $\leqslant nR.C$, where n is a nonnegative integer, R is a role and C is a (possibly complex) concept.

It is well known that number restrictions in general, and qualified number restrictions in particular, could be a serious cause of intractability due to the non-determinism that they introduce. In the tableaux algorithms used by state of the art DL systems, this non-determinism is manifested in the expansion rules dealing with $\leqslant nR.C$ concepts: the *choose*-rule and the $\leqslant$-rule.

These algorithms operate on trees where the nodes represent individuals and the edges represent roles. Each node is labelled with a set of concepts (we will use $\mathcal{L}(x)$ to denote the label of a node x) and each edge is labelled with a set of role names. A node y is said to be an R-neighbour of a node x if y is a successor of x and R is in the label of the edge connecting x to y, or if y is the predecessor of x, and R^- is in the label of the edge connecting y to x. The basic idea of the $\leqslant$-rule is that, given a node x such that $\leqslant nR.C \in \mathcal{L}(x)$, x has m R-neighbours that are "C-nodes", and $m > n$, then some of these neighbouring C-nodes must be identified (collapsed into single nodes) in order to reduce their number to n. Before this rule can be applied, however, the *choose*-rule must be used in order to explicate which of x's R-neighbours are C-nodes, i.e., which of them have the concept C in their labels.

The most obvious problem seems to be the $\leqslant$-rule, as there may be many different ways in which the m R-neighbour nodes can be collapsed into n nodes (in fact the number of n-partitions that can be formed from a set of size m), and this number grows rapidly with increasing values of n and m. Research has so far concentrated on minimising the search space resulting from such combinations, e.g., by using algebraic methods [5].

The UML derived KBs, however, only contain $\leqslant nR.C$ restrictions where n is equal to 1. At first sight these would appear to be relatively harmless, as the $\leqslant$-rule now becomes deterministic: all R-neighbouring C-nodes must be collapsed into a single node. The non-determinism introduced by the *choose*-rule remains, however, and in this case proves to be the cause of the intractability. When applied to a node x, the purpose of the *choose*-rule is to explicate the C status of the R-neighbours of x by non-deterministically adding C or $\neg C$ to the labels of R-neighbours not already containing one of these concepts.

3 Dependency directed backtracking

Dependency directed backtracking, in particular backjumping, is a key optimisation employed by DL systems [12]. The idea of backjumping is to tag concepts in the tableaux expansion tree to indicate any non-deterministic choices on which their existence depends. When the algorithm discovers a contradiction or *clash* (e.g., a node with both C and $\neg C$ in its label) the tags can be used to determine the most recent non-deterministic expansion where another choice could alleviate the cause of the clash.

For logics like $\mathcal{SHF}$, where disjunctions are the only source of non-deterministic expansion, the operation of backjumping is fairly straightforward. Roughly speaking, it operates as follows: at the start of the expansion, all concepts are tagged with the emptyset; when a concept is added to a node label as a result of one of the deterministic expansion rules, it is tagged with the union of the tags from the concepts that triggered the rule; when a concept is added to a node label by the i-th application of the $\sqcup$-rule, it is tagged with a set $\{i\}$; when a clash is detected, the algorithm jumps back to the j-th application of the $\sqcup$-rule, where j is the smallest number in the union of the tags of the concepts causing the clash.

In a $\mathcal{SHIQ}$ reasoner, the application of this technique becomes much more complex as the interaction of the *choose*-rule and the $\leqslant$-rule makes it difficult to determine the (optimal) dependencies of the concepts in the label of a collapsed node, and hence the cause of a clash (indirectly) resulting from an application of the $\leqslant$-rule. E.g.:

- The presence of concepts in the label of a collapsed node x may depend on qualifying concepts (possibly introduced by the *choose*-rule) in the labels of the nodes collapsed into x as well as on the $\leqslant nR.C$ concept that triggered the application of the $\leqslant$-rule.

- If a clash derives entirely from concepts coming from the same successor, however, then it is independent of the operation of either the *choose*-rule or the $\leqslant$-rule.
- $\mathcal{L}(x)$ only contains one copy of the qualifying concept C (the label is a set), and if this concept is involved in the unsatisfiability it may be difficult to determine the optimal dependency set.

Moreover, the algorithm may be sensitive to the order in which the *choose*-rule is applied to successors, whether the qualifying concept or its negation is tried first, and the order in which the successors of a node are explored. E.g., if the qualifying concept is C, and $\neg C$ leads (non-trivially) to unsatisfiability when added to neighbours of the node containing the $\leqslant nR.C$ concept, then choosing $\neg C$ first may lead to significant amounts of backtracking search (depending on the order in which successors are explored). On the other hand, if C is (non-trivially) unsatisfiable, then choosing C first may be equally disadvantageous.

Given the conjecture that qualified number restrictions with large values of n would rarely be encountered in realistic applications, and that the non-determinism resulting from such concepts would therefore be relatively insignificant, the initial implementation of FaCT's $\mathcal{SHIQ}$ reasoner paid little attention to these problems. When an application of the $\leqslant$-rule leads to a clash, the dependencies are computed from the tags of the concepts directly involved in the clash, plus those of the triggering $\leqslant nR.C$ concept and *all* the relevant qualifying concepts C.

3.1 A simple example

Consider the concept

$$\exists r.(B \sqcap C_1) \sqcap \ldots \sqcap \exists r.(B \sqcap C_n) \sqcap \leqslant 1r.D.$$

The concept is clearly satisfiable w.r.t. a KB $\mathcal{K} = \{B \sqsubseteq \exists r.(\forall r^-.D)\}$. If, however, the *choose*-rule selects the negated qualifying concept first (which seems to be a reasonable choice as it could obviate the need to apply the $\leqslant$-rule), then depending on the order in which successors are explored, this could cause the algorithm to search all 2^n possible ways of applying the *choose*-rule before discovering that selecting D in every case, followed by an application of the $\leqslant$-rule, results in a clash free expansion.

On the other hand, the same concept and the KB $\mathcal{K} = \{B \sqsubseteq \exists r.(\forall r^-.\neg D)\}$ could lead to similarly pathological behaviour if the *choose*-rule selects the non-negated qualifying concept first. Note that neither case involves number restrictions with a number greater than one.

The above examples may seem artificial, but as a result of the $\mathcal{DLR}$ encoding, KBs derived from UML models can contain just this kind of construction. Moreover, because of the cyclical nature of these KBs, and the fact that qualifying concepts may be complex and require significant additional expansion in order to discover any consequent unsatisfiability, the tableaux algorithm may build trees with several occurrences of such constructions in a single branch.[1] Inefficient exploration of the resulting search space, due to the failure of the backjumping optimisation, can cause excessive run time and/or excessive memory usage.[2]

4 Enhanced Backjumping

In order to address the problems discussed in Section 3, an enhanced version of the backjumping optimisation has been devised and implemented in the FaCT system (version 2.32.9). In order to simplify the algorithm (and the implementation), and to test its effectiveness w.r.t. the UML derived KBs, the enhanced optimisation currently only works with "functional" number restrictions, i.e., those where $n = 1$.

Enhanced backjumping works as follows:

1. When the $\leqslant$-rule causes a node y to be merged into a node x, the concepts from $\mathcal{L}(y)$ are tagged to indicate that they are derived from node y.
2. When an application of the $\leqslant$-rule leads to a clash, the algorithm determines which nodes the clashing concepts were derived from:
   - if the clash depends on more than one node, then the dependencies include those deriving from the concepts directly involved in the clash *plus* those deriving from the qualifying concepts in the relevant nodes and the relevant $\leqslant nR.C$ concept;
   - if the clash depends on only one node, then only the dependencies of the concepts directly involved in the clash are included.

In addition, the algorithm tries to apply the *choose*-rule to successor nodes in the same order in which they will be subsequently explored.

5 Empirical Evaluation

In order to facilitate experimentation, enhanced backjumping can be switched on or off using a flag; moreover, another flag controls whether the *choose*-rule first adds the qualifying concept or its negation.

The enhanced algorithm was tested using five KBs derived from UML diagrams, and the results are shown in Figure 1. In the column headings, E indicates the enhanced algorithm, B indicates the basic algorithm, Q indicates that the non-negated

[1] Improved blocking alleviates this problem, but does not eliminate it.

[2] The trace technique [8] cannot be used with $\mathcal{SHIQ}$, and state saving can easily exhaust system memory when exploring very large search spaces.

KB	E/Q	$E/\neg Q$	B/Q	$B/\neg Q$
hospital	38.69	2.59	T	2.78
library	2.09	0.28	3.43	0.34
restaurant	11.12	M	309.91	M
soccer	3.99	M	1,707.94	M
workshop	15.33	M	4,016.92	M

Figure 1: Results of evaluation with UML derived KBs

qualifying concept is tried first and $\neg Q$ indicates that the negated qualifying concept is tried first. The figures are the run time in seconds to classify the KB, with M indicating that system memory was exhausted and T indicating that the a time limit of 36,000s was exceeded. All the experiments were carried out using a 1GHz Pentium III processor with 512MB of RAM.

Several interesting points emerge from the results. Firstly, the behaviour of the algorithm is highly dependent on the order in which the *choose*-rule tries the negated or non-negated qualifying concept. In what seem to be relatively trivial problems (i.e., hospital and library, KBs derived from UML models in which there are no maximum cardinality constraints), trying the negated qualifying concept first is best, but for harder problems this strategy leads to trees/search spaces so large that they exhaust system memory. Trying the non-negated qualifying concept first generates smaller models (because many successors are then combined into a single node by the $\leqslant$-rule), but a much larger search space and thus much longer run times.

Secondly, the enhanced backjumping optimisation is very effective in pruning the larger search space generated by trying the non-negated qualifying concept first, with improvements typically ranging between 2 and 3 orders of magnitude. In the trivial cases, however, the improvement is not enough to offset the adverse effect caused by not trying the negated qualifying concept first.

6 Discussion

This experiment confirms what we know from earlier work: that backjumping is a crucial optimisation, and that many problems that are easy for algorithms using backjumping become difficult or impossible when it is absent or ineffective [9]. As Figure 1 shows, with the enhanced backjumping technique (and selecting non-negated qualifying concepts first) the FaCT system was able to deal with all of the UML derived KBs, some of which had previously been difficult or impossible to solve.

The current design and implementation of enhanced backjumping is very much a first prototype. There is still a great deal to be done regarding heuristic orderings, and the technique needs to be extended to deal with number restrictions where n is greater than 1. It seems likely that improvements in the design and implementation of the technique, and in particular improved ordering heuristics, will yield further significant performance improvements.

More importantly, however, it is clear that the interaction of (enhanced) backjumping with qualified number restrictions is sufficiently complex that it can no longer be considered an implementation detail: a precise analysis/specification of the algorithm and a formal proof of its correctness are urgently required. Such a specification and proof is the subject of ongoing work.

References

[1] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML Class Diagrams using Description Logic Based Systems. In *Proc. of the KI'2001 Workshop on Applications of Description Logics*. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-44/, 2001.

[2] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[3] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Description logics for conceptual data modeling. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*, pages 229–264. Kluwer Academic Publisher, 1998.

[4] Dieter Fensel, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, and Peter F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[5] V. Haarslev and R. Möller. Combining tableaux and algebraic methods for reasoning with qualified number restrictions. In *Proc. of the 2001 Description Logic Workshop (DL 2001)*, pages 152–161, 2001.

[6] Volker Haarslev and Ralf Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, 2001.

[7] Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, 2001.

[8] Bernhard Hollunder, Werner Nutt, and Manfred Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proc. of the 9th Eur. Conf. on*

Artificial Intelligence (ECAI'90), pages 348–353, London (United Kingdom), 1990. Pitman.

[9] I. Horrocks and P. F. Patel-Schneider. Comparing subsumption optimizations. 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)*, pages 90–94. CEUR (`http://ceur-ws.org/`), May 1998.

[10] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. How to decide query containment under constraints using a description logic. In *Proceedings of the 7th International Conference on Logic for Programming and Automated Reasoning (LPAR'2000)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2000.

[11] Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.

[12] Ian Horrocks and Peter F. Patel-Schneider. Optimizing description logic subsumption. *J. of Logic and Computation*, 9(3):267–293, 1999.

[13] Deborah L. McGuinness and Jon R. Wright. An industrial strength description logic-based configuration platform. *IEEE Intelligent Systems*, pages 69–77, 1998.