Abstract

Subsumption algorithms are used in resolution oriented theorem proving to eliminate redundant clauses from the search space. In a recent paper, the authors have introduced a new subsumption algorithm DC (Division into Components) which is much more efficient than the standard algorithms. In the present paper two new results are stated. First, an exponential lower bound for DC is presented. It is shown, that in certain cases DC is exponential in time, while IDC is only polynomial. However we prove when DC has polynomial time complexity, then also IDC is polynomial.

1. INTRODUCTION

The elimination of redundant clauses by subsumption is an essential feature of almost all efficient theorem proving systems. If C and D are clauses, one defines C SUBS D iff \(|C| \leq |D|\) and there is a substitution \(\theta\) such that \(C\theta \subseteq D\) holds (because clauses may be considered as disjunctions of literals, C is more general than D if such a \(\theta\) exists). The subsumption rule consists in removing clauses D if a C is found with C SUBS D (the finding of C may be determined by certain theorem proving strategies).

Because subsumption considerably reduces the search space, its application is of central importance for the most powerful theorem provers ([1],[2]). Despite of the usefulness of subsumption, the subsumption problem itself is relatively hard (NP-complete [3]). Two well known subsumption algorithms (decision algorithms for C SUBS D) are the algorithm of Chang and Lee (CL,[4]) and the algorithm of
Stillman (ST [5]). While CL uses resolution and a level saturation strategy, ST is a left to right matching algorithm with backtracking (a PROLOG-like strategy) trying to translate the literals of C into literals of D by substitution of variables.

In a recent paper [6] the authors analyzed the complexity of CL and ST; it turned out, that both algorithms have a very high worst-case time complexity. To overcome this disadvantage the authors developed a new algorithm DC "Division into Components" [6], whose complexity is significantly better.

The aim of this paper is a further improvement of DC by definition of a new algorithm IDC (improved DC). In [6] only an upper bound for the worst-case time complexity of DC was specified; here, we derive an exponential lower bound (subjected to the restriction, that the number of variables in C is twice the number of literals in C). Furthermore, it is shown, that in certain cases IDC is polynomial, but DC is exponential; on the other hand, the computing time of IDC is always polynomially bounded by that of DC.

2. NOTATION AND DEFINITIONS:

Elementary definitions (atomic formula, literal, clause, substitution, unification, ...) can easily be found in the standard literature on automated theorem proving [4,7]. If \( \theta \) is a substitution (in first-order logic) and C is a clause then \( C\theta \) denotes the clause after application of \( \theta \) to the literals of C. We write \( a \in S \) if \( a \) is a member of the set \( S \) and \( |S| \) for the cardinality of \( S \).

**Definition 2.1:** A clause \( C \) subsumes a clause \( D \) iff \( |C| \leq |D| \) and there is a substitution \( \theta \) such that \( C\theta \subseteq D \); notation: \( C \text{ SUBS } D \).

In the book of Loveland [7] the concept defined above is called \( \theta \)-subsumption, while \( C \) subsumes \( D \) is defined as equivalent to \( C \rightarrow D \) (the two concepts do not coincide). Because CL, ST, DC and IDC (which is developed in this paper) are all decision algorithms for \( \theta \)-subsumption (in the sense of [7]) we simply use the term "subsumption" and write \( C \text{ SUBS } D \). It should be remarked, that the condition \( |C| \leq |D| \) is not necessary to prove the results of this paper.