Algorithms for Logic-Based Abduction

Anja Remshagen[1] and Klaus Truemper[2]

[1] State University of West Georgia, Computer Science Department, 1600 Maple Street, Carrollton, GA 30118, USA
anja@westga.edu

[2] University of Texas at Dallas, Computer Science Program, EC31, Box 830688, Richardson, TX 75083-0688, USA
klaus@utdallas.edu

Abstract. Recent advances in solvers for the satisfiability problem (SAT) include compilation of CNF formulas as well as learning techniques. We show that these techniques are suitable for some forms of logic-based abduction problems at the second and higher level of the polynomial hierarchy. The approach leads to practical solution algorithms for small-scale problem instances.

1 Introduction

This work has been motivated by problems arising in diagnosis. An important model of diagnosis is the following formulation of logic-based abduction. The knowledge base is given as two theories R and S of propositional logic. Theory R formulates valid relationships among hypotheses represented by a variable set Q. Theory S models the knowledge domain including the negation of the manifestations. The theories R and S share the variable set Q. We want to find an explanation, given as a truth assignment for the variables of Q, which is consistent with R but not with S. We refer to this problem as Q-ALL SAT.

Transformations show that Q-ALL SAT is logically equivalent to the propositional abduction problem defined by Eiter and Gottlob [3]. Thus, Q-ALL SAT is Π_2^{p}-complete.

We also encounter the problem where a cost is associated with the truth assignment to each variable in R, and an explanation inducing minimum total cost is searched for. Eiter and Gottlob [3] proved that an equivalent form of this problem is Δ_3^{p}-complete.

A third problem considers worst-case scenarios. We want to determine the minimum cost of a cure or repair in the worst possible case. Here a truth assignment consistent with S represents a possible cure or repair. A cost is associated with some of the variables of S. Thus, we have to determine a truth assignment to the variables of Q that is consistent with R and S and whose minimum total cost induced in S are maximum among all such assignments.

Q-ALL SAT is a special form of a quantified Boolean formula which contains two subformulas in conjunctive normal form (CNF). First solution algorithms for quantified Boolean formulas have been constructed, for example, by Cadoli, Giovanardi, and Schaerf [2] and Rintanen [7]. These algorithms consider the case where all quantifiers constitute a prefix of a CNF formula. They cannot be applied to logic-based abduction. We will propose algorithms for Q-ALL SAT and its related problems that can take advantage of any efficient SAT solver and of any solver for the related minimization problem. For examples of SAT solvers, see GRASP (Marques-Silva and Sakallah [4]), SATO3 (Zhang [10]), or relsat4 (Bayardo and Schrag [1]). For an example of an effective solver for minimization, see Truemper [8].

Since a diagnosis system makes decisions based on the same theories, it is a suitable application for compilation and learning. We describe a solution algorithm for Q-ALL SAT and its related problems and show how learning can be used. We also outline a compiler for a common case of Q-ALL SAT using decomposition of the underlying theory.

2 Three Logic-based Abduction Problems

If a truth assignment for some of the variables of a CNF formula T is consistent with T, we call the assignment *T-consistent*. Otherwise, the assignment is *T-inconsistent*. An instance of MINSAT is given by a CNF formula T and, for each variable x of T, by a rational *True*-cost c_x. The MINSAT problem demands that one either determines T to be unsatisfiable or one produces a satisfying solution whose total cost—that is $\sum_{x \ni \{x=True\}} c_x$—is minimum. A MINIMAL SAT instance is a pair (T, W) where T is a CNF formula and W is a subset of the variables. A solution of (T, W) is a satisfying truth assignment for the variables that sets a minimal set of variables in W to *True*, or we determine that no such solution exists.

We treat three problems called Q-ALL SAT, Q-MIN UNSAT, and Q-MAX MINSAT and defined in Truemper [9]. An instance of each problem consists of two satisfiable CNF formulas R and S. The formulas R and S have common variables given by a variable set Q. The remaining variable set of R is denoted by X and the remaining variable set of S by Y. The sets X and Y are disjoint. In addition, an instance of Q-MIN UNSAT assigns *True*-costs to the variables of R. An instance of Q-MAX MINSAT assigns *True*-costs to the variables of S contained in set Y. Q-ALL SAT is the problem to decide if the quantified formula

$$\forall Q\, (\exists X\, R \Rightarrow \exists Y\, S) \tag{1}$$

is *True*. If the formula evaluates to *False*, a truth assignment for the variables of Q must be provided that is R-consistent but not S-consistent.

In the case of Q-MIN UNSAT and Q-MAX MINSAT, one also has to decide if formula (1) evaluates to *True*. If the formula is *False*, then Q-MIN UNSAT demands that one finds an R-consistent and S-inconsistent truth assignment for Q with the following property. Define the *induced R-cost* of an R-consistent assignment for Q to be the minimum total cost of the MINSAT instance R where the assignment for Q is enforced. Then the wanted assignment in Q-MIN UNSAT must have minimum induced R-cost.

If formula (1) evaluates to *True*, Q-MAX MINSAT demands that one finds an R- and S-consistent assignment for Q with maximum induced S-cost, where induced S-cost is defined analogously to induced R-cost.

We define two CNF formulas S_1 and S_2 derived from S. The formula S_1 is obtained by removing from each clause in S all literals that are variables or negated variables of Y. S_2 is obtained analogously by removing all literals that are variables or negated variables of Q. The CNF formulas S_1 and S_2 might contain multiple occurrences of the same clause since we do not consider a CNF formula as a set of clauses. For each nonempty clause c in S_1 we introduce a new variable z_c. The set of all new variables is denoted by Z. The clause c is part of a clause $c \vee c'$ in S where c' is the corresponding clause of S_2. By replacing each such clause c' in S_2 by the clause $c' \vee z_c$, we obtain a new CNF formula denoted by $S_2 \,|\, I$.

3 The Algorithm

We first consider Q-ALL SAT. Formula (1) suggests to determine successively all R-consistent truth assignments for Q and to check if the assignments are S-consistent. It is sufficient to consider only R-consistent assignments that satisfy a minimal set of clauses of S_1. We obtain these assignments by extending R by the following CNF formula T. For each nonempty clause c of S_1 with the literals $q_1, q_2, \ldots, q_k$, formula T contains the clauses

$$\begin{array}{c} \neg q_1 \vee z_c \\ \neg q_2 \vee z_c \\ \vdots \\ \neg q_k \vee z_c \;\; . \end{array} \tag{2}$$

It is easy to verify that any T-consistent truth assignment for Q assigns *True* to z_c if clause c is satisfied under that assignment for Q. Thus, an $R \wedge T$-consistent truth assignment for Q that sets a minimal set of variables to *True* is R-consistent and satisfies a minimal set of clauses of S_1. Equivalent to formula (1) is the formulation

$$\forall Q \, (\forall Y \, \neg S \Rightarrow \forall X \, \neg R) \;\; . \tag{3}$$

Formula (3) suggests to determine the S-inconsistent truth assignments for Q and to check if they are R-consistent. Instead, we will compute S-consistent truth assignments for Y that satisfy a maximal set of clauses of S.

To rule out any truth assignment for Q that has been processed already but did not yet lead to a solution, we add a new clause to $R \wedge T$. The algorithm Solve Q-All SAT works as follows.

Solve Q-ALL SAT
Input: Satisfiable CNF formula R, CNF formula S, common variable set Q of R and S.
Output: Either: "All R-consistent assignments for Q are S-consistent." Or: An R-consistent assignment α for Q which is not S-consistent.

1. Let H be the CNF formula with variable set Z and without any clauses.
2. Solve the MINIMAL SAT instance $(R \wedge T \wedge H, Z)$. If $R \wedge T \wedge H$ is unsatisfiable, output "All R-consistent assignments for Q are S-consistent," and stop. Otherwise, let (α, β, δ) be a solution where α, β, δ are assignments for Q, X, Z, respectively.
3. For all variables z_i of Z that are set to *False* by δ, fix z_i to *False* in $S_2 \,|\, I$. Denote the resulting formula by $S_2 \,|\, I^*$.
4. Solve the MINIMAL SAT instance $(S_2 \,|\, I^*, Z)$. If $S_2 \,|\, I^*$ is unsatisfiable, output assignment α, and stop. Otherwise, let (γ, δ^*) be a solution where δ^* is a truth assignment for Z.
5. Add the clause $\neg z_{i_1} \vee \neg z_{i_2} \vee \ldots \vee \neg z_{i_k}$ to H where δ^* assigns *True* to exactly the variables $z_{i_1}, z_{i_2}, \ldots, z_{i_k}$ in Z. If δ^* does not assign *True* to any variable, add the empty clause to H. Go to Step 2.

The truth assignment α in the R-consistent assignment (α, β) of Step 1 satisfies exactly the clauses c of S_1 where δ sets z_c to *True*. Thus, we need to find a truth assignment γ for Y that satisfies all clauses in S_2 that correspond to a clause c in S_1 where δ sets z_c to *False*. We eliminate these variables z_c in Step 3 to enforce such an assignment.

In Steps 2 and 4, we solve an instance of MINIMAL SAT. We can either use a SAT solver to determine first a solution. Then we try tentatively for each variable of Z that has been set to *True* if the truth assignment for Q is still $R \wedge T \wedge H$-consistent if the variable is set to *False*. Another approach is to assign *True*-cost 1 to each variable of Z and *True*-cost 0 to all other variables in $R \wedge T \wedge H$. A solution of this MINSAT instance will set a minimum number of variables to *True* and thus provides a solution of the MINIMAL SAT instance as well. In the first case, we have to solve repeatedly SAT instances derived from the same CNF formula by fixing some variables. A backtracking algorithm that learns clauses during the solution process can speed up the algorithm substantially. Also in the case of MINSAT, learning of clauses can be applied. For details of learning, see Remshagen [5] and Remshagen and Truemper [6]. Observe that learned clauses and simplifications of the respective CNF formula are valid in any preceeding execution of Step 2 and 4, respectively, since we only add constraints to $R \wedge T \wedge H$. The CNF formula $S_2 \,|\, I$ remains unchanged.

In general, the CNF formula H can grow exponentially. Otherwise, the above algorithm would prove that Q-ALL SAT is in Δ_2^{p}. This is unlikely since Q-ALL SAT is Π_2^{p}-complete.

Q-MIN UNSAT is handled similarly. Since a solution must have minimum total cost in R, we solve the MINSAT problem for $R \wedge T \wedge H$ in Step 2. The solution algorithm for Q-MAX MINSAT differs from Solve Q-ALL SAT in Step 4. Instead of a MINIMAL SAT instance, we solve the MINSAT instance $S_2 \,|\, I^*$ where each variable in Z has cost 0. In the case of unsatisfiability, the algorithm continues. Otherwise, we have a satisfying solution for $S_2 \,|\, I^*$. We modify the assigned value for each variable in Z by setting it to *False* if the resulting assignment still satisfies $S_2 \,|\, I^*$. The algorithm also keeps track of the current largest induced S-cost.

4 A Compiler-based Solution Algorithm

In a common form of logic-based abduction, the formulas R and S are derived from the same theory given as a CNF formula U with hypotheses represented by variable set Q. For each instance, some variables in U are set to given truth values. The resulting formula is R. Fixing the manifestations, that is setting some additional variables to given truth values, results in S. Let T be a CNF formula and W a subset of the variables of T. Then any truth assignment for W induces a *subrange* vector r of T for variable set W. Vector r has an entry r_c for each clause c of T. If the assignment for W satisfies c, then $r_c = 1$; otherwise, $r_c = 0$ holds. A subrange vector r is *feasible for* T if there exists a T-consistent truth assignment for W that induces r.

Truemper [8] describes a class of CNF formulas that can be decomposed in few so-called Boolean closed formulas. In such a case, the number of subrange vectors is restricted by a polynomial in the number of variables of U. A compiler can apply a learning scheme to U and can compute all subrange vectors of U for Q. Let $\mathcal{R}$ denote the set of the subrange vectors.

The solution algorithm for Q-ALL SAT is now simplified to the following process. We continue removing the smallest vector from $\mathcal{R}$ until a vector r corresponds to a subrange vector of R for Q. If r is a subrange vector of S as well, all vectors $v' > v$ are removed from $\mathcal{R}$, and the process continues. Otherwise, vector v corresponds to an R-consistent and S-inconsistent truth assignment for Q. In the worst case, checking the subrange property of a vector for the formula R or S corresponds to solving the SAT problem for a reduced CNF formula R or S, respectively. Thus the complexity of this class of Q-ALL SAT instances is scaled down to Δ_2^p, since the size of $\mathcal{R}$ is polynomial in the number of variables of U.

5 Summary

Many practical applications in diagnosis can be formulated by one of the three introduced forms of logic-based abduction. Instances of restricted size can be solved effectively by the proposed solution algorithms. The algorithms are easy to implement and take advantage of state of the art SAT and MINSAT solvers. They gain efficiency by compilation and learning.

References

1. Bayardo Jr., R. J., Schrag, R.: Using CSP Look-Back Techniques to Solve Real-World SAT Instances. Proceedings of the Fourteenth National Conference on Artificial Intelligence (1997) 203–208
2. Cadoli, M., Giovanardi, A., Schaerf, M.: An Algorithm to Evaluate Quantified Boolean Formulae. Proceedings of the Fifteenth National Conference on Artificial Intelligence (1998)
3. Eiter, T., Gottlob, G.:, The Complexity of Logic-based Abduction. Journal of the Association for Computing Machinery **42**, (1995) 3–42

4. Marques-Silva, J. P., Sakallah, K. A.: Conflict Analysis in Search Algorithms for Propositional Satisfiability. Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (1996)
5. Remshagen, A.: Learning for SAT and MINSAT, and Algorithm for Quantified SAT and MINSAT. PhD Thesis, University of Texas at Dallas (2001)
6. Remshagen, A., Truemper, K.: Learning in a Compiler for MINSAT Algorithms. Theory and Practice of Logic Programming, in review (2002)
7. Rintanen, J.: Improvements to the evaluation of quantified Boolean formulae. Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (1999) 1192–1197
8. Truemper, K.: Effective Logic Computation. Wiley (1998)
9. Truemper, K.: Design of Intelligent Systems. in preparation
10. Zhang, H.: SATO: an Efficient Propositional Prover. Proceedings of the International Conference on Automated Deduction (1997) 272–275