



Generating all maximal models of a Boolean expression[☆]

Dimitris J. Kavvadias^a, Martha Sideri^b, Elias C. Stavropoulos^{c,*}

^a University of Patras, Department of Mathematics, GR-265 00 Patras, Greece

^b Athens University of Economics and Business, Department of Computer Science, GR-104 34 Athens, Greece

^c University of Patras, Computer Engineering & Informatics Department, GR-265 00 Patras, Greece

Received 20 June 1999; received in revised form 7 December 1999

Communicated by S. Zaks

Abstract

We examine the computational problem of generating all maximal models of a Boolean expression in CNF. We give a resolution-like method that reduces the unnegated variables of an expression while preserving its set of maximal models. We present an output-polynomial algorithm for the 2CNF case and we show that the problem cannot be solved in output-polynomial time in the case of Horn expressions, unless $P = NP$, despite an affinity of this case to the recently subexponentially solved transversal hypergraph problem. The problem is of course trivial for 1-valid and anti-Horn expressions, and open for exclusive-ors; it is NP-hard in all other cases. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Algorithms; Computational complexity; Output-polynomial; Maximal model generation; Transversal hypergraph

1. Introduction

There are two useful and informative styles of representing a Boolean expression: the *syntactic* one, in which the expression is given by its logical representation, often in conjunctive normal form (CNF), and the *semantic* one, in which the expression is given in terms of its *models* (that is, its satisfying truth assignments). The computational complexity of translating between these two representations has recently been the subject of extensive study (see, for example, [14, 15,7]). An interesting problem in this regard is, given a CNF expression, to generate all its models. Since the number of models may grow exponentially, one cannot hope to solve this problem in polynomial time,

but instead in time polynomial in the size of the input and the output. There has been a recent surge of interest in such algorithms, called *output-polynomial algorithms* [13,17]. The most notable result in this area is an incrementally output-subexponential algorithm for generating all minimal transversals of a hypergraph, a central and long unresolved question [8,12,6].

One is often interested not in all models of a Boolean expression, but in the most “representative” or “natural” ones. For example, concepts of propositional circumscription [3] and minimal diagnosis [5] restrict interest in the set of models of an expression that are *minimal*. Moreover, finding a *maximal* model is an essential task in model-preference default inference [20]. A model is maximal if there is no other model that is componentwise greater than or equal to it; minimal is the opposite. Complexity questions related to minimal or maximal models have been discussed in [3,4]. This consideration suggests the fol-

[☆] Research supported by the University of Patras Research Committee (Project Caratheodory under contract no. 1939).

* Corresponding author.

lowing computational problem (we concentrate on the version regarding maximal models; with rare exceptions pointed out, the corresponding problem with minimal models has the same complexity): Given a Boolean expression, generate all its maximal models (GENERATE ALL MAXIMAL).

Naturally, for general expressions the problem cannot be solved in output-polynomial time, unless $P = NP$. Even in the special case of all-negative clauses (no positive literal appears in the expression), the problem is equivalent to the generation of all minimal transversals of a hypergraph (TRANSVERSAL HYPERGRAPH problem), arguably the most important problem in this area. (For all-positive clauses it is, of course, trivial.) It was shown in [8] that TRANSVERSAL HYPERGRAPH can be solved in output-subexponential time while a heuristic algorithm with good performance in practice was recently presented in [16]. Certain algorithmic approaches to GENERATE ALL MINIMAL are presented in [1]. We examine the case of Horn expressions that are in some sense the next step from the all-negative ones. Despite the superficial similarity between GENERATE ALL MAXIMAL for Horn expressions and TRANSVERSAL HYPERGRAPH, we show that the former is NP-hard (Theorem 4). However, we present an output-polynomial algorithm that generates all maximal models in the 2CNF case. This algorithm uses a technique that may be useful in practice for reducing the complexity of the general problem: We give a resolution-like procedure (Theorem 1) that reduces the number of positively-occurring variables, and thus brings the problem closer to the TRANSVERSAL HYPERGRAPH, without changing the set of maximal models. A related procedure, but within the context of query answering in circumscription, was given in [18]. Finally, GENERATE ALL MAXIMAL for exclusive-or expressions is left as a very interesting open problem.

2. Definitions

Let $X = \{x_1, \dots, x_n\}$ be a set of variables. A *literal* is either a variable (*positive literal*) or its negation (*negative literal*). A *clause* is a disjunction of one or more literals. A *Boolean expression* in *conjunctive normal form* (CNF) is a set (conjunction) of clauses $\phi = \{C_1, C_2, \dots, C_k\}$. A clause is called *0-valid* (respectively, *1-valid*) [19] if it is satisfied by the all ze-

ros (respectively, all ones) truth assignment. A clause is called *Horn* if it contains at most one positive literal. A *Horn expression* is a set of Horn clauses. If all clauses are purely negative, we have a *purely negative expression*. An *anti-Horn* clause has at most one negative literal. We also consider two other special cases of expressions: a *2CNF expression* is a CNF expression with at most two literals per clause. Finally, an *exclusive-or* expression is a system of equations of the form $Ax = b \pmod 2$, where A is a 0–1 matrix, b a 0–1 vector, and x is a vector of variables. We may consider such a system as a conjunction of “clauses”, where each equation is regarded as a clause.

A *model* m of a CNF expression ϕ is a satisfying truth assignment of ϕ (we write $m \models \phi$). For an exclusive-or expression, a model is a 0–1 vector that satisfies all equations. We write $m(x) = 1$ (or $m(x) = 0$) to denote that m assigns 1 (or 0, respectively) to variable x . If m and m' are models, we say that $m \leq m'$ if $m(x) = 1$ implies $m'(x) = 1$. Models m and m' are said to be *incomparable* if neither $m \leq m'$ nor $m' \leq m$ holds. A model m is *maximal* if no other model m' such that $m < m'$ exists. Minimal is the opposite. Let $models(\phi)$ denote the set of models of ϕ and $maximal(\phi)$ denote the set of all maximal models of ϕ . Obviously, all models in $maximal(\phi)$ are pairwise incomparable. Deciding whether $models(\phi) \neq \emptyset$ is the well-known *satisfiability* problem (SAT) which in its generality is NP-complete [10]. A Dichotomy Theorem was shown in [19], establishing that the only polynomial-time solvable cases of SAT are precisely the six ones introduced above; in all other cases SAT is NP-complete. Schaefer’s result is actually more general and includes the class of *generalized formulas* whose clauses’ truth value is determined according to an arbitrary truth table, or Boolean relation, of bounded arity k . It was also shown in [15] that the problem of generating *all* models of a Boolean expression has a dichotomy theorem akin to that of [19].

Complexity theory has traditionally focused on decision problems. Certain computational problems, however, require output that may be exponentially larger than the input. The only hope for such problems is the existence of a polynomial algorithm in *both* the input and the output. Such algorithms are called *output-polynomial*. A slightly stronger requirement is that the algorithm generates a new output bit

in time polynomial in the input *and the output so far*. These latter algorithms are called *incrementally output-polynomial* [13,17]. One of the most interesting problems in this regard is the TRANSVERSAL HYPERGRAPH problem defined next: A *hypergraph* \mathcal{H} is a family of sets $\{S_1, \dots, S_n\}$ over some universe [2]. The *transversal hypergraph* of \mathcal{H} , $\text{tr}(\mathcal{H})$, is the family of all minimal hitting sets of \mathcal{H} , that is, all sets T such that

- (a) T intersects all n sets of \mathcal{H} , and
- (b) no proper subset of T does.

A hypergraph is *simple* if no hyperedge is totally contained in another one. TRANSVERSAL HYPERGRAPH is the problem of generating $\text{tr}(\mathcal{H})$ given a simple hypergraph \mathcal{H} , and is a common subproblem in many applications, including databases [11], distributed systems [9], etc. The precise complexity of this problem is still unknown. However, it was shown in [8,12] that it can be solved in time $O(c^{\log c})$, where c is the combined size of the input and the output. Recently, a heuristic algorithm with good performance in practice was presented in [16].

The main concern of this paper is the following computational problem: Given a Boolean expression ϕ , generate all its maximal models (GENERATE ALL MAXIMAL). Notice that for purely negative expressions GENERATE ALL MAXIMAL is precisely the TRANSVERSAL HYPERGRAPH, the sets S_i being the clauses and each element in S_i corresponding to a negated variable in the clause. Thus, the generation of all maximal models of a purely negative CNF expression can be done in incremental subexponential time by the algorithm of [8].

3. Elimination of positive variables

The presence of positive literals in a Boolean expression forbids the direct application of the algorithm of [8]. In this section we present a procedure that transforms an arbitrary CNF expression ϕ into a purely negative one while preserving its maximal models. Thus, the generation of all maximal models of ϕ can be done by the algorithm of [8].

Let us view ϕ as a set of clauses $S = \{K_1, K_2, \dots, K_{|S|}\}$ defined on the set of variables X and let y be an arbitrary variable in X . Then, S can be partitioned into three subsets S_y , $S_{\bar{y}}$, and A_y , where

y appears unnegated, negated, and does not appear at all, respectively. Hence, $\phi = S_y \cup S_{\bar{y}} \cup A_y$. Let $K_i = (C_i \vee y)$ be a clause in S_y , $i = 1, \dots, |S_y|$, and $K_j = (D_j \vee \bar{y})$ be a clause in $S_{\bar{y}}$, $j = 1, \dots, |S_{\bar{y}}|$. Here by C_i and D_j we denote the disjunction of all other variables (apart from y) that appear in K_i and K_j , respectively. The *resolvent* of K_i and K_j with respect to y is defined as the clause $(C_i \vee D_j)$. Finally, let R_y be the set of resolvents with respect to y of all pairs of clauses in S_y and $S_{\bar{y}}$, that is

$$R_y = \{C_i \vee D_j \mid (C_i \vee y) \in S_y, (D_j \vee \bar{y}) \in S_{\bar{y}}\}.$$

Consider now the expression $\phi_y = R_y \cup S_{\bar{y}} \cup A_y$. It is easy to see that $\text{models}(\phi) \subseteq \text{models}(\phi_y)$ (by the resolution rule) but the converse does not hold. Although ϕ and ϕ_y may have different sets of models, the next theorem states that their sets of maximal models are identical:

Theorem 1. *Let ϕ be an arbitrary CNF expression and ϕ_y be the CNF expression produced from ϕ as described above. Then $\text{maximal}(\phi) = \text{maximal}(\phi_y)$.*

Proof. We first show that $\text{maximal}(\phi) \subseteq \text{maximal}(\phi_y)$. Let m be a maximal model of ϕ . Then, m is a model of ϕ_y . To show that $m \in \text{maximal}(\phi_y)$, assume to the contrary that there exists a maximal model m' of ϕ_y such that $m' > m$. If $m'(y) = 1$, then m' satisfies all clauses in S_y and therefore $m' \in \text{models}(\phi)$. This contradicts the fact that $m' > m$ and $m \in \text{maximal}(\phi)$. Thus, $m'(y) = 0$. Hence, there exists at least one clause $(D_k \vee \bar{y}) \in S_{\bar{y}}$ such that D_k is not satisfied by m' (otherwise, the vector m'' with $m''(y) = 1$ and $m''(x) = m'(x)$ for every $x \neq y$, would be a model of ϕ_y such that $m'' > m' > m$). Model m' satisfies all resolvents $(C_i \vee D_k)$, $i = 1, \dots, |S_y|$, in R_y and therefore all clauses C_i in S_y . Thus, $m' \in \text{models}(\phi)$. However, this can not hold since $m' > m$ and $m \in \text{maximal}(\phi)$. Hence, $m \in \text{maximal}(\phi_y)$.

For the opposite direction, let $m \in \text{maximal}(\phi_y)$. If $m(y) = 0$ then there exists at least one clause $(D_k \vee \bar{y})$ in $S_{\bar{y}}$ that is not satisfied by m (otherwise, the vector m' such that $m'(y) = 1$ and $m'(x) = m(x)$ for every $x \neq y$, would be a model of ϕ_y such that $m' > m$). Since m satisfies all resolvents in R_y , m satisfies all $(C_i \vee D_k)$, $i = 1, \dots, |S_y|$, and therefore m satisfies all C_i in S_y . Hence, m is a model of ϕ . If $m(y) = 1$ then again m satisfies all clauses of S_y . In any case,

a maximal model m of ϕ_y is a model of ϕ . Suppose now that $m \notin \text{maximal}(\phi)$. Then there exists a model m' such that $m' > m$ and $m' \in \text{maximal}(\phi)$. But $m' \in \text{models}(\phi_y)$. Hence, there exists a model m' of ϕ_y such that $m' > m$, a contradiction. \square

Repeated applications of Theorem 1 for every variable having at least one unnegated appearance will result in a CNF expression which has the same set of maximal models as the given one but no unnegated variables. Thus, the given instance is transformed into a purely negative CNF which is equivalent to the TRANSVERSAL GENERATION problem and can be solved by the algorithm of [8]. However, the question emerging here is how big the produced expression can become after repeated applications of the theorem. We next show that the final set of clauses can be exponentially large even for simple Horn expressions.

Proposition 2. *There is a Horn expression ϕ for which the smallest purely negative CNF expression with the same set of maximal models is exponentially larger than ϕ .*

Proof. Consider the irredundant Horn expression defined on $3n$ variables, with $2n + 1$ clauses:

$$\phi = \left(\bigwedge_{i=1}^n (x_i \vee \bar{y}_{i,1}) \wedge (x_i \vee \bar{y}_{i,2}) \right) \wedge (\bar{x}_1 \vee \dots \vee \bar{x}_n).$$

We apply Theorem 1 to ϕ for the n variables x_1, \dots, x_n in turn, that have positive appearance. Since all $y_{i,j}$ are different, no resolvent will be subsumed by any other clause during the whole process. By this we mean that no resolvent will be a logical implication of any other clause. Thus, once a resolvent is constructed, it remains until the end. Initially, there is only one clause that contains \bar{x}_1 . The application of Theorem 1 for variable x_1 results in 2 resolvents containing the literal \bar{x}_2 . Thus, at the resulting irredundant expression \bar{x}_2 appears in 3 clauses. If we apply Theorem 1 for variable x_2 , the resulting expression will have 6 more clauses and \bar{x}_3 will appear in 9 clauses in total. Continuing in the same manner, it is easy to see that the final purely negative CNF expression has $3^n - 2n$ clauses in total.

Notice now that no other purely negative CNF expression with fewer clauses exists. This follows from

Berge's theorem stating that if a hypergraph \mathcal{H} is simple, then $\text{tr}(\text{tr}(\mathcal{H})) = \mathcal{H}$ [2]. A straightforward result is that, if \mathcal{H} and \mathcal{G} are simple hypergraphs, then $\text{tr}(\mathcal{H}) = \text{tr}(\mathcal{G})$ if and only if $\mathcal{H} = \mathcal{G}$ (see, also, [6]). As already mentioned, a purely negative CNF expression corresponds to a hypergraph; accordingly, a purely negative CNF expression where no clause is subsumed by another one corresponds to a simple hypergraph. Since the above expression contains no subsumed clauses, no other purely negative CNF (without subsumed clauses) having the same set of maximal models exists. \square

Nevertheless, when the number of variables having at least one positive occurrence is constant or even $O(\log \log m)$ (notice that here we count the variables with at least one positive occurrence, not the positive literals in total), Theorem 1 together with the algorithm in [8] establish the following:

Corollary 3. *Let ϕ be a CNF expression with m clauses and $O(\log \log m)$ positively occurring variables. Then there is an incremental $O(m^{\text{polylog}(m)})$ algorithm for generating the maximal models of ϕ .*

4. Generating maximal models

In this section we address the problem of generating all maximal models of an arbitrary CNF expression ϕ . As already stated, this problem is satisfactorily solved for purely negative expressions. A more general case is to consider arbitrary monotone expressions (ϕ is monotone if $t_1 \geq t_2$ implies $\phi(t_1) \geq \phi(t_2)$, where t_1 and t_2 are truth assignments of ϕ). This problem can also be reduced to the dualization of a monotone expression using the method described in [12].

Naturally, all cases of SAT for which discovering one maximal model is hard, are also hard for GENERATE ALL MAXIMAL. Therefore, all hard cases in the Schaefer dichotomy are hard for our problem, too. As for the six easy ones, first observe that the problem is trivial for 1-valid and anti-Horn expressions. For the 0-valid case however, it follows from [3, Theorem 3] that checking the maximality of a model is co-NP-complete. Hence, unlike SAT, GENERATE ALL MAXIMAL is hard for 0-valid expressions. We next show that this also holds for the case of Horn expressions.

To this end, we define a suitable decision problem closely related to GENERATE ALL MAXIMAL. Let INCREMENTAL MAXIMAL be the problem of deciding whether, given an expression ϕ and a set M of maximal models of ϕ , there exists another maximal model of ϕ not contained in M . Notice that, if GENERATE ALL MAXIMAL was solvable in output-polynomial time for some class of expressions (say in time $O(c^k)$, where c is the combined size of the input and the output and $k > 0$ is a constant), then INCREMENTAL MAXIMAL would be in P for the same class of expressions: We could decide in polynomial time whether there is no other maximal model for the expression by running the output-polynomial algorithm for GENERATE ALL MAXIMAL for time at most $O(\text{size}(\phi, M)^k)$ and checking if the generated models are exactly those in M .

Theorem 4. INCREMENTAL MAXIMAL is NP-complete for Horn expressions.

Proof. Membership in NP is direct since checking the maximality of a model of a Horn expression is polynomial. To show completeness, we reduce ONE-IN-THREE-3SAT, restricted to positive literals, to our problem. This special case of ONE-IN-THREE-3SAT remains NP-complete [10] and in effect is the following problem: Given n items and a family of m sets each consisting of three items, is there a subset of items containing exactly one of the three items of a set?

Consider an instance of ONE-IN-THREE-3SAT,

$$f = \bigwedge_{i=1}^m C_i = \bigwedge_{i=1}^m (x_{i,1} \vee x_{i,2} \vee x_{i,3}).$$

To construct an instance ϕ of INCREMENTAL MAXIMAL, we introduce a variable y_i for each C_i and another extra variable z . For each C_i we construct the sets of clauses

$$\mathcal{G}_i = \{(y_i \vee \bar{x}_{i,j} \vee \bar{z}), j = 1, 2, 3\},$$

and

$$\mathcal{H}_i = \{(\bar{x}_{i,1} \vee \bar{x}_{i,2} \vee z), (\bar{x}_{i,2} \vee \bar{x}_{i,3} \vee z), (\bar{x}_{i,3} \vee \bar{x}_{i,1} \vee z)\}.$$

We also construct the set of clauses

$$C_y = \{(\bar{y}_i \vee z), i = 1, \dots, m\}$$

and the clause

$$C = (\bar{y}_1 \vee \dots \vee \bar{y}_m \vee \bar{z}).$$

The union of the above sets is a Horn expression ϕ defined on $(n + m + 1)$ variables, with at most $(7m + 1)$ clauses. To complete the construction, we define the set M of the given maximal models of ϕ as the set of vectors t_1, \dots, t_m defined on the $(n + m + 1)$ variables, where any arbitrary t_k in M assigns 1 to z , 0 to y_k , 0 to the three variables appearing in clause C_k , and 1 to the rest of the variables. It is easy to see that every model in M is a maximal model of ϕ .

Models in M are precisely the only maximal models of ϕ assigning 1 to z . This holds since by assigning 1 to z , at least one of the y_i 's has to be assigned 0. Hence, all variables of the corresponding set \mathcal{G}_i must be 0. No other variables, apart from the above ones, have to be assigned 0. Thus, ϕ has indeed m maximal models assigning 1 to z and, therefore, any other maximal model of ϕ , if it does exist, has to assign 0 to z . Suppose now that there is a maximal model t of ϕ not contained in M . Since t assigns 0 to z , every y_i has to be assigned 0, too. In order for t to satisfy every set of clauses \mathcal{H}_i , at least two literals between $x_{i,1}$, $x_{i,2}$, and $x_{i,3}$ must be 0. Since t is maximal, it has to assign 1 to at least one of them. Hence, t assigns 1 at exactly one variable in each clause of f . Thus, t specifies a one-in-three satisfying truth assignment for f .

For the converse, consider a one-in-three satisfying truth assignment of f . We construct a vector t as follows: Set z and all y_i 's to 0 and set x_j 's to the value assigned by the one-in-three assignment. It is easy to see that t satisfies ϕ . Moreover, t is incomparable with any other maximal model in M . Hence, t is a maximal model of ϕ not contained in the given set M . \square

Corollary 5. Unless $P = NP$, there is no output-polynomial algorithm that solves the GENERATE ALL MAXIMAL problem for Horn expressions.

In contrast, as the next theorem states, generating the maximal models of 2CNF expressions can be done in output-polynomial time:

Theorem 6. There is an output-polynomial time algorithm for generating all maximal models of a 2CNF expression.

Proof. Repeated applications of Theorem 1 on a 2CNF expression with n variables produce in time $O(n^3)$ a purely negative CNF expression having at most $O(n^2)$ clauses, since the resolvent of two 2CNF clauses is also 2CNF and there are at most $O(n^2)$ 2CNF clauses that can be defined on n variables. The problem is then reduced to the generation of all maximal independent sets of a graph and can be solved in output-polynomial time by the algorithm of [13]. \square

In conclusion, we pose a very intriguing open problem: Is there an output-polynomial algorithm for generating all maximal solutions of systems of equations modulo 2?

References

- [1] R. Ben-Eliyahu, R. Dechter, On computing minimal models, *Ann. of Math. Artificial Intelligence* 18 (1996) 3–27.
- [2] C. Berge, *Hypergraphs*, North-Holland Mathematical Library, Vol. 45, Elsevier Science Publishers B.V., Amsterdam, 1989.
- [3] M. Cadoli, The complexity of model checking for circumscriptive formulae, *Inform. Process. Lett.* 42 (1992) 113–118.
- [4] Z. Chen, S. Toda, The complexity of selecting maximal solutions, *Inform. and Comput.* 119 (1995) 231–239.
- [5] J. de Kleer, A.K. Mackworth, R. Reiter, Characterising diagnosis and systems, *Artificial Intelligence* 56 (1992) 197–222.
- [6] T. Eiter, G. Gottlob, Identifying the minimal transversals of a hypergraph and related problems, *SIAM J. Comput.* 24 (6) (1995) 1278–1304.
- [7] O. Ekin, P.L. Hammer, U.N. Peled, Horn functions and submodular Boolean functions, *Theoret. Comput. Sci.* 175 (2) (1997) 257–270.
- [8] M.L. Freidman, L. Khachiyan, On the complexity of dualization of monotone disjunctive normal forms, *J. Algorithms* 21 (1996) 618–628.
- [9] H. Garcia-Molina, D. Barbara, How to assign votes in a distributed system, *J. ACM* 32 (4) (1985) 841–860.
- [10] M.R. Garey, D.S. Johnson, *Computers and Intractability—A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, 1979.
- [11] D. Gunopulos, R. Khardon, H. Mannila, H. Toinonen, Data mining, hypergraph transversals, and machine learning, in: *Proc. of 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Tucson, AZ, May 12–14, 1997, pp. 209–216.
- [12] V. Gurvich, L. Khachiyan, Generating the irredundant conjunctive and disjunctive normal forms of monotone Boolean functions, Technical Report LCSR-TR-251, Department of Computer Science, Rutgers University, New Brunswick, NJ, 1995.
- [13] D.S. Johnson, M. Yannakakis, C.H. Papadimitriou, On generating all maximal independent sets, *Inform. Process. Lett.* 27 (1988) 119–123.
- [14] D. Kavvadias, C.H. Papadimitriou, M. Sideri, On Horn envelopes and hypergraph transversals, in: *Proc. 4th Annual International Symposium on Algorithms and Computation (ISAAC'93)*, Hong Kong, 1993, pp. 399–405.
- [15] D. Kavvadias, M. Sideri, The inverse satisfiability problem, *SIAM J. Comput.* 28 (1) (1999) 152–163.
- [16] D. Kavvadias, E.C. Stavropoulos, Evaluation of an algorithm for the transversal hypergraph problem, in: *Proc. 3rd Workshop on Algorithm Engineering (WAE'99)*, Lecture Notes in Comput. Sci., Springer, Berlin, 1999, pp. 72–84.
- [17] C.H. Papadimitriou, NP-completeness: A retrospective, in: *Proc. of ICALP 98*, Bologna, Italy, 1998.
- [18] T.C. Przymusiński, An algorithm to compute circumscription, *Artificial Intelligence* 38 (1989) 49–73.
- [19] T.J. Schaefer, The complexity of satisfiability problems, in: *Proc. 10th Annual ACM Symposium on Theory of Computing*, San Diego, CA, 1978, pp. 216–226.
- [20] B. Selman, H.K. Kautz, Model preference default theories, *Artificial Intelligence* 45 (1990) 287–322.