Report on the

Workshop on Satisfiability: Assessing the Progress

The Maritime Institute of Technology and Graduate Studies
692 Maritime Boulevard
Linthicum, Maryland 21090

March 3 to March 5, 2008

Organized by

Sean Weaver, U.S. Dept. of Defense (also local arrangements chair)
John Franco, University of Cincinnati
Victor Marek, University of Kentucky

Prepared for the

National Security Agency

by

Sean Weaver, John Franco, Victor Marek, and Michal Kouril

Contents

1 Introduction

Interest in solvers of propositional logic (Satisfiability) problems has been growing of late both within and outside the security community. This interest is due to significant advances in solver technology that have enabled the practical solution of some formerly difficult problems, particularly in the areas of computer security, hardware verification, equivalence checking, model checking as well as areas of Computer Engineering. In response to this interest a workshop on Satisfiability was held in early March, 2008, at the Maritime Institute in Linthicum, Maryland. The workshop adjoined the annual High Confidence Software Systems conference and many people attended sessions in both. It is estimated that 40 DoD members and affiliates from, for example, Intel, attended at least one session of the workshop in addition to about 40 researchers from the Satisfiability community.

The workshop was organized by Sean Weaver of the NSA, Victor Marek, a professor at the University of Kentucky, and John Franco, a professor at the University of Cincinnati. Speakers were chosen from a list of leaders in specific subareas of Satisfiability research. Subareas were chosen based on perceived interest to the security community. These can be grouped into four categories: theory, applications, solver technology, and extensions. Theory subareas covered the probabilistic analysis of solver algorithms, thresholds, upper bounds, and model counting, Application subareas covered include equivalence checking, bounds on van der Waerden numbers, applications to biomedical research, planning, and hardware verification. Solver technology subareas covered modeling, expressibility, knowledge representation, heuristics, non-clausal solvers, local search, autarkies, data structures, Gröbner basis algorithms, and resolution algorithms. Extensions covered include Answer Set Programming, Constraint Programming, Satisfiability Modulo Theories, pseudo-Boolean formulations, and quantified Boolean formulations. Talks ranged from broad surveys to specific results to expectations for the future. In most cases, the organizers allowed the speakers to choose their topic keeping in mind the aim of the workshop which is stated next.

The aim of the workshop was to assess the future of research in areas that are perceived to hold great potential for providing results that will build on the extraordinary success that has been seen to date in solving difficult problems by translation to Boolean forms and applying Satisfiability solvers or extensions to those forms. Assessment in this context means the application of coarse metrics to determine the following: 1) matchings of applications to developing technologies; 2) possibility of encouraging Satisfiability development based on the need to efficiently solve currently difficult problems; 3) the degree to which stronger connections between theory and solver development will ultimately benefit the solution to difficult problems; and 4) if possible, what will be the limitations of Satisfiability solvers with respect to solving particular classes of difficult problems.

In what follows the organizers wish to present their assessment of progress and directions based largely on the proceedings of the workshop, particularly the panel discussion at the end. This is done by stating and briefly defining an area, briefly mentioning past successes, and loosely measuring probable impact on application areas due to what we expect to be future results in the area. This discussion is preceded by a sketch of the history of Satisfiability research and development which we hope that people new to this area will find useful in understanding the discussion on the assessment of research progress and that all readers will find interesting.

Acknowledgments: The organizers of the workshop express their gratitude to Mr. William Martin and Mr. Timothy Thimmesh of the National Information Assurance Research Laboratory for their encouragement and financial support of the workshop.

2 A Brief History of Satisfiability R&D

The notion of Satisfiability and the means to determine it is evident in *categorical logic* which was proposed over 2000 years ago by Aristotle. Aristotle and his followers were interested in understanding how the mind works. To that end they developed an *effective process* which uses known facts about the world to deduce, or prove, new facts. They represented facts as categorical propositions and used syllogisms and a small set of inference rules to drive the derivation. Thus, to derive fact q from facts $p_1, \ldots, p_n$ they would create syllogisms $(p_1 \wedge p_2) \rightarrow r_1, (r_1 \wedge p_3) \rightarrow r_2, \ldots, (r_{n-2} \wedge p_n) \rightarrow q$, and repeatedly apply the inference rules to determine the validity of all the syllogisms and therefore q.

Categorical logic was extended and refined considerably through the medieval and renaissance periods. The goal became to realize a system powerful enough to capture all of scientific reasoning. Notable work in this direction is attributed to Leibniz who developed a language that could express either ideas or world entities. Although his language did not achieve what was intended for it, he came close to defining, with modern rigor, complete axiom systems for well-defined formal languages that also possessed decision procedures for identifying the sentences that are satisfied in every interpretation. Also notable during this period is the development of the first machine capable of solving problems in formal logic by Charles Stanhope, 3rd Earl Stanhope.

The development of Boolean algebra by Boole, Jevons, and Venn during the latter half of the 19th century was a tremendous advance that changed the course of logic and supplied some of the basic logic tools needed for the information age. Boole was one of the first to employ a symbolic language. He used this to provide a more general theory of *term logic* which has the traditional syllogistic logic as a special case. Boole was primarily interested in obtaining *inferences*. Jevons employed the *exclusive-or* operator to show how to use Boole's ideas to *reason* in categorical logic. Venn realized the emerging importance of 0-1 logic over categorical logic and refined Boolean algebra to what we know of it today. The combined work of all three rang down the curtain on syllogistic logic, which had reigned supreme for over 2000 years.

A little later Frege conceived of the project known as Logicism which intended to show that all of mathematics can be deduced strictly from the laws of logic. Although Gödel later showed this to be impossible, limited parts of mathematics remained axiomatizable, for example by first-order logic. It was in the framework of first-order logic that the term Satisfiability became rigorously defined by Tarski.

At the crack of dawn in the information age Boolean algebra was applied by Shannon to the problem of optimal relay circuit design. In his master's thesis Shannon [154] described expansions leading to DNF and CNF canonical forms of Boolean formulas and provided operations on these forms which are now known as *consensus* (DNF), *resolution* (CNF), *subsumption*, as well as others. Shannon's work was paralleled or followed by considerable refinements, for example consensus and *prime implicants* by Blake [21] and the logic circuit minimization algorithm by Quine [139] and McCluskey [118] using the notion of *essential prime implicants*.

By this time Davis and Putnam, working to develop an effective procedure for first-order logic by grounding in Boolean logic, realized that it was more efficient to solve such problems by transforming to CNF instead of DNF. Their Davis-Putnam procedure (DP) [51], based on Shannon's expansion, was almost immediately improved by Loveland and Logemann [54] to the familiar DPLL algorithm which is the basis of most present day Satisfiability solvers. Almost immediately after this Robinson [141] lifted resolution to first-order logic and Tseitin introduced *extended resolution*. Tseitin [162] also showed that *regular resolution* required superpolynomially many steps on some families of inputs, thus beginning the study of complexity lower bounds on Satisfiability algorithms.

At about the same time DP was developed, Lee [107] introduced the functional representation and data structure known as *Binary Decision Diagram* (BDD) which is a direct consequence of Shannon's expansion. About 20 years later, in the late 1970s, BDDs were publicized by Akers and Boute [8, 28]. After another 10 years Bryant [31] introduced the notion of reduced order BDD (OBDD) and its implementation which supports subproblem sharing over collections of OBDDs and the efficient manipulation of those OBDDs. At about the same time, the work of Ken McMillan [119] in developing the concept of *symbolic model checking* helped popularize OBDDs.

Until the 1960's, logicians were primarily concerned with the development of proof systems and the extent to which *true* logical statements were provable in them. After 1960, in support of formal concepts and problem-solving methods in Computer Science, Operations Research, and Artificial Intelligence, computational requirements of such systems also became important. A central question became: What are the most powerful systems that will admit only short proofs? This question turned out to be very difficult to answer, yet considerable progress was made on this beginning in 1968 with the superpolynomial lower bounds proved by Tseitin that were mentioned above. In 1971 Cook [43] showed that Boolean Satisfiability is NP-complete. In 1977 Galil [67] improved Tseitin's results to exponential lower bounds for regular resolution. Then, in rapid succession in the late 1980s, Haken [78] proved a $2^{\Omega(n)}$ lower bound for the pigeonhole clauses, Urquhart [163] improved this to an exponential lower bound, and Chvátal and Szemerédi [39] proved an exponential lower bound on random sets of clauses. A number of improvements were made by Beame, Pitassi, Karp [18] and some others. Eventually these results were generalized and unified in a striking result, due to Ben-Sasson and Wigderson [20] in the early 2000s, relating the minimum size of a resolution refutation and the maximum width of a resolvent.

The 1980s and 1990s also saw advancements in the average case and probabilistic analysis of algorithms for Satisfiability, particular variants of DPLL. In the first such analysis Goldberg [73] showed that a variant of DPLL has polynomial average time complexity. The distribution used by Goldberg was investigated by Franco and Paull [62] who showed a random assignment satisfied a random Goldberg formula with high probability. Using the now time-tested random k-SAT model they showed that the same algorithm requires exponential time on the average. Later, Chao and Franco, Chvátal and Reed, Achlioptas, and some others who are mentioned below [35, 38, 4] showed that some DPLL variants perform efficiently on random k-SAT inputs but only if their density, defined as the ratio of clauses (m) to variables (n), is $c \cdot 2^k/k$ where constant c depends on the variant.

The mathematical tools used to obtain such results were formalized by Wormald and Friedgut [166, 66] and applied to help understand the relationship between the Satisfiability threshold behavior (the rate at which the probability of Satisfiability changes with density) and the hardness of formulas. Notable contributions in this came from computer scientists Selman, Kautz, Achlioptas, Mitchell, Istrate, [117, 1, 124, 6], Dubois, Creignou, Daudé, Moore, Perez, [56, 46, 2], Kaporis, Kirousis, Krizanc, Lalas, Stamatiou, Bollobás [94, 97, 23] among others, and physicists Zecchina, Monasson, Chayes, Borgs, and Mertens [126, 23]. The contribution of the physicists was particularly important as their work provided insight which later developed into rigorous analysis.

Additional theoretical work beginning in the 1980s was directed toward finding upper bounds on the complexity of Satisfiability algorithms. The first such result for k-SAT, due to Monien and Speckenmeyer [128], was a bound of α_k^n, where, for example, $\alpha_3 = 1.618$, and $\alpha_k \rightarrow 2$ as k gets large. They invented the notion of *autark* assignments to achieve this result. This thread was continued by several researchers who eventually improved the bound to $O((2(1 - 1/(k+1)))^n)$, which is $O(1.5^n)$ for $k = 3$. In the late 1990s Schöning, Pudlak, Paturi, Zane, Dantsin, Wolpert, and Schuler [145, 135, 48, 49, 147] considered randomized algorithms resulting in a current best upper bound for unrestricted clause width of $O\left(2^{n(1-\frac{1}{ln(\frac{m}{n})+O\ \ ln(ln(m))})}\right)$. For $k = 3$ the best bound is due to Iwama and Tamaki [90] and is $O(1.324^n)$.

There is a long history of the analysis of restricted classes of CNF formulas for which efficient algorithms exist. *Horn* formulas were proposed in 1951 and by the early 1980s Dowling, Gallier, Itai, Makowski, and Lewis [55, 89, 108] had contributed to showing that unit propagation can solve Horn and *renameable Horn* formulas in linear time. In the late 1970s Aspvall, Plass, and Tarjan [12] developed a linear time algorithm for *2-SAT* which performs a depth-first search on the corresponding implication graph. In the 1990s a number of polynomially solvable relaxations of Linear Programming problems were shown to be equivalent to classes of Satisfiability. These resulted in classes *q-Horn* by Boros, Crama and Hammer [25] *extended Horn* by Chandru and Hooker [33], *CC-balanced* by Cornuéjols and Conforti [42], and *simple extended Horn* by Swaminathan and Wagner [160], among others. Schlipf, Annextein, Franco and Swaminathan [150] discovered a relatively large class called *SLUR* for Single Lookahead Unit Resolution and showed that nearly all the others were contained in this class. In the early 2000s, Kullmann [104] discovered the class of *linear autark* formulas and van Maaren [111] showed this class to contain q-Horn but not SLUR. However, in 2003 Franco and Van Gelder [63] presented evidence that the relative size of these classes is small compared to that of the set of formulas which are solvable efficiently by DPLL. In that work the notion of *deficiency* in a class-variable graph was popularized.

The classes of the previous paragraph contain mostly satisfiable formulas, but there are some polynomial-time solvable classes that contain only unsatisfiable formulas. The best known of these is the class of *minimally unsatisfiable* formulas which was introduced in 1986 by Aharoni and Linial [7]. A clause set is minimally unsatisfiable if it is unsatisfiable and removal of any clause from the set results in a satisfiable clause set. Several people independently showed this class to be solvable in time that depends on deficiency d. For example, by 2000 Kleine-Büning [99] discovered an algorithm with $n^{O(d)}$ complexity. This was improved to $O(2^d)n^4$ by Szeider [161] in 2003.

As theoretical results dominated the 1980s, practical algorithm development dominated the 1990s and 2000s. The result was a rapid improvement in SAT solver performance to the point where it made sense to solve a growing number of important problems by transforming to a CNF formula and applying a SAT solver. Perhaps the first notable advance was due to Stålmarck [156] who, in *Prover*, exploited a fixed-depth lookahead for inferences and a representation similar to that proposed by Tseitin. Freeman [65] introduced, in *POSIT*, a heuristic that tends to balance the search space. Marques-Silva used backjumping and conflict-resolution to support a directed acyclic search space in *GRASP* [114]. Malik, Zhang, and others [129] at Princeton refined conflict resolution, developed the VSIDS heuristic, watched literals, and used *restarts*, a form of depth-first lookahead, in *zChaff* to take a giant leap forward in SAT solver performance. Goldberg improved upon this and created *Berkmin* [74]. Eén and others created *Minisat* [58], an incremental solver which has proved extremely useful in some applications such as equivalence checking.

At approximately the same time this accelerated activity began on DPLL variants, Clegg and others were applying Gröbner bases (GB) to Satisfiability. The usefulness of GB algorithms is just now beginning to be understood and explored.

Another important thread in SAT algorithm research is the development of Stochastic Local Search algorithms which have shown to be extremely useful compared to DPLL variants for solving some formulas that are satisfiable. Notable contributors to that thread are Gu, Selman, Kautz, Hoos, Walsh, Gent, McAllester [76, 117, 87, 70]. Based on the fundamental idea of dynamically changing clause weights, Wah and Wu [168] developed an approach known as Discrete Lagrangian Method, whose later variants were shown to be highly effective, particularly for solving structured SAT instances.

It is the applications of SAT solvers that have driven research in the field lately. Some important applications for SAT solvers are as subsolvers or search directors for other solvers, for example *pseudo-Boolean solvers*, *Satisfiability modulo theories*, and *answer set programming.* Satisfiability has been applied successfully to problems in Artificial Intelligence, for example *planning.* Satisfiability has received great attention for successes in the field of formal verification, particularly *equivalence checking* and *model checking.* It is important to note that many of these successes have not yet been completely explored and it is likely that some will be superseded with further research.

3 Assessment of Research Progress and Directions

In Section 3.1 we cover topics that were discussed at the workshop. Of necessity, the workshop did not cover Satisfiability research in its entirety. In Section 3.2, we cover some important areas not explicitly discussed during the workshop.

Section Summary: Resolution and CNF expressions have held center stage in SAT solver design since the 1960s. Both have some limitations and progress involving resolution-based CNF SAT solvers may have peaked. The latter has the disadvantage that few practical problems are naturally expressed as CNF expressions and, although polynomial time translations can generally be found, domain-specific information that can be exploited to reduce search tends to become garbled. Many problems can be solved without translation if they are expressed as, for example, a collection of pseudo-Boolean inequalities. If unassisted by some secondary operation, theoretical and empirical results tell us that resolution has some severe limitations which force unnecessarily large search spaces on many practical problems. The workaround for these problems has been to use some auxiliary representation and operations that deal with cases that are difficult for resolution. Besides, advances in SAT solver design over the last 15 years, especially backjumping, conflict analysis and clausal learning, breadth-first and depth-first lookahead (restarts), watched literals, lemma caches, and symmetry breaking, have resulted in many orders of magnitude improvement in speed, discounting hardware speedups. But, apparently ideas related to CNF SAT solving are beginning to run out, although there remain some that have been under-explored such as adding symmetry breaking constraints and autarky recognition.

Another important SAT tool is the family of Stochastic Local Search (SLS) solvers such as Walksat and Novelty+. But these are incomplete algorithms which means they return results with certainty only when inputs are satisfiable. Moreover, DPLL-based algorithms have been refined to work well on those instances as well. Thus, the future of SLS is uncertain. Nevertheless, SLS has been successful in some cases and research in SLS will continue.

Research in SAT technology seems to be shifting to non-clausal. These mainly include Satisfiability Modulo Theories (SMT), Answer Set Programming (ASP), Pseudo-Boolean formulations (PB), and more recently, Gröbner bases have been getting a second look. This research remains driven primarily by applications.

3.1 SAT Technology Discussed During The Workshop

3.1.1 Satisfiability Modulo Theories and Solvers

The idea that SAT technology may be used as an organizing tool for solvers that are native to other applications has received a lot of attention recently under the name *Satisfiability Modulo Theories* (SMT). In this role, SAT has been used to *manage* one or more other solvers and has been used to *control* the search process in native domains. An example of the former concerns decidable open fragments of predicate logic. This area is related to various classical decidability algorithms for theories such as Presburger Arithmetic [138], theory of unassigned functions, theory of real linear arithmetic, difference logic, and theory of arrays. In fact there exists a publicly available SMT library of benchmarks with respect to background theories for which specialized decision procedures exist such as the theory of lists, arrays, linear arithmetic, and so on [158]. To some extent this direction repackages several classical results (Nelson and Oppen [130], Shankar [153], and others).

In the role of control the well-known fundamental speed up techniques that are crucial to the success of conventional SAT solvers such as conflict analysis and clause learning have not yet been generalized to the context of SMT. Although it is clear that parallelizing solver control is important, there has not yet been a significant advance in that direction.

There are several applications for which SMT should prove suitable. These include formal verification, computer aided design, compiler optimization, and scheduling, among others. A well known application is related to recent attacks on cryptographic hash functions. Researchers have shown that SAT can be used to manage the search process associated with finding collisions for hash functions such as MD5 and SHA [122]. While SAT was not the main technique used to attack the hash functions, this and previous applications appear to indicate that SAT will be applicable as a management tool for a variety of processes involving search.

Currently a large number of SMT solvers are available, in the US and Europe, from academia and industry. There are now competitions for SMT solvers along the lines of the SAT and ASP communities' competitions. Some notable solvers include *Yices* from SRI, *Z3* from Microsoft Research, *Aria* from the University of Michigan, *BarceLogic* from the University of Barcelona, Spain, *MathSat* from the University of Trento, Italy, *Sateen* from Colorado University, and *CVC* from New York University [171, 172, 115, 47].

The workshop's technical program contained two presentations devoted to the area, both of very different character. Professor R. Sebastiani of the University of Trento discussed the theoretical underpinnings of SMT and the progress in research. This research represents a *Lazy* approach to SMT and has been incorporated in the *MathSat* solver. Professor Sebastiani urged more research in the area, highlighting the need to generalize and incorporate recent SAT advances. Specifically, SMT could benefit from research into the development of theory-aware DPLL heuristics and the efficient generation of partial assignments. Sebastiani also proposed looking at non-DPLL alternatives such as mixing OBDDs with DPLL and he proposed looking at mixed lazy and eager approaches to SMT.

A different point of view was presented by Professor K. Sakallah of the University of Michigan, stressing the need to enhance SMT expressibility by incorporating techniques from Knowledge Representation. In his opinion, SMT research is driven by applications, in his case Electronic Design Automation. The paper *Trace-Driven Verification of Multi-Threaded Programs* presents a preliminary report on one specific application. To some extent the position of Professor K. Sakallah dovetails with other application papers presented during the workshop.

3.1.2 Pseudo-Boolean Formulations and Solvers

Let $\mathcal{B}^n$ be the set of binary vectors of length n. Mappings $f : \mathcal{B}^n \mapsto \Re$ are called *pseudo-Boolean* functions. Let $V = \{v_1, v_2, \ldots, v_n\}$ be a set of n binary variables. There is a natural connection between pseudo-Boolean functions and binary optimization which has been exploited since the 1960s, especially following the seminal treatment of the topic by Hammer and Rudeanu in 1968 [82]. Areas impacted by pseudo-Boolean functions are diverse and include VLSI design, maximum satisfiability, clustering in statistics, economics, graph theory, and scheduling, among many others.

All pseudo-Boolean functions may be uniquely represented as *multi-linear polynomials* of the form

$$f(v_1, v_2, \ldots, v_n) = c_f + \sum_{S \subseteq V} c_S \prod_{v \in S} v$$

where c_S is some real number that depends on S and c_f is a constant. The degree of such a polynomial is the largest S for which $c_S \neq 0$. A degree 1 polynomial is called *linear*, a degree 2 polynomial is called *quadratic* and so on.

Polynomials can also be written with non-negative constants using products of literals instead of products of variables as follows:

$$f(v_1, v_2, \ldots, v_n) = c_f + \sum_{S \subseteq V \cup \overline{V}} c_S \prod_{l \in S} l$$

where $\overline{V} = \{\neg v_1, \neg v_2, \ldots, \neg v_n\}$ is the set of negative literals associated with variables (and positive literals) V and $c_S \geq 0$ (note that c_S must be 0 if there is a complementary pair $\{v, \neg v\} \in S$). This is called the *posiform* representation of a pseudo-Boolean function. Although a posiform uniquely determines a pseudo-Boolean function, a pseudo-Boolean function may be represented by more than one posiform. Moreover, it is easy to compute a posiform from a polynomial representation but it may be difficult to compute a *unique* polynomial representation corresponding to a given posiform. Because factors c_S are non-negative, minimizing a posiform is essentially the same as maximizing the number of terms that are 0. Thus, the posiform representation is closely related to instances of maximum satisfiability and the problem of determining satisfiability of a Boolean formula may be viewed as testing whether the minimum of a posiform is equal to its constant term.

Originally, pseudo-Boolean solvers were intended to find values to maximize or minimize a given Boolean function. Early solvers were motivated by the result that any pseudo-Boolean function can be translated to a quadratic pseudo-Boolean function in polynomial time [143]. They were developed by operations researchers (for example, the basic algorithm [80, 81, 82] and DDT [26]) and have the look-and-feel of algorithms typically invented by members of that community.

Recent development of pseudo-Boolean solvers originates from within the SAT community and takes a different tack. Since any Boolean function can be expressed as a CNF expression and since a CNF clause has a posiform representation as follows:

$$l_1 \vee \ldots \vee l_k \quad \mapsto \quad l_1 + \ldots + l_k \geq 1$$

the solution to pseudo-Boolean problems may be obtained by modeling as a system of linear posiforms which are written as inequalities. This is the approach that is taken by some recent pseudo-Boolean solvers. Others solve systems of multi-linear polynomials. The advantage of both approaches is that a significant collection of mathematical tools unavailable to a strictly CNF solver, such as cutting plane operations, may be used to help solve such systems. An advantage of the latter approach over the former is that a complete translation of a given problem to the linear form is unnecessary - such a translation may garble domain-specific information and make it harder to solve.

A number of solvers (for instance *minisat* [58], *SBSAT* [64], and *aspps* [11]) provide the user with the possibility of formulating constraints either implicitly or explicitly as *pseudo-Boolean.*

It should be noted that this is an area where SAT intersects with Integer Programming (IP). Whereas IP deals with a wider area and offers a large number of techniques discovered in over 70 years of investigations of numerous researchers, SAT provides a more focused framework, with many specialized techniques. Both areas have much to offer each other. During the workshop the presentation of Dr. Boros, director of the Rutgers University Center for Operations Research (reported in Section 3.1.13), highlighted connections between the two areas.

3.1.3 Gröbner Bases

In [41] (1996) a variant of the Gröbner basis (GB) algorithm was applied to systems of posiforms (see Section 3.1.2) restricted to modulo 2 arithmetic. The base operations of that algorithm were posiform addition, modulo 2, and variable multiplication. Application of one of those operations is called a derivation. It was shown in [41] that the algorithm uses a number of derivations that is guaranteed to be within a polynomial of the minimum number possible and that the minimum number of derivations cannot be much greater than, and may sometimes be far less than, the minimum number needed by resolution. For over 10 years this theoretical result has not translated to practical algorithms and GB algorithms have not been considered competitive with DPLL and SLS. This is partly because there are many options in handling systems of polynomials, each can have drastic effects on performance, and it is not clear which will be best, a priori. But, now there seems to be developing a possibility that GB will take its rightful place in SAT [53], perhaps as integrated with conventional SAT solvers and perhaps for solving problems of specific classes.

Dr. Reeves of the Center for Communication Research, in her presentation *Uses (and abuses) of Gröbner Bases in SAT solving*, introduced the Gröbner Basis algorithm for solving Boolean functions that are represented by polynomials over the field $Z[2]$. Dr. Reeves showed that the time taken by GB variants depends heavily on the term order and data structures and discussed some optimizations to GB that have been made over the years. Finally, Dr. Reeves showed advantages that GB has which should increase its usefulness with time. Regarding Symbolic Model Checking, for example, once a basis has been calculated, a number of properties can be checked without having to recalculate the basis, and conversion from one basis to another with different variable order can be done efficiently. By preprocessing with GB an input may increase in size but actually be faster to solve. To summarize, GB will probably never replace SAT solvers but may be optimal for some applications and may assist in reducing search time for others.

3.1.4 Quantified Boolean Formulas and Solvers

The concept of quantified Boolean formulas [120] (QBF) is an extension of propositional logic that allows existential ($\exists$) and universal ($\forall$) quantifiers. The intended semantics of quantified Boolean formulas, without free variables, is that a universally quantified formula $\psi = \forall x \phi$ is true if and only if for every assignment of the truth values 1 and 0 to the variable x, ψ is true. For existentially quantified formulas

$\psi = \exists x \phi$, ψ is true if and only if there is an assignment to x for which ϕ is true. In the case of free variables, ψ is called *satisfiable* if there is a truth assignment to the free variables such that ψ is true. The satisfiability problem for quantified Boolean formulas is often denoted as QSAT.

Most of the research in QBF has been done for formulas in prenex form, that is, formulas of the form $Q_1 x_1 \ldots Q_n x_n \phi$, where $Q_i \in \{\exists, \forall\}, x_1, \ldots, x_n$ are variables, and ϕ is a propositional formula called the *matrix*. By standard techniques, every quantified Boolean formula can be transformed into a logically equivalent formula in prenex form. In the same way, by the well-known procedure for propositional formulas, logically equivalent formulas with a CNF matrix or 3-CNF matrix can be obtained. These classes are denoted as QCNF and Q3-CNF.

Quantified Boolean formulas with free variables are logically equivalent to Boolean functions. But, clearly, there is no polynomial p such that every n-ary Boolean function can be represented as a quantified Boolean formula of length $p(n)$. However, for various applications, QBFs lead to shorter formulas in comparison to propositional formulas. See, for example, [92].

For quantified Boolean formulas in prenex form, the prefix type is defined as follows:

1. The prefix type of a propositional formula is $\Sigma_0 = \Pi_0$.
2. Let Φ be a formula with prefix type Σ_n (Π_n respectively), then the formula $\forall x_1 \ldots \forall x_n \Phi$ ($\exists x_1 \ldots \exists x_n \Phi$ respectively) is of prefix type Π_{n+1} (Σ_{n+1} respectively).

It has been proved that for $k \geq 1$, the satisfiability problem for formulas with prefix type Σ_k (Π_k respectively) is Σ_k^P-complete (Π_k^P-complete respectively) [157, 167]. Since $\mathcal{P}SPACE = \mathcal{N}PSPACE$ [144], almost all quantified Boolean formula problems are solvable in $\mathcal{P}SPACE$. This is a notable difference between QBF and SAT: for example, in propositional logic, the equivalence problem is $co\mathcal{N}P$-complete, whereas for quantified Boolean formulas, the problem remains $\mathcal{P}SPACE$-complete and, in contrast to the polynomial-time solvability of the equivalence problem for Horn formulas, the equivalence problem for quantified Horn formulas is $co\mathcal{N}P$-complete [100].

During the workshop two presentations related to QBF were presented.

Professor Kleine Büning of the University of Paderborn, Germany, in his presentation *Expressive Power of Sub-Classes of Quantified Boolean Formulas*, presented a wide perspective of the complexity results for various classes of QBFs. He discussed a number of techniques, principally graph-theoretic, for getting complexity results for a variety of classes of QBFs.

Professor Remshagen of the University of West Georgia, in her presentation *Constrained Quantified Formulas*, discussed a subclass of QBF and her work on a solver for theories consisting of the formulas of that class.

3.1.5 Autarkies

An *autarky* for a clause set $\mathcal{C}$ is a partial assignment a with the property that any of the clauses $C \in \mathcal{C}$ touched by a is satisfied by a. Autarkies are a generalization of pure literals and are intended for use primarily by DPLL based solvers although adaptations to other domains exist. From the definition, if the clauses remaining after an autarky is applied are unsatisfiable, then any other assignment to the variables of the autarky will also lead to unsatisfiable residuals and need not be explored.

Autarkies were discovered by members of the SAT community [128] while searching for upper bounds on SAT solver performance. The concept was studied in great detail in papers of Kullmann [104, 105]. Autarky recognition has been employed in the preprocessing portion of a number of solvers including *OKsolver* [106], and *SB-SAT* [64]. Currently, it is felt that autarky recognition, in general, is too expensive to be useful during search although it is a trivial matter to recognize pure literals. However, there is evidence that the expense of recognizing autarkies in a limited way during search can prune the search space significantly in some cases. Professor Boros pointed out at the workshop that autarkies possess an interesting algebraic structure [104] that can be exploited to significantly improve the performance of MAX-2-SAT and $\{0, 1\}$-Integer Programming solvers (see Section 3.1.13).

3.1.6 Data Structures

Among several fundamental contributions of the past 15 years in the area of Satisfiability was the realization that various data structures (for instance *watched literals*) can significantly improve the performance of SAT solvers. Dr. Malik of Princeton University, in his presentation *SAT solvers: Efficient implementations*, gave a wide panorama of modern SAT solving techniques, stressing the contribution of data structures and heuristic functions to efficient SAT solving. The presentation presented the picture of a technology that improved rapidly and in this process contributed to applications, especially in electronic design automation.

3.1.7 Theory

More so than many other technical areas, there is a close relationship between "theoreticians" and "practitioners" in the SAT community. In the past 15 years the theoretical advances in SAT have driven improvements in SAT solvers, and significant progress in solvers has resulted in advances in the theoretical analysis of logic-based search. The workshop included a number of presentations related to the theory of Satisfiability (see also Section 3.1.8).

Professor Van Gelder of the University of California, Santa Cruz, in his presentation *Has Resolution-Based Clause Learning Hit the Wall?*, discussed the theoretical underpinnings of one of the fundamental improvements in SAT technology, namely *learning*. Each backtrack in the DPLL process results in learning one or more clauses that are consequences of the input theory. These additional learned clauses serve at least two purposes: they facilitate backjumping [16] which may result in pruning substantial portions of the search space by compacting search paths and, perhaps more importantly, they turn a tree-like search space into a DAG-like search space by preventing exploration of subspaces that had earlier been learned not to contain a solution. Thus, learning has mitigated what had been a strong advantage of OBDDs over DPLL: namely, no subfunction is expressed twice in a BDD structure.

However, learning is a *mixed blessing*. It can be expensive to manage all the clauses learned during search: there is a practical limit to the number of learned clauses that can be saved so there must be an effective way to determine the significance of a learned clause in reducing the remainder of the search space. From the point of view of theory, learning is a form of proof system and the advances in proof systems provide insight into what actually can be learned. The presentation contained the results of such proof systems and raised the question of whether further significant improvements in learning techniques are possible.

Professor Speckenmeyer of the University of Kiel, Germany, in his presentation *Upper Bounds for Satisfiability Solving*, discussed the issue of upper bounds for Satisfiability solvers. Such bounds are obtained from the theoretical analysis of Satisfiability algorithms and provide a check on existing SAT-solving systems [90].

Professor Dechter of the University of California, Irvine discussed theoretical advances on the *Constraint Satisfaction* Problem (CSP) which is a generalization of the Satisfiability problem. It is easy to represent Satisfiability as a CSP and conversely finite-domain CSPs can easily be reduced to SAT. Much of the presentation discussed the role of a well-known and widely studied parameter of CSPs called *tree-width* [140, 44]. This parameter is associated with graphs naturally occurring in CSPs and facilitates better estimates of the complexity of those CSPs. The presentation pointed to specific applications of tree-width based algorithms in studies of bioinformatics, social networks analysis, Markov decision processes, and other areas.

3.1.8 Probabilistic and Statistical Methods

Early work in Satisfiability concerned the probabilistic analysis of algorithms, deterministic and non-deterministic, complete and incomplete, with the hope of showing that one or more can be relied on to solve most problems in a reasonable amount of time. As time passed, one input distribution, called random k-SAT, became the benchmark for such results [62]. There are three parameters associated with this distribution: the fixed number of literals k possessed by every clause, the number n of variables from which clauses are constructed, and the number m of clauses in a random instance. All m clauses in a random k-SAT formula are selected independently and uniformly from the set of all possible k literal clauses taken from n variables.

It was not long before it was noticed that if $m/n > 2^k$ then a random formula is unsatisfiable with high probability and that some variants of DPLL without backtracking (even one with backtracking) find a solution with high probability if $m/n < c(2^k/k)$ where constant c depends on the algorithm [34, 35, 38]. To some, the clause density m/n provided a measure by which algorithms could be compared. But some also became curious about the speed with which random formulas become unsatisfiable as a function of density. This led to a formal proof that this transition happens very rapidly [66, 2, 3]. This rapid transition caught the eye of applied physicists as it mimicked the behavior of glassy materials as they moved through state changes. A study of this analogy led to the hypothesis that the hardness of Satisfiability problems was actually due to the nature of the transition from satisfiable to unsatisfiable as density increases. Although this turned out not to be strictly the case [6], the flurry of research that came out of the probabilistic study of Satisfiability and Satisfiability algorithms all through the 1980s, 1990s, and into the 2000s gave us a great deal of insight into the nature of hard problems and the performance of quite a number of algorithms. It also revealed instance structures that can be exploited to improve performance, revealed instance structures that cause problems for resolution [39, 20], and afforded a comparison of otherwise incomparable polynomial time solvable classes of Satisfiable [63].

The presentation of Professor J. Franco of the University of Cincinnati, *Probability in the Analysis of CNF Satisfiability Algorithms and Properties*, discussed the use of probabilistic methods in analysis of algorithms for SAT. This analysis explains the behavior of the currently used algorithms such as DPLL. The presentation discussed the applications of methods (such as the first and second moment method) both to satisfiable and unsatisfiable clause sets, classical combinatorial problems such as Hitting Set, and the possibilities of the use of such techniques for enhancing MAXSAT. With respect to probabilistic analysis, the Hitting Set algorithm is far superior to resolution on unsatisfiable formulas. Its novelty is that it transforms a combinatorial problem into one of finding the maximum eigenvalue of a matrix that corresponds to a graphical representation of a random formula. The use of eigenvalues in this way is far too under-explored at the moment and undoubtedly more surprises await.

Two probabilistic methods for solving Satisfiability problems that have received a lot of attention include Stochastic Local Search [88] (SLS) and Survey Propagation [121] (SP). Deterministic search, such as that offered by DPLL, is vulnerable to getting mired in fruitless portions of the search space and much has been added to the basic DPLL algorithm to mitigate this problem. SLS avoids the problem by admitting long jumps through the search space. These can be controlled by probabilistic means. However, the systematic tabulation of results that deterministic search provides is largely lost in SLS. Hence SLS typically cannot verify that an input is not satisfiable and SLS cannot use learning techniques as effectively as deterministic search methods. This also holds for SP. The motivation for SP comes from statistical mechanics. The idea is to treat satisfiability solving as an analogue to state changes in glassy material. Thus, the number of solutions a formula has corresponds to an energy level. On random k-SAT formulas percolation effects arise which are analogous to those found in materials near state-change thresholds. SP is designed to exploit this fact and mimic material behavior in the vicinity of the threshold by seeking analogously lower energy states. Although SP has been shown to be extremely effective on random k-SAT formulas, the reasons have not been clearly enunciated until now.

Professor Achiloptas of the University of California, Santa Cruz, in his presentation *The Physics of Random Formulas*, introduced the audience to the techniques of *Survey Propagation* and explained through rigorous probabilistic analysis why SP works well on random k-SAT formulas and probably has also explained why a number of incomplete algorithms that were mentioned above work well when $m/n < O(2^k/k)$. He showed that for low clause density there exists a single local energy minimum. Then as density is increased, there is a sudden change to several local minima. Finally, there is another jump where exponentially many local minima materialize. Thus, it is not surprising that so many algorithms perform miserably on unsatisfiable formulas with density just above the transition point. Presumably, as density is increased further, these local minima coalesce (since the minima cannot go below 0) and verifying unsatisfiability becomes easy again. In fact, theoretical results corroborate this [18, 72].

Another presentation devoted to the same topic was delivered by Mr. E. Hsu of the University of Toronto, *Statistical Message-Passing Techniques for SAT*. This presentation introduced a number of techniques in this area, including the details of Belief Propagation [136] and Survey Propagation, and their generalization called *Expectation Maximization*. All these techniques use statistical properties of the bipartite graph that naturally relates clauses with variables. The talk presented a perspective of propagation methods and their likely applications.

3.1.9 SAT and ACL2

Professor Hunt of the University of Texas, Austin, in his presentation *Integrating SAT into ACL2*, discussed the use of SAT as one of the libraries for the first-order theorem prover ACL2 (the Boyer-Moore theorem prover). Here SAT supports some tasks in ACL2 and is used to achieve faster proofs. This work relates both to SMT (see Section 3.1.1) and to first-order theorem proving, and shows a useful but non-conventional application of SAT.

3.1.10 SAT and FPGAs

Dr. Zhang of Microsoft Research, Silicon Valley, delivered a talk on *Designing a Practical Hardware Accelerator for SAT using FPGAs.* He reported on an experimental hybrid design which consists of a collection of FPGAs and a CPU. The FPGAs are responsible for Binary Constraint Propagation (that is, discovering inferences and adjusting the current formula accordingly) and perform several orders of magnitude faster[1] than traditional sequential SAT solvers. This and additional improvements in data structures as well as in handling the parallelism inherent in FPGAs resulted in an experimental system that appears to significantly improve performance of the SAT solver. FPGAs have been used previously to speed up SAT solving, for example see [159] and [174].

DPLL SAT solvers, apart from backtracking, do a lot of independent checking for contradictions and inferences. If these checks could be handled independently, then BCP could be executed in one step. FPGAs can provide a tool for achieving this but only for small or highly specialized problems since it is generally not the case that these checks are independent. Building a general SAT solver on reconfigurable hardware (say, on FPGAs) is yet another challenging problem that has been under-explored.

3.1.11 Knowledge Representation

Up to now, some expert knowledge of Satisfiability has been necessary to get the most out of any SAT solver. But, as SAT technology matures and SAT solvers are used in a wider variety of applications such as hardware and software verification of industrial projects, it will be necessary to ensure that a user without expert knowledge of Satisfiability can still achieve optimal or near optimal results with a given SAT solver. Thus, the issue of programmer environments and, in particular, interfaces to other programming languages will become urgent and requires attention now. Programmers need to be able to *model* problems as SAT instances using higher-level languages.

Dr. Truszczynski of the University of Kentucky, in his presentation *Knowledge Representation Languages – a programmer's interface to Satisfiability*, reported on his work on the language PS+ of propositional schemata [11]. This language in intended to serve as such a modeling language. It allows for the representation of problems in the class $\mathcal{N}PMV$ and search problems in the class $\mathcal{N}P$ [68]. Its implementation is supported by a program called a *grounder* that translates the expressions of the language to a format suitable for a back-end SAT solver. Dr. Truszczynski also reported on the related research of Denecker, Mitchell and Ternovska on ID-logic, a formalism that extends the logic PS+ [52].

Professor Mitchell of Simon Frazer University, Canada, in his presentation *Building and Using Model-Expansion Based SAT Front End*, discussed the use of SAT as a back-end for a declarative system that allows the programmer the use of object variables, but no function symbols, and inductive definitions. His language, called $\mathcal{L} - MX$ [125], allows the programmer to formulate a problem and then compute its solutions. This technology, closely related to the one presented by Dr. Truszczyn-

[1]This can be thought of as a fine grain parallelism. Ultimately, distributed computing can be combined with this fine grain parallelism.

ski, allows for formulating and solving complex constraints in first-order language with restrictions as stated above. Applications in bioinformatics and combinatorial optimization were discussed.

The issue of the relationship of Satisfiability solvers with imperative formalisms was discussed in the presentation of Professor Le Berre of the University of Artois, France. In his presentation *Some applications of SAT, What SAT4J users do*, Professor Le Berre presented the use of a SAT library which is available to JAVA programmers. The presentation discussed both the interface provided by the system SAT4J (a Satisfiability solver embedded into JAVA) as well as specific applications of SAT4J. Those applications come from bioinformatics and software engineering. SAT4J serves as a knowledge representation tool, enabling JAVA programmers to have a seamless connection to SAT technology.

3.1.12 Answer Set Programming

Dr. Marek of the University of Kentucky reported progress in the area of *Answer Set Programming* (ASP) [113, 15], a formalism that is closely related (but not identical) to SAT. ASP addresses both modeling (semantic) and solver (syntactic) issues while SAT is concerned primarily with speedups that come from exploiting syntactic properties. ASP is based on an extension of Horn logic and has its roots in logic-based Artificial Intelligence. The theory on which ASP is based, known as *non-monotonic logic*, originated with McCarthy and Reiter, and was further developed by Gelfond, Lifschitz. Progress in ASP *solvers* has been closely related to (and in the recent years driven by) the search technology developed in the SAT community. However, probably due to its origin, ASP always pays more attention to the issue of programming, i.e. the ease of *representation* of constraints and other *knowledge representation* issues. ASP, in its various versions, solves (in principle) the same class of problems as SAT, although more powerful extensions were implemented as well. The issue of SAT solvers and ASP solvers working in *portfolio* arrangements was discussed.

3.1.13 MAX-SAT

Since many practical problems are optimization problems, it is often more natural to model these as instances of Maximum Satisfiability or one of its variants. Examples include finding most probable explanations in graphical networks such as Bayesian belief networks from bioinformatics and computer aided design, finding minimal conflicts in a contradictory database, debugging the design of a VLSI circuit, and finding optimal scheduling solutions with “hard” and “soft” constraints. There are at least four important main varieties of Maximum Satisfiability problems. These include:

1. **MAX-SAT**: Find a truth assignment that maximizes the number of clauses satisfied.

2. **Weighted MAX-SAT**: Assign a positive weight to each clause and find a truth assignment that maximizes the total weight of all satisfied clauses.

3. **Partial MAX-SAT**: Let some clauses be labeled "hard" and some "soft" and find a truth assignment that satisfies all "hard" clauses and a maximum number of "soft" clauses.

4. **Weighted Partial MAX-SAT**: Let some clauses be labeled "hard" and some "soft," assign positive weights to the "soft" clauses, and find a truth assignment that satisfies all "hard" clauses and maximizes the total weight of all satisfied "soft" clauses.

Each of these has subcases, for example in MAX-2-SAT the limiting the number of literals in a clause to 2. MAX-2-SAT is important because a number of other $\mathcal{N}P$-complete problems, for example graph problems such as Maximum Cut and Independent Set [36, 112] can be reduced to it in a natural way.

Significant, standard SAT techniques such as constraint propagation and conflict analysis (i.e. learning) do not directly apply to MAX-SAT problems. Many of the proposed methods for MAX-SAT are based on approximation algorithms [48]. Some use branch-and-bound methods [79, 24, 14, 86, 132, 71] and some rely on translations to SAT [170, 10].

Worst-case upper bounds have been obtained with respect to three parameters: the length L of the input formula (*i.e.*, the number of literals in the input), the number m of input clauses, and the number n of input variables. The best known bounds for MAX-SAT are $O(L2^{m/2.36})$ and $O(L2^{L/6.89})$ [14]. The question of whether there exist exact algorithms with complexity bounds down to $O(L2^n)$ has been open and of great interest (see [131, 9, 75]) since an algorithm which enumerates all the 2^n assignments and then counts the number of satisfied clauses in each assignment would take time $O(L2^n)$. Recently, it has been shown that a branch-and-bound algorithm can achieve $O(b2^n)$ complexity where b is the maximum number of occurrences of any variable in the input. Typically, $b \simeq L/n$.

For MAX-2-SAT, the best general bound is $O(m2^{m/5})$ [75] which was obtained in 2003. A recent branch-and-bound algorithm [173] has a bound of $O(n2^n)$ which becomes $O(\sqrt{m}1.414^{\sqrt{m}})$ for the case $m = 4n^2$. In this case the bound of the branch-and-bound algorithm is substantially better than the general bound.

Two presentations during the workshop were related to MAXSAT.

The presentation by Dr. Marques-Silva of the University of Southampton *Maximum Satisfiability: New Algorithms and Applications*, introduced research on applications of variations of SAT technology to MAX-SAT. Specific techniques discussed were those of *unsatisfiable cores* which are parts of the input CNFs that are minimally unsatisfiable. The implementation of the MAX-SAT algorithm was introduced in the context of pseudo-Boolean constraints and called *msu* 1.0.

A much different presentation was delivered by Dr. Boros of the Rutgers University Center for Operations Research and entitled *A Strongly Polynomial Preprocessing of MAX2SAT Instances.* This work looks at the problem of preprocessing collections of 2-clauses in MAX-SAT. An interesting aspect is the use of techniques that are clearly influenced by SAT research, such as autarkies, in the context of Operation Research. In fact, various facts presented in the presentation have both SAT and OR versions. It is well-known that there is a duality between various techniques developed in OR and in SAT, and the presentation of Dr. Boros is an example of the usefulness of different perspectives offered by these two areas on the same class of problems.

3.1.14 Applications

Professor Truemper and Ms. Moreland of the University of Texas at Dallas, in their presentation *EXARP and SUBEX Algorithms for Explaining Data Sets*, discussed the issue of finding explanations for observed data. A SAT engine is used for back-end processing for the explanation system. The reported research has applications in, among others, bioinformatics and finance.

Professor Kautz of the University of Rochester contributed a presentation *Deconstructing Planning as Satisfiability; A Revised Report.* Use of Satisfiability as a planning engine was pioneered by Kautz and Selman in 1992 and is an example of a significant application of Satisfiability in Artificial Intelligence. The presentation describes the progress both in planning and as an application of Satisfiability in Planning.

Dr. Heule of Delft University of Technology, Holland, in his presentation *Improving the odds, New lower bounds for van der Waerden numbers*, presented results on the use of SAT in one area of Combinatorics. It is known that combinatorial problems, especially in Ramsey Theory, provide a large lode of hard SAT problems. Conversely, SAT contributed significantly to establishing interesting special results in Combinatorics. In this talk, bounds on van der Waerden numbers were obtained by observing *solution patterns* rather than the traditional *syntactic patterns.*

Professor Lynce of the Technical University of Lisbon, Portugal, in her presentation *Haplotype Inference with Boolean Satisfiability* presented an application of SAT techniques to a practical problem (haplotype inference by pure parsimony) stemming out of Bioinformatics. The findings of the paper were that there is a natural representation of that problem as an extension of SAT by Pseudo-Boolean constraints.

Dr. Velev of ARIES Design Automation, presented *Automatic Formal Verification of Pipelined Microprocessors and DSPs.* Specific findings of this presentation were that an efficient translation of various problems related to model checking into SAT leads to applications of SAT improving various procedures of Electronic Design Automation by (possibly) several orders of magnitude.

Dr. Eén of Cadence Research Laboratories, in his presentation *Cut Sweeping, a* *scalable alternative to SAT sweeping*, discussed a new technique used in *equivalence checking*, an important application of technologies such as SAT to testing correctness of hardware design. The presented technique can serve both as a way to minimize circuit representations and as a preprocessing method before equivalence checking is used.

3.1.15 A Vision

Professor Kullmann of the University of Swansea discussed a number of challenges facing the SAT community. Entitled *New theories for SAT*, the presentation of Professor Kullmann contained a number of proposed directions grouped in "challenges" that deal with possible improvements to both SAT algorithms and SAT applicability.

Professor Kullmann urged the following:

1. A wider application of techniques of continuous mathematics, especially the theory of heuristic functions and techniques from mathematical economics.
2. Investigation of connections with combinatorics and computational algebra, especially computational algebraic geometry.
3. Classification of NP-search problems from the point of view of applications of SAT techniques as a basis for solving them.
4. Investigation of relationships between SAT and Constraint Satisfaction.
5. Investigation of techniques related to graph and hypergraph width to transform the theoretical advances due to Courcelle, Gottlob and others to practical algorithms. This issue was raised, independently, in the presentation of Professor Dechter.

On more practical issues, Kullmann suggested creation of common standards for software development in SAT and offers to the user community a SAT-library and framework that he is currently building.

3.2 Topics Not Covered in the Workshop

Of necessity, the workshop *presentations* could cover only a fraction of the areas covered by Satisfiability research. In contrast, *discussions* during the workshop involved additional material. As a result, this section stresses the topics neither directly covered in the presentations nor extending the various points of view made in the presentations.

3.2.1 Processing Industrial Benchmarks

Most of the current interest in Satisfiability formulations and methods is due to refinements to the basic DPLL framework that have resulted in speed-ups of many orders of magnitude and have turned many problems that were considered intractable in the 1980s into trivially solved problems now. These refinements include improvements to variable choice heuristics, early pruning of the search space, replacement of a tree-like search space with a DAG-like search space, and breadth-first and depth-first lookahead for inferences.

DPLL was originally designed with two important choice heuristics: the pure-literal rule and the unit-clause rule. But, variable choices in case no unit clauses or pure literals exist were arbitrary. Thus, extensions of both heuristics were proposed and analyzed in the 1980s through the early 2000s. For example, the unit clause rule was generalized to the shortest clause rule: choose a variable from an existing clause containing the fewest unset literals [35, 38] and the pure literal rule was extended to linear autarkies [111]. Other notable heuristics include the majority rule: choose a variable with the maximum difference between the number of its positive and negative literals [34], probe-order backtracking [137], and a greedy heuristic [77, 94]. The above heuristics represent parts of many heuristics that have actually been implemented in SAT solvers and were studied primarily because their performance could be analyzed probabilistically.

Early heuristics designed to empirically speed up SAT solvers were based on the idea that eliminating small clauses first tends to reveal inferences sooner. Possibly the earliest well-known heuristic built on this approach is due to Jeroslow and Wang [91]: they choose a value for a variable that comes close to maximizing the chance of satisfying the remaining clauses, assuming the remaining clauses are statistically independent in some sense that is too complicated to describe here. They assign weights $w(\mathcal{S}_{i,j})$ for each variable v_j and each value $i \in \{0,1\}$ where, for a subset of clauses $\mathcal{S}$, $\mathcal{S}_{i,j}$ is the clauses of $\mathcal{S}$ less those clauses satisfied and those literals falsified by assigning value i to v_j and $w(\mathcal{S}_{i,j}) = \sum_{C \in \mathcal{S}_{i,j}} 2^{-|C|}$ ($|C|$ is the width of clause C). The Jeroslow-Wang heuristic was intended to work better on satisfiable formulas but Böhm [22] and Freeman [65] came up with heuristics that work better on unsatisfiable formulas: they choose to eliminate a variable that in some sense maximally causes an estimate of both sides of the resulting search space to be roughly of equal size. This can be done, for example, by assigning variable weights that are the product of positive and negative literal weights as above and choosing the variable of maximum weight. Later, it was observed that the activity of variable assignment was an important factor in search space size. This led to the VDLIS heuristic of GRASP [114] and eventually to the VSIDS heuristic of Chaff [129]. Some variant of at least one of the latter two exist in most current DPLL based SAT solvers.

Another important refinement to SAT solvers is the early generation of inferences during search. This has largely taken the appearance of lookahead techniques. It was Stålmarck [156] who demonstrated the effectiveness of lookahead in Prover: the particular type of lookahead used there is best described as breadth-first. This scheme is limited by the fact that search space width can be quite large so the lookahead must be restricted to a small number of levels, usually 2. Nevertheless, Prover was regarded to be a major advance around 1992. Later, Chaff used depth-first lookahead (this form is generally called restarts) to greater effectiveness [129]. Depth-first lookahead also solves a problem that comes up when saving non-unit inferences (see below): such inferences may be saved in a cache which may possibly overflow at some point. In the case of depth-first lookahead, the inferences may be applied and search restarted with a cleared cache.

Extremely important to SAT solver efficiency are mechanisms that reduce the size of the search space: that is, discover early that a branch of the search space does not have to be explored. A *conflict analysis* at a falsified node of the search space will reveal the subset V_s of variables involved in that node becoming false. Thus, the search path from root to that falsified node can be collapsed to include only the variables in V_s, effectively pruning many branch points from the search space. This has been called non-chronological backtracking or backjumping [114] and is commonly seen in all SAT solvers as well as Constraint Satisfaction solvers. The result of conflict analysis can also be saved as a multi-variable inference in the form of a clause. If later in the search the values of a subset of variables are set to match a saved inference, backtracking can immediately be applied to avoid repeating a search that previously ended with no solution. This idea, known as *clausal learning*, is a part of all modern SAT solvers: its power comes from turning what would be a tree-like search space into a DAG-like search space. This has gone a long way toward mitigating the early advantage of BDDs, which are non-repetitive structures with respect to subfunction storage, over search. However, the learning must be managed carefully to prevent too many unimportant inferences from overflowing the inference cache. This is still not a completely solved issue.

Finally, the development of clever data structures have been crucial to SAT solver success. The most important of these is the structure that supports the notion of the watched literal.

3.2.2 SAT Competitions

The SAT community conducts periodic *SAT competitions*. During these competitions SAT solvers solve a number of problems in a particular category (either *random problems*, *industrial problems*, or *hand-made problems*). A complex scoring process is used, but a simple view of it is this: a solver runs for a set small amount of time, ex. 10 minutes, and the number of problems solved is then counted, with systems that solve more problems winning.

One downside to SAT competitions is that they do not test if a solver can solve extremely difficult yet modestly large problems or extremely large scale problems. The spectacular result of Dr. Kouril computing the van der Waerden number $W(2, 6)$, which required over a year's computation by a specialized solver on a problem with a relatively small number of variables, does not relate to the SAT competition. On some very large problems some solvers that use clause learning do not safely manage learned clauses and crash once they run out of available memory: a behavior that is not testable in 10 minutes.

Nevertheless, competitions play a positive role in SAT research, and indicate which ideas may lead to improved performance if implemented. Some prominent SAT solvers are *kcnfs*, *March*, *MiniSat*, *OKSolver*, *RSat*, *SATzilla*, *UBCSAT*, and *zchaff*. Some others are mentioned throughout this report.

3.2.3 Reconfigurable Hardware

The availability of reconfigurable hardware, such as FPGAs, creates new possibilities for the implementation of SAT solvers, and more generally other solvers, in a variety of formalisms that are well-suited to such devices. The feasibility of this approach, potentially speeding up SAT solving by orders of magnitude, has been confirmed by experiments of Dr. Kouril when finding $W(2, 6)$ as mentioned in Section 3.2.2, Dr. Zhang's FPGA SAT solver as mentioned in Section 3.1.10, and other reports. These results show that hardware can speed up SAT computation, while getting around the brick wall of unchanging processor clock rates, and is suitable for large problems that go beyond the current capabilities of general purpose computers.

3.2.4 Multicore and Distributed Computing

With the increasing number of cores per socket in the state of the art processor (both Intel and AMD have 16 core CPUs on the road map) and stagnation of the single CPU core speeds, it is unavoidable that the parallelization of SAT solvers will become more and more important. Several attempts have already been made to design an efficient parallel SAT solver [61, 133].

3.2.5 SAT and Computer Architecture

The architecture of modern computers and, in particular, the use of data and program caches and graphics accelerators, suggests that these features may be exploited to yield significant improvements in processing time. For this reason, the architects of SAT solvers pay attention to issues such as fitting prominent procedures and data in $L2$-cache. Likewise, the organization of the storage of input problems into memory pages may affect the processing time. Experiments show that careful programming that takes into account moving the data and programs from RAM to the $L2$ cache can significantly affect the performance of SAT solvers. Exploitation of AGP hardware is still waiting to be looked at seriously.

During the workshop the point has been raised that SAT solvers might benefit from specialized instructions built into general purpose processors. There are already signs of effort on this front in the form of various bit counting instructions (occasionally on the web these are called NSA instructions). Just like multimedia processing taking great advantage of vector processing instructions (such as Intel SSE), we believe that SAT solvers can be sped up by taking better advantage of the underlying hardware. A combination of FPGA and general purpose processors or dynamic modification of the microcode in CPUs might lead to significant increases in the processing power of SAT solvers.

3.2.6 Equivalence Checking

Combinatorial Equivalence Checking (CEC), pertinent to areas such as formal verification, verifying compilers, version control, and circuit minimization, is used to prove or disprove the functional equivalence of two combinatorial circuits: that is, prove whether or not two circuits produce the same output for every possible input. CEC is miter-based, meaning that the equivalence check can be done by solving a Satisfiability instance that represents the miter of two circuits [30].

One straightforward way to check the equivalence of two circuits is to solve a CNF representation of their miter formula. Using this method, the two circuits are equivalent if the corresponding CNF is unsatisfiable. In general, SAT solvers that primarily exploit input structure cannot solve large miter formulas in a reasonable amount of time.

The latest equivalence checking techniques involve developing ways to partition miter formulas into many smaller, easier to solve sub-formulas. Using these techniques, incremental SAT solvers such as Minisat are able check the equivalence of circuits with millions of internal nodes [109]. The structure underlying these techniques is known as And/Inverter Graphs (AIGs) [102, 103] and this is used with random simulation [101] to detect pairs of internal circuit nodes with a high probability of being equivalent. A technique called SAT sweeping [103] is then used to prove the equivalence of the two circuits by solving smaller SAT instances that represent the proofs of the candidate equivalent nodes.

Unlike the traditional approach of using BDDs to represent circuits, AIGs are allowed to contain segments that are functionally redundant. By merging equivalent nodes (proven through SAT sweeping and structural hashing), some redundancy is removed and the AIG is reduced to a FRAIG (Functionally Reduced AIG) [123]. FRAIG technology, or more specifically circuit minimization, is relevant to areas such as hardware accelerated emulation and FPGA synthesis [59]. One recent FRAIG approximation algorithm, called Cut Sweeping, is 10x-100x faster than SAT Sweeping while maintaining 50%-90% of the reductions [59].

3.2.7 Model Checking

A variety of tasks related to testing properties of various classes of automata and, more generally, state machines can be expressed in temporal modal logic and in particular logics such as LTL and CTL. The Satisfiability of formulas of, say, LTL over a finite Kripke structure can be efficiently reduced to propositional Satisfiability. In this context, the use of Satisfiability has been in a form known as Bounded Model Checking. Since properties such as *reachability*, *liveness*, *exclusion*, among others, for machines are fundamental for applications in Computer Engineering, Bounded Model Checking and Model Checking more generally, is one of the prime areas of Satisfiability with significant industrial applications. Satisfiability is not the only back-end technology for Model Checking; other techniques, such as those involving BDDs, can be used for these purposes as well. Experiments show that Bounded Model Checking systems using Satisfiability are effective in handling large industrial benchmarks.

3.2.8 Lookahead

Throughout this report "lookahead" has been mentioned numerous times. This idea is now so ingrained in SAT solver design that it is taken for granted and there is no particular expert on the subject. Lookahead techniques can enhance both the heuristic and the deduction capabilities of a SAT solver, speeding up search by allowing for smarter decisions and earlier conflicts. Because lookahead techniques can be used to help create a balanced DPLL search-tree, they tend to help solve both random problems and problems with "intrinsic combinatorial hardness" [84].

Lookahead techniques are the key component in solving many classes of SAT problems. Many recent advances in look-ahead technology, such as *equivalence reasoning*, *local learning*, *adaptive heuristics*, and *double look-ahead* ([110, 85, 106]) have been used to create competitive SAT solvers such as `kcnfs`, `march`, `OKsolver` and `satz`. However, lookahead based SAT solvers can suffer from the computational costs of computing the lookahead. Reducing these costs and discovering new techniques is essential for making lookahead based SAT solvers competitive on a wider range of SAT problems [83].

3.2.9 Preprocessing

Syntactic transformations of input theories before search often result in an improvement in processing time. A variety of techniques for preprocessing have been reported in the literature and various SAT solvers apply preprocessors. Transformation techniques include partial closure under resolution or its variations such as hyperbinary resolution, closure under resolution with 2-clauses, translation of pseudo-Boolean constraints into SAT, translating sets of clauses back into Boolean functions (especially linear polynomials, but also other Boolean functions), addition of symmetry breaking constraints (Section 3.2.10), discovery and application of inferences and autarkies, and other techniques.

3.2.10 Symmetry Breaking

SAT problems may exhibit symmetries, e.g. they may be invariant under some permutation of literals. When such a phenomenon happens, preprocessing can result in *symmetry breaking*. This amounts to the simplification of the problem since in the process of symmetry breaking some variables may get an assigned values. It should be clear that symmetries may only appear in a subproblem resulting from a partial assignment made during search. The SAT community has recently devoted significant attention to symmetry breaking.

4 Panel Discussion

The panel discussion involved both theoretical and practical aspects of SAT. The fundamental questions that face the community concerns the development and use of new ideas that can speed up SAT solvers. Most future breakthroughs in SAT solver speedup will probably come from non-clausal solvers using primitive operations other than resolution, new and developing ways to represent knowledge, efficient encodings to base-level Boolean forms, preprocessing of other kinds, parallelization, and reconfigurable hardware. Past breakthroughs have usually been influenced by some theoretical results or at least some form of analysis. It is believed this will continue to be the case.

4.1 User Community Issues

One of the concerns presented during the discussion was the interface between Satisfiability research and solvers and the potential user community. The participants claimed that SAT technology is powerful but outsiders cannot yet understand *when* it will be useful. The potential user needs a better sense of this. Steps should be taken to popularize the use of SAT solvers in domains such as bioinformatics, computational mathematics and other domains where the search of large finite spaces is performed. In particular, better "packaging" of SAT technology is needed. Front ends (such as SAT4J) are needed to help the user community. Progress in Knowledge Representation and SAT will definitely help to alleviate this situation.

A concern that was not mentioned but has a place in this document is a consequence of the conduct of the annual SAT competitions. These competitions have been helpful in introducing people to SAT technology and have made available ready-made solvers for people to try out. However, it is often the case that the solvers submitted to the competitions have been designed to do well in competition and lack the robustness a user would need. Section 3.2.2 has more details on this problem. Moreover, competition versions do not come with a warning to that effect. The result is that a potential user may give up on SAT technology prematurely if he or she chooses a stripped-down, competition-version SAT solver to test and the results are less than satisfactory. Also bad is that such an unsatisfactory result may strengthen the opinion of a reluctant tester that SAT technology will not be useful in solving the problems tested. This is especially likely now since it is a given that SAT expertise is generally needed to use a SAT solver effectively.

4.2 Generalizing Ideas

There is a need to generalize the algorithmic ideas that have been successful for Satisfiability to other domains such as Constraint Satisfaction as well as applications such as scheduling and reasoning. This includes various forms of lookahead, conflict analysis and learning, dynamic reordering of the search space, search heuristics, heuristics for dynamically restructuring the database of learned facts, and preprocessing. This is part of the broader question of whether it is better to solve problems in their native domain with specialized or even general search solvers or to transform to a domain that has been heavily researched and is able to bring to the table many solving tools that are not currently available in the original problem domain. The answer most likely depends on the problem that is being solved. Allen Van Gelder commented that experience shows graph coloring problems benefit from translations to CNF Satisfiability but scheduling problems do not. This leads to the question of which problem types can benefit hugely from translations to CNF Satisfiability. It has been claimed that Satisfiability solvers seem to do well when reasoning in combinatorial space.

5 Recommendations

Interest in Satisfiability is expanding for a variety of reasons, not in the least because nowadays more problems are being solved faster by SAT solvers than other means. This is probably because Satisfiability stands at the crossroads of Logic, Graph Theory, Computer Science, Computer Engineering, and Operations Research. Thus, many problems originating in one of these fields typically have multiple translations to Satisfiability and there exist many mathematical tools available to the SAT solver to assist in solving at least one of those with improved performance. In other words, the success of SAT technology is due to *improvements in the technology itself* and also due to *improvements in the use of those technologies.*

An excellent example of this is the problem of checking whether two combinational circuits are functionally equivalent. This problem is at the core of many applications such as verifying that a vendor's circuit precisely meets specifications and does nothing more. Just a few years ago this problem was considered intractable, now it is considered to be a well-solved problem due to new algorithms that have been developed to work with new SAT solvers. SAT technologies that are indispensable to solving equivalence checking problems are conflict analysis, clause learning, and the development of an efficient incremental solver. Structures and methods that were invented to best use the new SAT technologies include And-Inverter Graphs and Cut Sweeping. Remarks from the audience seem to confirm that in a number of cases trying some SAT technology, for example some pseudo-Boolean solver, based on some educated guesses, has been successful at solving other important problems.

But SAT solvers have failed to provide good solutions to many problems as well. Several reasons for this can be identified. These are:

1. Current "best" SAT solvers are resolution-based. Resolution has been the workhorse operation of SAT solvers since 1960. It has a number of wonderful properties that have captured the interest of considerably many theoreticians and practitioners and it is simple enough to support a remarkably detailed analysis of how it works. But, as analysis shows, resolution has some severe weaknesses and we know that it cannot solve all problems without some assistance from other technologies. Moreover, new, effective ideas for using resolution are getting harder to come by. This has caused people to look at new SAT methods that are less dependent on resolution. But, it is only relatively recently that R&D has sprouted a few such branches. It is along those and other such branches, as yet non-existent, that most future successes will undoubtedly appear.

2. The structure of some problems is such that inferences cannot be derived effectively until much of the search space is explored. This negates the power of conflict analysis and clause learning and seriously hurts the chance that a resolution-based solver will perform well. The only effective way to deal with this is to add constraints as though they had been inferred from the given input. There have been several attempts to do this and some have been quite effective. The most notable successes concern symmetry breaking constraints: for example, add constraints that force only one of a family of solutions to hold, if any hold. The major problem here is who decides what the symmetry breaking constraints are. Some work has been directed at automating this process, but as of yet no such system is guaranteed to find all symmetry breaking constraints or even the most effective ones.

3. Most users do not have the time or inclination to become experts in SAT technology. This should be obvious. But solving this problem is difficult. From empirical results it seems that no one SAT technology will work best out-of-the-box for all problems. Rather, careful manipulation of SAT solver parameters and careful translation to an input form are currently required for most SAT

solver successes. This seems unlikely to change in the near future. One possibility that has not been completely explored, though, is the development of a high-level logical language in which a user would be able to express succinctly the problem that needs to be solved and an automated translator of sentences in that language down to an input form that a SAT solver can handle. Galois, a Portland based high assurance company, already has something along these lines in its offering called Cryptol, which is still under development. More on this below.

4. Theory needs to catch up with practice. Theory can benefit SAT solver technology in at least two ways: it can provide intuition about the reason something does or does not work and it can facilitate direct comparisons between features that are not dependent on an available database of benchmarks. The intuition usually leads to improvements in solver design. For example, we can trace the theoretical study of heuristics alongside their implementation in experimental solvers. But, the direct comparisons have become more important. For quite a while practitioners have been saying their new features result in an order of magnitude improvement over conventional solvers on some benchmarks and that has been fine because orders of magnitude can make the difference in the solvability of a class of problems. But orders of magnitude improvements may not be adequate in the future. Eventually it will be the order of complexity that will matter, both in the worst case and probabilistically. Space complexity may also become important as space becomes traded for speed. Lower bounds on the size of a search space for a given method is already important and will become more so. Many such results have been obtained for resolution and BDDs but not for the new technologies that are looming on the horizon. Theoreticians may also seek conditions under which problems are guaranteed to be solved efficiently. For example, the theory of minimally unsatisfiable subsets has inspired some additions to SAT solvers and work on Horn formulas, 2-SAT formulas and other polynomially solvable sets has inspired the notion of backdoor sets.

5. Reconfigurable hardware has not been fully exploited. A result in the 1980s astonished the SAT community of that era: simply partitioning a CNF instance ϕ of Satisfiability among 2^k parallel solvers by choosing k variables and handing each solver ϕ with those k variables uniquely set as one of 2^k possible assignments resulted in a speedup that was significantly greater than 2^k. It wasn't long before the answer to this riddle was solved: the parallelization of a standard DPLL algorithm resulted in a new sequential algorithm that probably would not have been discovered otherwise. This is one of the things that reconfigurable hardware may be able to do. More obviously, as feature size and processor and memory speeds hit a brick wall, parallelization, either tightly coupled (as in Beowulf clusters) or loosely coupled (as in distributed computing) is about the only way to continue speedups outside of algorithmic improvements. Orders of magnitude speedups over software have been reported running a van der Waerden number SAT solver on a few commercially available FPGAs at just 400MHz

each. In this application each FPGA had the power of approximately 40 SAT solvers. A possible problem with FPGAs is that they must be programmed using an archaic language called VHDL. However, this problem is being solved by new tools that automatically generate VHDL code from a given specification. One example is Cryptol by Galois which in many cases produces verified VHDL code! Cryptol is still under development and tools like this will make a crucial contribution to the success of SAT solver technology in the future, especially with respect to reconfigurable hardware.

Based on the observations above and the potential for greater impact of Satisfiability on solving important problems we can confidently recommend supporting the following lines of research.

1. **Build a next generation of Satisfiability solvers both in hardware and software**. This entails support for new problem solving ideas, particularly in SMT, pseudo-Boolean, Gröbner bases, and other non-clausal methods for solving problems in propositional logic. This also requires a relevant benchmark database for testing: as was mentioned earlier, the SAT competitions are not sufficient for large-scale testing. An important part of this effort will be to support cooperation between industry and academia. Currently, certain companies maintain their own proprietary SAT solvers, the details of which are kept secret. This is inefficient with respect to timely progress in SAT technology. Cooperation needs to be encouraged. This sort of cooperation has worked very well for the open source community where companies have discovered there are creative ways to make money even after giving away their code.

2. **Achieve significant progress in the theory of Satisfiability**. In the past, progress in solvers has been closely related to progress in theory. This has become less so of late, partly because of a trend away from basic research and toward applications. For reasons mentioned above, basic research in SAT technology should be encouraged.

3. **Achieve significant progress in knowledge representation with Satisfiability**. As mentioned above, knowledge representation is now and will further become an important issue for anyone with a combinatorial problem to solve. SAT technology, by its nature, is amenable to a variety of models for representing knowledge and therefore is a prudent choice in supporting research in knowledge representation. A variety of front-ends to SAT representing a variety of models for representing knowledge should be explored and built. The ultimate aim will be to allow non-SAT savvy users to feel assured that their problems will be handled optimally and correctly by SAT technology without the need for understanding the underlying problem solving mechanisms that are used to get the result.

We believe that a concerted effort on the Federal level can result in achieving the goals specified above.

References

[1] D. Achlioptas. Lower bounds for random 3-SAT via differential equations. *Theoretical Computer Science*, **265**:159–185, 2001.

[2] D. Achlioptas, and C. Moore. The asymptotic order of the random k-SAT thresholds. *43rd Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, 2002.

[3] D. Achlioptas, and Y. Peres. The threshold for random k-SAT is $2^k(\ln 2 + o(1))$. *Journal of the American Mathematical Society*, **17**:947–973, 2004.

[4] D. Achlioptas, L.M. Kirousis, E. Kranakis, D. Krizanc, M. Molloy, and Y. Stamatiou. Random constraint satisfaction: a more accurate picture. *3rd Conference on the Principles and Practice of Constraint Programming* (Linz, Austria), LNCS, 1330:107–120, Springer, 1997.

[5] D. Achlioptas, L.M. Kirousis, E. Kranakis, D. Krizanc. Rigorous results for random $(2+p)$-SAT. *Theoretical Computer Science*, **265**(1-2):109–129, 2001.

[6] D. Achlioptas, A. Chtcherba, G. Istrate, and C. Moore. The phase transition in 1-in-k SAT and NAE 3-SAT. In *Proceedings of the 12th ACM-IEEE Symposium on Discrete Algorithms*, 721–722, 2001.

[7] R. Aharoni and N. Linial. Minimal Non-Two-Colorable Hypergraphs and Minimal Unsatisfiable Formulas. *Journal of Combinatorial Theory, Series A*, **43**:196–204, 1986.

[8] S.B. Akers. Binary Decision Diagrams. *IEEE Transactions on Computers*, **C-27**(6):509–516, 1978.

[9] J. Alber, J. Gramm, R. Niedermeier. Faster exact algorithms for hard problems: a parameterized point of view. *Discrete Mathematics*, **229**(1-3):3–27, 2001.

[10] F.A. Aloul, A. Ramani, I.L. Markov, and K.A. Sakallah. Generic ILP versus specialized 0-1 ILP: An update. In *Proceedings of the 2002 International Conference on Computer-Aided Design* (ICCAD '02), 450–457, IEEE Computer Society Press, Los Alamitos, CA, 2002.

[11] D. East and M. Truszczynski. ASPPA An implementation of Answer-Set Programming with propositional schemata. *Proceedings of LPNMR 2001*, 2001.

[12] B. Aspvall, M.F. Plass, and R.E. Tarjan. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8(3):121–132, 1979.

[13] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, **58**(3,Ser. A):295–324, 1993.

[14] N. Bansal, V. Raman. Upper bounds for MaxSat: Further improved. In *Proceedings of 10th Annual conference on Algorithms and Computation*, ISSAC'99, Aggarwal and Rangan (eds.), Lecture Notes in Computer Science, **1741**:247–258, Springer-Verlag, 1999.

[15] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving.* Cambridge University Press, 2003.

[16] R.S. Bayardo and R. Schrag. Using CSP look-back techniques to solve real-world SAT instances. *Proceedings of the* 14^{th} *National Conference on Artificial Intelligence*, 203–208, 1997.

[17] P. Beame, and T. Pitassi. Simplified and improved resolution lower bounds. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, 274–282, IEEE Computer Society Press, Los Alamitos, CA, 1996.

[18] P. Beame, R.M. Karp, T. Pitassi, and M. Saks. On the complexity of unsatisfiability proofs for random k-CNF formulas. In *Proceedings of the 30th Annual Symposium on the Theory of Computing*, 561–571, 1998.

[19] P. Beame, H. Kautz, and A. Sabharwal. On the power of clause learning. In *Proceedings of the* 18^{th} *International Joint Conference in Artificial Intelligence*, 94–99, Acapulco, Mexico, 2003.

[20] E. Ben-Sasson, and A. Wigderson. Short proofs are narrow - resolution made simple. *Journal of the Association for Computing Machinery*, 48:149–169, 2001.

[21] A. Blake. Canonical Expressions in Boolean Algebra. Ph.D. Dissertation, Department of Mathematics, University of Chicago, 1937.

[22] M. Buro and H. Kleine Büning. Report on a SAT Competition. Universität Paderborn, 1992.

[23] B. Bollobás, C. Borgs, J. Chayes, J.H. Kim, and D.B. Wilson. The scaling window of the 2-SAT transition. *Random Structures and Algorithms*, **18**:201–256, 2001.

[24] B. Borchers, J. Furman. A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems. *Journal of Combinatorial Optimization*, **2**(4):299–306, 1999.

[25] E. Boros, Y. Crama, and P.L. Hammer. Polynomial-time inference of all valid implications for Horn and related formulae. *Annals of Mathematics and Artificial Intelligence*, **1**:21–32, 1990.

[26] E. Boros, P.L. Hammer, and X. Sun. The DDT method for quadratic 0-1 optimization. Tech. Report RRR 39-1989, Rutgers, the State University of New Jersey, 1989.

[27] E. Boros, Y. Crama, P.L. Hammer, and M. Saks. A complexity index for satisfiability problems. *SIAM Journal on Computing*, **23**:45–49, 1994.

[28] R.T. Boute. The Binary Decision Machine as a programmable controller. *EUROMICRO Newsletter*, **1**(2):16–22, 1976.

[29] K.S. Brace, R.L. Rudell, and R.E. Bryant. Efficient Implementation of a BDD Package. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, 40–45, IEEE Computer Society Press, 1990.

[30] D. Brand. Verification of large synthesized designs. In *Proceedings of the 1993 International Conference on Computer-Aided Design* (ICCAD '93), 534–537, IEEE Computer Society Press, Los Alamitos, CA, 1993.

[31] R.E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, **C-35**(8):677–691, 1986.

[32] R.E. Bryant. Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams. *ACM Computing Surveys*, **24**(3):293–318, 1992.

[33] V. Chandru, and J.N. Hooker. Extended Horn sets in propositional logic. *Journal of the Association for Computing Machinery*, **38**:205–221, 1991.

[34] M.-T. Chao, and J. Franco. Probabilistic analysis of two heuristics for the 3-Satisfiability problem. *SIAM Journal on Computing*, 15:1106–1118, 1986.

[35] M.-T. Chao, and J. Franco. Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k-satisfiability problem. *Information Sciences*, 51:289–314, 1990.

[36] J. Cheriyan, W.H. Cunningnham, L. Tuncel, Y. Wang. A linear programming and rounding approach to Max 2-Sat. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, **26**:395–414, 1996.

[37] G. Chu, A. Harwood, and P.J. Stuckey. Cache Conscious Data Structures for Boolean Satisfiability Solvers. `http://www.cs.mu.oz.au/~pjs/papers/jsat08.pdf`.

[38] V. Chvátal and B. Reed. Mick gets some (the odds are on his side). *33th Annual Symposium on Foundations of Computer Science*, 620–627, IEEE Computer Society Press, Los Alamitos, CA, 1992.

[39] V. Chvátal and E. Szemerédi. Many hard examples for resolution. *Journal of the Association for Computing Machinery*, **35**:759–768, 1988.

[40] A. Cimatti, E. Giunchiglia, P. Giunchiglia, and P. Traverso. Planning via model checking: a decision procedure for AR. *Lecture Notes in Artificial Intelligence*, **1348**:130–142, Springer-Verlag, New York, 1997.

[41] M. Clegg, J. Edmonds, and R. Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, 174–183, Association for Computing Machinery, New York, 1996.

[42] M. Conforti, G. Cornuéjols, A. Kapoor, K. Vušković, and M.R. Rao. Balanced Matrices. Mathematical Programming: State of the Art. J.R. Birge and K.G. Murty, eds. Braun-Brumfield, United States. Produced in association with the 15th International Symposium on Mathematical Programming, University of Michigan, 1994.

[43] S.A. Cook. The complexity of theorem-proving procedures. *3rd Annual ACM Symposium on Theory of Computing*, 151–158, ACM, New York, 1971.

[44] B. Courcelle. Monadic second-order logic of graphs III: tree-decomposition, minors and complexity issues. *Informatique Theor. et App.*, **26**:257-286, 1992.

[45] N. Creignou and H. Daudé. Generalized satisfiability problems: minimal elements and phase transitions. *Theoretical Computer Science*, **302**(1-3):417–430, 2003.

[46] N. Creignou and H. Daudé. Combinatorial sharpness criterion and phase transition classification for random CSPs. *Information and Computation*, **190**(2):220–238, 2004.

[47] CVC: an SMT solver. `http://www.cs.nyu.edu/acsys/cvc3/`.

[48] E. Dantsin, A. Goerdt, E.A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, and U. Schöning. A deterministic $(2-2/(k+1))^n$ algorithm for k–SAT based on local search. *Theoretical Computer Science*, **289**:69–83, 2002.

[49] E. Dantsin, and A. Wolpert. Derandomization of Schuler's algorithm for SAT. In *Lecture Notes in Computer Science*, **2919**:69–75, Springer, New York, 2004.

[50] E. Dantsin, and A. Wolpert. An improved upper bound for SAT. In *Lecture Notes in Computer Science*, **3569**:400–407, Springer, New York, 2005.

[51] M. Davis, and H. Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, **7**(3):201–215, 1960.

[52] M. Denecker and E. Ternovska. A logic of non-monotone inductive definitions and its modularity properties. *Proceedings of LPNMR*, 2004.

[53] N. Dershowitz, J. Hsiang, G.-s. Huang, and D. Kaiss. Boolean ring Satisfiability. *Proceedings of SAT 2004* , 2004.

[54] M. Davis, G. Logemann, D. Loveland. A machine program for theorem proving. *Communications of the ACM*, **5**:394–397, 1962.

[55] W.F. Dowling, and J.H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, **1**:267–284, 1984.

[56] O. Dubois, and Y. Boufkhad. A general upper bound for the satisfiability threshold of random r-SAT formulae. *Journal of Algorithms*, **24**:395–420, 1997.

[57] O. Dubois. Upper bounds on the satisfiability threshold. *Theoretical Computer Science*, **265**:187–197, 2001.

[58] N. Eén, and N. Sörensson. An extensible SAT solver. *Lecture Notes in Computer Science*, **2919**:502–518, 2004.

[59] N. Eén. Cut Sweeping. Cadence Technical Report CDNL-TR-2007-0510, May (2007). `http://minisat.se/downloads/CutSweeping.ps.gz`.

[60] N. Eén, and N. Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal of Satisfiability, Boolean Modeling and Computation*, **2**:1–26, 2006.

[61] Y. Feldman, N. Dershowitz, and Z. Hanna. Parallel multithreaded Satisfiability solver: design and implementation. *Electronic Notes in Theoretical Computer Science*, **3**:75–90, 2005.

[62] J. Franco, and M. Paull. Probabilistic analysis of the Davis-Putnam procedure for solving the satisfiability problem. *Discrete Applied Mathematics*, 5:77–87, 1983.

[63] J. Franco, and A. Van Gelder. A Perspective on Certain Polynomial Time Solvable Classes of Satisfiability. *Discrete Applied Mathematics*, **125**(2-3):177–214, 2003.

[64] J. Franco, M. Kouril, J. Schlipf, J. Ward, S. Weaver, M. Dransfield, and W.M. Vanfleet. SBSAT: a state-based, BDD-based satisfiability solver. *Lecture Notes in Computer Science*, **2919**:398–410, 2004.

[65] J. W. Freeman, Improvements To Propositional Satisfiability Search Algorithms. Ph.D. Dissertation, University of Pennsylvania, 1995.

[66] E. Friedgut, and an appendix by J. Bourgain. Sharp thresholds of graph properties, and the k-sat problem. *Journal of the American Mathematical Society*, **12**(4):1017–1054, 1999.

[67] Z. Galil. On the complexity of regular resolution and the Davis-Putnam procedure. *Theoretical Computer Science*, **4**:23–46, 1977.

[68] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, San Francisco, 1979.

[69] M. Gelfond, and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings. of the International Joint Conference and Symposium on Logic Programming*, 1070–1080, 1988.

[70] I.P. Gent, and T. Walsh. An empirical analysis of search in GSAT. *Journal of Artificial Intelligence Research*, **1**:47–59, 1993.

[71] S. de Givry, J. Larrosa, P. Meseguer, T. Schiex. Solving MAX-SAT as weighted CSP. *Lecture Notes in Computer Science*, **2833**:363–376, Springer, 2003.

[72] A. Goerdt, and M. Krivelevich. Efficient recognition of random unsatisfiable k-SAT instances by spectral methods. In *Lecture Notes in Computer Science*, **2010**:294–304, 2001.

[73] A. Goldberg. On the complexity of the satisfiability problem. *4th Workshop on Automated Deduction* (Austin, TX), 1–6, 1979.

[74] E. Goldberg and Y. Novikov. BerkMin: a fast and robust SAT solver, In *Proceedings of DATE 2002*, 142–149, 2002.

[75] J. Gramm, E.A. Hirsch, R. Niedermeier, and P. Rossmanith. New worst-case upper bounds for MAX-2-SAT with application to MAX-CUT. *Discrete Applied Mathematics*, **130**(2):139–155, 2003.

[76] J. Gu. Efficient local search for very large scale satisfiability problems. *SIGART Bulletin*, **3**(1):8–12, 1992.

[77] M.T. Hajiaghayi, and G.B. Sorkin. The satisfiability threshold of random 3-SAT is at least 3.52.
Available from http://arxiv.org/pdf/math.CO/0310193.

[78] A. Haken. The intractability of resolution. *Theoretical Computer Science*, **39**:297–308, 1985.

[79] P. Hansen, B. Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, **44**:279–303, 1990.

[80] P.L. Hammer, I. Rosenberg, and S. Rudeanu. On the determination of the minima of pseudo-boolean functions. *Studii si Cercetari Matematice*, **14**:359–364, 1963.

[81] P.L. Hammer, I. Rosenberg, and S. Rudeanu. Application of discrete linear programming to the minimization of Boolean functions. *Revue Roumaine de Mathématiques Pures et Appliquées*, **8**:459–475, 1963.

[82] P.L. Hammer and S. Rudeanu. *Boolean Methods in Operations Research and Related Areas.* Springer-Verlag, Berlin, 1968.

[83] M.J.H. Heule, J. van Zwieten, M. Dufour and H. van Maaren. March_eq: implementing additional reasoning into an efficient lookahead SAT solver. *Lecture Notes in Computer Science*, **3542**:345–359, 2005.

[84] M.J.H. Heule and H. van Maaren. March_dl: adding adaptive heuristics and a new branching strategy. *Journal on Satisfiability, Boolean Modeling and Computation*, **2**:47–59, 2006.

[85] M.J.H. Heule and H. van Maaren. Effective incorporation of double look-ahead procedures. *Lecture Notes in Computer Science*, **4501**:258–271, 2007.

[86] E.A. Hirsch. A new algorithm for MAX-2-SAT. In *Proceedings of 17th International Symposium on Theoretical Aspects of Computer Science*, STACS 2000, Lecture Notes in Computer Science, **1770**:65–73, Springer-Verlag, 2000.

[87] H.H. Hoos. *Stochastic Local Search - Methods, Models, Applications.* Ph.D. Thesis, TU Darmstadt, FB Informatik, Darmstadt, Germany, 1998.

[88] H.H. Hoos, and T. Stützle. Stochastic Local Search. Elsevier, Amsterdam, 2005.

[89] A. Itai, and J. Makowsky. On the complexity of Herbrand's theorem. Working paper 243, Department of Computer Science, Israel Institute of Technology, 1982.

[90] K. Iwama, and S. Tamaki. Improved upper bounds for 3-SAT. In *Proceedings of the* 15th *annual ACM-SIAM Symposium on Discrete Algorithms*, 328–328, Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2004.

[91] R.E. Jeroslow and J. Wang. Solving propositional satisfiability problems. *Annals of Mathematics and Artificial Intelligence*, **1**:167–187, 1990.

[92] T. Jussilla and A. Biere. Compressing bmc encodings with QBF. In *Proceedings of the 44th International Workshop on Bounded Model Checking*, Electronic Notes in Theoretical Computer Science **174**(3):27–39, Elsevier, the Netherlands, 2006.

[93] A.C. Kaporis, L.M. Kirousis, and E.G. Lalas. The probabilistic analysis of a greedy satisfiability algorithm. In *10th Annual European Symposium on Algorithms*, (Rome, Italy), 2002.

[94] A.C. Kaporis, L.M. Kirousis, and E.G. Lalas. Selecting complementary pairs of literals. *Electronic Notes in Discrete Mathematics*, **16**(1):1-24, 2004.

[95] A.C. Kaporis, L.M. Kirousis, and Y.C. Stamatiou. How to prove conditional randomness using the principle of deferred decisions. `http://www.ceid.upatras.gr/faculty/kirousis/kks-pdd02.ps`, 2002.

[96] L. M. Kirousis, E. Kranakis, and D. Krizanc. A better upper bound for the unsatisfiability threshold. In *DIMACS series in Discrete Mathematics and Theoretical Computer Science*, **60**, 1996.

[97] L.M. Kirousis, E. Kranakis, D. Krizanc, and Y.C. Stamatiou. Approximating the unsatisfiability threshold of random formulae. *Random Structures and Algorithms*, **12**:253–269, 1998.

[98] S. Kirkpatrick, and B. Selman. Critical behavior in the satisfiability of random formulas. *Science*, **264**:1297–1301, 1994.

[99] H. Kleine Büning. On subclasses of minimal unsatisfiable formulas. *Discrete Applied Mathematics*, **107**(1-3):83–98, 2000.

[100] H. Kleine Büning, and T. Lettmann. *Propositional Logic: Deduction and Algorithms.* Cambridge University Press, 1999.

[101] F. Krohm, A. Kuehlmann, and A. Mets. The use of random simulation in formal verification. In *Proceedings of the 1996 International Conference on Computer Design* (ICCD '96), 371–377, IEEE Computer Society Press, Los Alamitos, CA, 1996.

[102] A. Kuehlmann, V. Paruthi, F. Krohm, and M.K. Ganai. Robust Boolean reasoning for equivalence checking and function property verification. *IEEE Transactions on Computer Aided Design*, **21**(12):1377–1394, 2002.

[103] A. Kuehlmann. Dynamic transition relation simplification for bounded property checking. In *Proceedings of the 2004 International Conference on Computer-Aided Design* (ICCAD '04), 50–57, IEEE Computer Society Press, Los Alamitos, CA, 1994.

[104] O. Kullmann. Investigations on autark assignments. *Discrete Applied Mathematics*, **107**:99-137, 2000.

[105] O. Kullmann. Lean clause-sets: generalizations of minimally unsatisfiable clause-sets. *Discrete Applied Mathematics*, **130**(2):209–249, 2003.

[106] O. Kullmann. `http://cs.swan.ac.uk/~csoliver/OKsolver.html`.

[107] C.Y. Lee. Representation of Switching Circuits by Binary-Decision Programs. *Bell Systems Technical Journal*, **38**:985–999, 1959.

[108] H.R. Lewis. Renaming a set of clauses as a Horn set. *Journal of the Association for Computing Machinery*, **25**:134–135, 1978.

[109] J. Lewis. Cryptol: specification, implementation and verification of high-grade cryptographic applications. In *Proceedings of the 2007 ACM Workshop on Formal Methods in Security Engineering*, 41–41, 2007.

[110] C.M. Li Anbulagan: look-ahead versus look-back for Satisfiability problems. *Principles and Practice of Constraint Programming*, 341-355, 1997.

[111] H. vanMaaren. A short note on some tractable classes of the satisfiability problem. *Information and Computation*, **158**(2):125–130, 2000.

[112] M. Mahajan, and V. Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms*, **31**:335–354, 1999.

[113] V.W. Marek and M. Truszczyński. Stable logic programming - an alternative logic programming paradigm. In: *25 years of Logic Programming Paradigm*, 375–398, Springer-Verlag, 1999.

[114] J.P. Marques-Silva and K.A. Sakallah. GRASP – A Search Algorithm for Propositional Satisfiability. *IEEE Transactions on Computers*, **48**:506–521, 1999.

[115] MathSat 4, an SMT solver. `http://mathsat4.disi.unitn.it/`.

[116] J. Marques-Silva and I. Lynce. Toward robust CNF encodings of cardinality constraints. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 483–497, 2007.

[117] D. McAllester, B. Selman, and H. Kautz. Evidence for invariants in local search. In *Proceedings of the 14th National Conference on Artificial Intelligence*, 321–326, AAAI Press/The MIT Press, Monlo Park, CA, 1997.

[118] E.I. McCluskey, Jr. Minimization of Boolean functions. *Bell System Technical Journal*, **35**:1417–1444, 1959.

[119] K.L. McMillan. Symbolic Model Checking: An Approach to the State Explosion Problem. Kluwer Academic Publishers, 1993.

[120] A.R. Meyer, and L.J. Stockmeyer. Word problems requiring exponential time. In *Proceedings of the 5th Annual Symposium on the Theory of Computing*, 1–9, Association for Computing Machinery, New York, 1973.

[121] M. Mézard, and Riccardo Zecchina. The random k-satisfiability problem: from an analytic solution to an efficient algorithm. Technical report available from http://arXiv.org 2002.

[122] I. Mironov and L. Zhang. Applications of SAT solvers to cryptanalysis of hash functions. *Lecture Notes in Computer Science*, **4121**:102–115, 2006.

[123] A. Mishchenko, S. Chatterjee, R. Jing, and R. Brayton. FRAIGs: a unifying representation for logic synthesis and verification. Technical Report, UC Berkeley, 2005.
`http://www.eecs.berkeley.edu/\textasciitildebrayton/publications/2005/tech05\_fraigs.pdf`.

[124] D. Mitchell, B. Selman, H. Levesque. Hard and easy distributions for SAT problems. In *Proceedings of the* 10^{th} *National Conference on Artificial Intelligence*, 459–465, 1992.

[125] D. Mitchell, E. Ternovska, F. Hach, and R. Mohebali. Model expansion as a framework for modeling and solving search problems. `http://www.cs.sfu.ca/~mitchell/papers/CMPT2006-24.pdf`.

[126] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic "phase transitions." *Nature*, **400**:133–137, 1999.

[127] B. Monien, and E. Speckenmeyer. 3-Satisfiability is testable in $O(1.62^r)$ steps. Reihe Informatik, Bericht Nr. 3 (1979), Universität-GH Paderborn

[128] B. Monien, and E. Speckenmeyer. Solving Satisfiability in less than 2^n steps. *Discrete Applied Mathematics*, **10**:287–295, 1985.

[129] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC2001)*, 530–535, IEEE Computer Society Press, Los Alamitos, CA, 2001.

[130] G. Nelson and D.C. Oppen. Simplification of cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, **2**:245–257, 1979.

[131] R. Niedermeier. Some prospects for efficient fixed parameter algorithms. In *Proceedings of the 25th Conference on Current Trends in Theory and Practice of Informatics* (SOFSEM'98), *Lecture Notes in Computer Science*, **1521**:168–185, Springer, 1998.

[132] R. Niedermeier, P. Rossmanith. New upper bounds for maximum satisfiability. *Journal of Algorithms*, **36**:63–88, 2000.

[133] S. Plaza, I. Kountanis, Z. Andraus, V. Bertacco and T. Mudge. Advances and insights into parallel SAT solving. *International Workshop on Logic Synthesis*, 2006.

[134] R. Paturi, P. Pudlak, and F. Zane. Satisfiability coding lemma. In *Proceedings of the* 38^{th} *annual IEEE Symposium on Foundations of Computer Science*, 566–574, IEEE Computer Society Press, Los Alamitos, CA, USA, 1997.

[135] R. Paturi, P. Pudlak, and F. Zane. An improved exponential-time algorithm for k-SAT. In *Proceedings of the* 39^{th} *annual IEEE Symposium on Foundations of Computer Science*, 628–637, IEEE Computer Society Press, Los Alamitos, CA, USA, 1998.

[136] J. Pearl. Reverend Bayes on inference engines: a distributed hierarchical approach. In *Proceedings of AAAI, National Conference on Artificial Intelligence*, 133–136, 1982.

[137] P.W. Purdom and G.N. Haven. Probe order backtracking. *SIAM Journal on Computing*, **26**:456–483, 1997.

[138] M. Presburger. Uber die Vollstandigkeit eines gewissen Systems der Arithmetik ganzer Zahlen im welchem die Addition als einzige Operation hervortritt. *Comptes Rendus du I Congres de Mathematicien des pays Slaves*, 92–101, 1929.

[139] W.V.O. Quine. On cores and prime implicants of truth functions. *American Mathematics Monthly*, **66**:755–760, 1959.

[140] N. Robertson and P. Symour. Graph minors: III, Plane tree width. *Journal of Combinatorial Theory*, **36**:49–64, 1984.

[141] J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, **12**:23–41, 1965.

[142] J.A. Robinson. The generalized resolution principle. In *Machine Intelligence*, **3**:77–94, Dale and Michie (eds.), American Elsevier, New York, 1968.

[143] I.G. Rosenberg. Reduction of bivalent maximization to the quadratic case. *Cahiers du Centre d'Etudes de Recherche Operationnelle*, **17**:71–74, 1972.

[144] W.S. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and Systems Sciences*, **4**:177–192, 1970.

[145] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, 410–414, IEEE Computer Society Press, Los Alamitos, CA, USA, 1999.

[146] U. Schöning. A probabilistic algorithm for k-SAT based on limited local search and restart. *Algorithmica*, **32**:615-623, 2002.

[147] R. Schuler, U. Schöning, and O. Watanabe. An improved randomized algorithm for 3-SAT. Technical Report TR-C146, Department of Mathematics and Computer Sciences, Tokyo Institute of Technology, Japan, 2001.

[148] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the 10th National Conference on Artificial Intelligence*, 440–446, AAAI Press/The MIT Press, Menlo Park, CA, 1992.

[149] B. Selman, and H. Kautz. Domain-independent extensions to GSAT: solving large structured satisfiability problems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 290–295, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1993.

[150] J.S. Schlipf, F. Annexstein, J. Franco, and R. Swaminathan. On finding solutions for extended Horn formulas. *Information Processing Letters*, 54:133–137, 1995.

[151] R. Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. *Journal of Algorithms*, **54**(1): 40–44, 2005.

[152] Y. Shang, and B.W. Wah. A discrete Lagrangian-based global-search method for solving satisfiability problems. *Journal of Global Optimization*, **12**(1):61–100, 1998.

[153] N. Shankar. Little engines of proof. In *Proceedings of LICS 2002*, 2002.

[154] C.E. Shannon. A symbolic analysis of relay and switching circuits. Masters Thesis, Massachusetts Institute of Technology, 1940. `http://hdl.handle.net/1721.1/11173`.

[155] C. Sinz. Towards an optimal CNF encoding of Boolean cardinality constraints. In *Proceedings of the 11th Intl. Conf. on Principles and Practice of Constraint Programming* (CP 2005), 827–831, 2005.

[156] M. Sheeran, and G. Stålmarck. A tutorial on Stålmarck's proof procedure for propositional logic. In *Proceedings of the 2nd International Conference on Formal Methods in Computer-Aided Design* (FMCAD'98), 4–6, 1998.

[157] L.J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, **3**:1–22. 1977.

[158] The SMT library. `http://combination.cs.uiowa.edu/smtlib/`.

[159] T. Suyama, M. Yokoo, and A. Nagoya. Solving Satisfiability problems on FPGAs using experimental unit propagation. In *Proceedings of Principle and Practice of Constraint Programming* (CP'99), *Lecture Notes in Computer Science*, **1713**:434–445, Springer, 1999.

[160] R.P. Swaminathan, and D.K. Wagner. The arborescence–realization problem. *Discrete Applied Mathematics*, **59**:267–283, 1995.

[161] S. Szeider. Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. In *Proc. 9th COCOON: Annual International Conference on Computing and Combinatorics*, *Lecture Notes in Computer Science*, **2697**:548–558. Springer, 2003.

[162] G. Tseitin. On the complexity of derivation in propositional calculus. In *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, 115–125, 1968.

[163] A. Urquhart. Hard examples for resolution. *Journal of the Association for Computing Machinery*, 34:209–219, 1987.

[164] B.W. Wah, and Y. Shang. Discrete Lagrangian-based search for solving MAX-SAT problems. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 378–383, 1998.

[165] I. Wegener. BDDs - design, analysis, complexity, and applications. *Discrete Applied Mathematics*, **138**(1-2):229–251, 2004.

[166] N.C. Wormald. Differential equations for random processes and random graphs. *Annals of Applied Probability*, **5**:1217–1235, 1995.

[167] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, **3**:22–33, 1977.

[168] Z. Wu, and B.W. Wah. Trap escaping strategies in discrete Lagrangian methods for solving hard satisfiability and maximum satisfiability problems. In *Proceedings of the 16th National Conference on Artificial Intelligence*, 673–678, AAAI Press/The MIT Press, Menlo Park, CA, USA, 1999.

[169] Z. Wu, and B.W. Wah. An efficient global-search strategy in discrete Lagrangian methods for solving hard satisfiability problems. In *Proceedings of the 17th National Conference on Artificial Intelligence*, 310-315, AAAI Press/The MIT Press, Menlo Park, CA, USA, 2000.

[170] H. Xu, R.A. Rutenbar, K. Sakallah. sub-SAT: A formulation for related boolean satisfiability with applications in routing. In *Proceedings of the 2002 ACM International Symposium on Physical Design*, 182–187, 2002.

[171] Yices, an SMT solver. `http://yices.csl.sri.com/`.

[172] Z3, an SMT solver. `http://research.microsoft.com/projects/Z3/`

[173] X. Zhao, W. Zhang. An efficient algorithm for maximum boolean satisfiability based on unit propagation, linear programming, and dynamic weighting. Preprint, Department of Computer Science, Washington University, 2004.

[174] P. Zhong, P. Ashar, S. Malik, and M. Martonosi. Using reconfigurable computing techniques to accelerate problems in the CAD domain: a case study with Boolean Satisfiability. In *Proceedings of the Design Automation Conference*, 1998.

Appendices

A Program

Workshop On Satisfiability: Assessing the Progress

March 3 - 5, 2008, Maritime Center, Linthicum, Maryland

Monday, March 3

8:30 - 8:45	**Introduction** S. Weaver (US DoD)
8:45 - 9:10	**SAT Solvers: Efficient Implementation** S. Malik (Princeton University)
9:10 - 9:30	**Designing a Practical Hardware Accelerator for SAT using FPGAs** L. Zhang (Microsoft Research)
9:30 - 9:50	**Cut Sweeping: A Scalable Alternative to SAT Sweeping** N. Eén (Cadence)
– – ––	Coffee Break
10:10 - 10:40	**Haplotype Inference with Boolean Satisfiability** I. Lynce
10:40 - 11:20	**From Constraint Programming to Graphical Models; The Role of AND/OR Search** R. Dechter (University of California, Los Angeles)
– – ––	Coffee Break
11:40 - 12:20	**Uses (and abuses) of Groebner Basis in SAT solving** A. Reeves (Center for Computing Sciences)
– – ––	Lunch
1:50 - 2:30	**Building and Using a Model-Expansion Based SAT Front End** D. Mitchell (Simon Fasier University)
2:30 - 3:00	**EXARP and SUBEXP: Algorithms for Explaining Data Sets** K. Truemper & K. Moreland (University of Texas, Dallas)
– – ––	Coffee Break
3:20 - 3:55	**The Physics of Random Formulas** D. Achlioptas (University of California, Santa Cruz)
3:55 - 4:25	**Statistical Message-Passing Techniques for SAT** E. Hsu (University of Toronto)
4:25 - 5:00	**Exploiting Structure and Randomization in SAT Solving and Model Counting** C. Gomes (Cornell University)

8:30 - 9:00	**(Lazy) Satisfiability Modulo Theories** R. Sebastiani (Università di Trento)
9:00 - 9:40	**Trace-Driven Verification of Multi-Threaded Programs** K. Sakallah (University of Michigan)
– – ––	Coffee Break
9:55 - 10:25	**Maximum Satisfiability: New Algorithms and Applications** J. Marques-Silva
10:25 - 11:05	**A Strongly Polynomial Preprocessing of MAX2SAT Instances** E. Boros (Rutgers University)
– – ––	Coffee Break
11:20 - 11:50	**Probability in the Analysis of CNF Satisfiability Algorithms and Properties** J. Franco (University of Cincinnati)
11:50 - 12:20	**Improving the Odds: New Lower Bounds for Van der Waerden Numbers** M. Heule (University of Delft)
– – ––	Lunch
1:50 - 2:20	**Upper Bounds for Satisfiability Solving** E. Speckenmeyer (Universität zu Köln)
2:20 - 2:50	**Expressive Power of Subclasses of Quantified Boolean Formulas** H. Kleine Büning (Univerität Paderborn)
2:50 - 3:10	**Constrained Quantified Formulas** A. Remshagen (University of West Georgia)
– – ––	Coffee Break
3:30 - 4:15	**Knowledge Representation Languages: A Programmer's Interface to Satisfiability** M. Truszczynski (University of Kentucky)
4:15 - 5:00	**Answer Set Programming** V. Marek (University of Kentucky)

8:30 - 9:00	**Some Applications of SAT** D. LeBerre (Université d'Artois)
9:00 - 9:25	**Has Resolution-Based Clause Learning Hit The Wall?** A. Van Gelder (University of California, Santa Cruz)
9:25 - 10:05	**New Theories for SAT** O. Kullmann (Swansea University)
– – ––	Coffee Break
10:15 - 10:40	**Automatic Formal Verification of Pipelined Microprocessors and DSPs** M. Velev (Aries Design Automation)
10:40 - 11:00	**Integrating SAT into ACL2** W. Hunt (University of Texas, Austin)
– – ––	Coffee Break
11:00 - 12:20	**Panel Session - Assessing the Progress** O. Kullmann, M. Saaltink, K. Sakallah, D. Smith

B Biographies

Endre Boros, Rutgers the State University of New Jersey

Dr. Boros studied mathematics at the Eötvös Loránd University in Budapest, Hungary from 1973 to 1978, finishing with an MS degree in Mathematics. He worked as a Research Associate in the Department of Operations Research of the Computer and Automation Institute of the Hungarian Academy of Sciences, in Budapest, Hungary from 1978 to 1989 and received a doctorate in Mathematics in 1985. He also taught part time at the Technical University of Budapest from 1976 to 1978, and at the Eötvös Loránd University from 1979 to 1986. Dr. Boros has been on the faculty of Operations Research at RUTCOR, Rutgers University since 1989 serving as associate professor until 1995, then as professor to the present.

Dr. Boros is Associate Editor of the Annals of Mathematics and Artificial Intelligence since 2000, and a member of the Editorial Boards of the journals Constraints, since 1995, Discrete Applied Mathematics, since 2000, and the Journal of Combinatorial Optimization, since 1995. He has served on the program committees of several international conferences and regularly organizes or co-organizes workshops and sessions at other meetings.

Rina Dechter, University of California, Irvine

Rina Dechter is a professor of Computer Science at the University of California, Irvine. She received her Ph.D. in Computer Science at UCLA in 1985, a M.S. degree in Applied Mathematics from the Weizmann Institute and a B.S in Mathematics and Statistics from the Hebrew University, Jerusalem.

Her research centers on computational aspects of automated reasoning and knowledge representation including search, constraint processing and probabilistic reasoning. Professor Dechter is an author of the book "Constraint Processing" published by Morgan Kaufmann, 2003, and has authored over 100 research papers, and has served on the the editorial boards of: Artificial Intelligence, the Constraint Journal, Journal of Artificial Intelligence Research and the Encyclopedia of AI. She was awarded the Presidential Young investigator award in 1991 and is a fellow of the American association of Artificial Intelligence. She was a Radcliffe fellow from 2005 to 2006.

John Franco, University of Cincinnati

Professor Franco received a B.S. in Electrical Engineering from the City College of New York in 1969. He was a Member of Technical Staff in the Digital Signal Processing group at Bell Laboratories in Holmdel, New Jersey from 1969 to 1975. He earned a M.S. in Electrical Engineering from Columbia University in New York in 1971 and Ph.D. in Computer Science from Rutgers University in New Brunswick, New Jersey in 1981. He served on the faculties of Case Western Reserve University and Indiana University before coming to the University of Cincinnati in 1990. He is associate editor of the Journal of Satisfiability, Boolean Modeling, and Computation, assistant editor of Annals of Mathematics and Artificial Intelligence, theory editor of the Encyclopedia of Computer Science and Engineering, and has co-edited several special issues of AMAI and Discrete Applied Mathematics. He has served on the steering committee of the annual Conference on the Theory and

Applications of Satisfiability Testing from its beginning as the Workshop on Satisfiability in 1996. He has recently co-organized the "Workshop on Satisfiability: Assessing the Progress" which was coupled with this year's High Confidence Software and Systems conference in Baltimore, Maryland.

Franco's early work concentrated on the probabilistic analysis of algorithms for and properties of instances of CNF Satisfiability. Franco later did some theoretical work comparing polynomial time solvable classes of Satisfiability. More recently he has worked with the NSA to research and develop Satisfiability algorithms to enhance formal verification tools. That project resulted in a solver called SBSAT. With John Martin he has contributed the History Chapter of the upcoming Handbook on Satisfiability. His current Satisfiability research interests include adding safe and unsafe constraints, integrating algebraic and search algorithms, and function substitution.

Carla Gomes, Cornell University

Carla P. Gomes obtained a Ph.D. in Computer Science in the area of Artificial Intelligence and Operations Research from the University of Edinburgh. She also holds a M.Sc. in Applied Mathematics from the University of Lisbon. Gomes is the director of the Intelligent Information Systems Institute (IISI) at Cornell. Her research has covered many areas in Artificial Intelligence and Computer Science, including planning and scheduling, integration of constraint and mathematical programming techniques for solving combinatorial problems, complete randomized search methods, and algorithm portfolios. Gomes research spans the full range of theory to applications. Gomes central research themes program are the integration of concepts from mathematical programming with constraint programming; the study of the impact of structure on problem hardness; and the use of randomization techniques to improve the performance of exact (complete) search methods.

Gomes current projects focus on the interplay between problem structure and computational hardness, the use of approximation methods in large-scale constraint-based reasoning systems, and applications of constraint-based reasoning and optimization to combinatorial problems, such as those arising in combinatorial design, autonomous distributed agents, and most recently, game theory and combinatorial auctions.

Gomes is the program co-chair of the Twenty-Third Conference on Artificial Intelligence (AAAI-08) and was the program co-chair of the Ninth International Conference on Theory and Applications of Satisfiability Methods (SAT-06), and the conference chair of the Eighth International Conference on Principles and Practice of Constraint Programming (CP-02).

Gomes is a Fellow of Association for the Advancement of Artificial Intelligence.

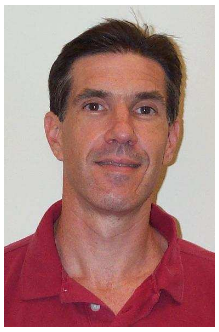
Marijn Heule, University of Delft, the Netherlands

Dr. Heule is a post-doc within the Algorithmics Group of TU Delft, Faculty of Engineering, Mathematics and Computer Science (EWI), Department of Software Technology. His research focuses on the development of algorithms to solve instances of the satisfiability problem (SAT). He developed a well-regarded SAT solver called March based on the results of this research. The March solver has won several awards during the SAT competitions. Recently, he started developing a local search solver called UnitMarch and a cardinality solver. Besides SAT, he is interested in other hard combinatorial problems such as the Van der Waerden numbers. For these numbers, he discovered various improved lower bounds, some of which where obtained using SAT solvers. His research is currently supported by the Dutch Organisation of Scientific Research (NWO), at the TU Delft.

Eric Hsu, University of Toronto, Canada

Eric Hsu is a Ph.D. student researching Artificial Intelligence under Sheila McIlraith's supervision at the University of Toronto. His interests involve combining continuous, that is statistical, techniques with discrete, that is logical, ones. Typically this involves finding the hidden structure in otherwise complex problems.

Warren Hunt, University of Texas, Austin



Dr. Warren A. Hunt, Jr. is a Professor in the Department of Computer Sciences at UT Austin, where he is also a Professor of Electrical and Computer Engineering. Dr. Hunt has a B.S. in Electrical Engineering (1980) from Rice University and a PhD in Computer Sciences (1985) from UT Austin. Dr. Hunt completed the first mechanically verified microprocessor model in 1985, and he later verified the FM9001, the only formally verified microprocessor to ever be manufactured. Dr. Hunt served as Vice President of Computational Logic for ten years and as a manager for IBM Research for five years before joining UT in 2002. Dr. Hunt has published papers in the area of formal hardware and software verification, and in the development of verification tools. He serves on the editorial board of Formal Methods in Systems Design and is Chairman of the FMCAD steering committee.

Henry Kautz, University of Rochester

Henry Kautz joined the faculty of the Department of Computer Science at the University of Rochester in September 2006, and is a member of the UR Center for Computation and the Brain. He directed the Intelligent Systems Research Center at Kodak Research Laboratories from 2006 to 2007. He was a Professor in the Department of Computer Science and Engineering of the University of Washington from 2000 to 2006, after a career at Bell Labs and AT&T Laboratories, where he was Head of the AI Principles Research Department. His academic degrees include an A.B. in Mathematics from Cornell University, an M.A. in Creative Writing from the Johns Hopkins University, an M.Sc. in Computer Science from the University of Toronto, and a Ph.D. in Computer Science from the University of Rochester. He is a Fellow of the American Association for the Advancement of Science, a Fellow of the American Association for Artificial Intelligence, and a recipient of the Computers and Thought Award from the International Joint Conference on Artificial Intelligence. He is the author of the infamous AI limericks.

Hans Kleine-Büning, Universität Paderborn, Germany

Hans Kleine Büning received the Doctor Rerum Naturalium from Westfälische Wilhelms-Universität Münster in 1977. Prof. Dr. Büning has made important contributions to understanding connections between Computer Science and Logic, especially in the areas of function equivalence and quantified Boolean formulas. He is co-author of *Aussagenlogik: Deduktion und Algorithmen*, with Theodor Lettmann (1994), which was translated to English as *Propositional logic: deduction and algorithms* in 1999. He was a co-founder of the successful series *Workshop on Satisfiability* which later became the *International Conference on the Theory and Practice of Satisfiability Testing*.

Michal Kouril, Cincinnati Children's Hospital Medical Center

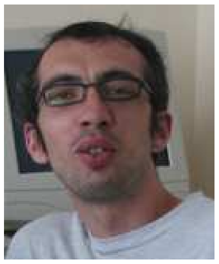
Dr. Kouril received a B.S. in Information Technology from the University of Economics in Prague, Czech Republic in 1997. He worked in industry before starting his graduate work at the University of Cincinnati in 1999. During his time at UC he was part of the core development team for a state-based Satisfiability solver, SBSAT. He was site administrator for the well-known annual Satisfiability competition in 2002 and 2004. He helped organize and was site director of the East Central North American regional ACM programming contest, serving as technical director from 2001 to the present. He received a Ph.D. in Computer Science from UC in 2006 and joined the faculty of the Division of Biomedical Informatics at Cincinnati Children's Hospital Medical Center where he is currently working on applications of HPC and Satisfiability. His interests include applications of Satisfiability, high performance computing, FPGAs and extremal combinatorics, specifically the computability of van der Waerden numbers.

Oliver Kullmann, Swansea University, Wales

Oliver Kullmann studied Mathematics and Computer Science at the University of Frankfurt from 1984 to 1992, receiving the Diploma examination in Mathematics and Computer Science. In 1997 he received a Ph.D. in Mathematics at the University of Frankfurt. From 1987 to 93 he held the position of "Wissenschaftliche Hilfskraft" at the Department of Mathematics, University of Frankfurt and from 1993 to 1998 was "Wissenschaftlicher Mitarbeiter" there. He was visiting researcher at the CWI in Amsterdam in the group of Prof. Jan Friso Groote in 1997, visiting researcher at the Technical University of Delft in the Operations Research group in 1998, from 1999 to 2001 was a postdoctoral fellow at the University of Toronto, Department of Computer Science, in the group of Stephen A. Cook, and in 2001 he was visiting researcher at the Institute of Discrete Mathematics in Vienna. He was awarded the post of Lecturer at the University of Wales Swansea, Computer Science Department, in 2001. In 2003 he was visiting researcher at the Abdus Salam International Centre for Theoretical Physics (ICTP), Trieste, Italy, participating in the Thematic Institute of the Complex Systems Network of Excellence (EXYSTENCE): "Algorithms and challenges in hard combinatorial problems and in optimization under 'uncertainty'." In 2007 he was visiting researcher at the Computer Science Department of the University of Kentucky and in 2008 visiting researcher at the Sun Yat-Sen University, Guangzhou, China.

Dr. Kullmann is best known for his work on autarky assignments and linear autarkies.

Daniel Le Berre, Université d'Artois, France



Premier et second cycle universitaire a l'Université de Bretagne Occidentable (UBO), Brest (Bretagne). Troisième cycle universitaire à l'Université Paul Sabatier, Toulouse III, Midi Pyrénées sous la direction de Michel Cayrol, à l'IRIT, équipe Raisonnement Plausible, Décision et Méthodes de Preuve. Thèse de doctorat intitulée Autour de SAT: le calcul d'impliquants P-restreints, algorithmes et applications soutenue le 12 janvier 2000. Assistant de recherche dans le laboratoire de recherche Business and Technology dirigé par Mary-Anne Williams, à l'Université de Newcastle, Australie, de mars 2000 à août 2001. Maître de conférences à la faculté Jean Perrin depuis septembre 2001.

Undergraduate studies at the Université de Bretagne Occidentable (UBO), Brest (Brittany). PhD thesis defended in January 2000, at IRIT, Université Paul Sabatier, Toulouse III in the Plausible reasoning, Decision and Proof methods research group, supervised by Michel Cayrol. Research associate at the University of Newcastle, Australia, from march 2000 to august 2001. Lecturer at the faculté Jean Perrin since september 2001.

Mes activités actuelles/My current interests: SAT Live!i, SAT4J, SAT Competition, Alloy4Eclipse, Master Pro ILI.

Inês Lynce, INESC-ID Lisboa, Portugal

Inês Lynce has been assistant professor of Computer Engineering in the Instituto Superior Técnico at the Technical University of Lisbon since 2005. She is also a senior researcher at INESC-ID. She has visited Cornell University, New York, Université d'Artois, France, University of St. Andrews, Scotland, University College Cork, Ireland, and the University of Southampton, England. Her research interests are centered in modeling and solving combinatorial problems using efficient techniques including symmetry breaking, constraint programming, and Boolean satisfiability solvers.

Sharad Malik, Princeton University

Sharad Malik received a Bachelor of Technology degree in Electrical Engineering from the Indian Institute of Technology, New Delhi, in 1985. At the University of California, Berkeley, he received an M.S. in Computer Science in 1987 and a Ph.D. in Computer Science in 1990. At Princeton University he was Assistant Professor from 1991 to 1996, Associate Professor from 1996 to 1999, Professor from 1999 to 2005, and is now George Van Ness Lothrop Professor of Engineering, Director of the Center for Innovation in Engineering Education, and Associate Director of the Gigascale Systems Research Center (GSRC). He spent the summer of 1989 as a research intern at AT&T Labs in Murray Hill, New Jersey.

Dr. Malik received the President of India's Gold Medal in 1985 for undergraduate academic excellence, an NSF research initiation award in 1992, the Walter C. Johnson prize for teaching excellence in 1993, became an IEEE fellow in 2002 and received an IBM faculty award in 2006 and 2007, in addition to numerous other awards and honors.

Dr. Malik holds 16 U.S. patents, has been on the program committee of many conferences including SAT'03, SAT'04, SAT'05, and SAT'06, and has been on the editorial board of the Journal of Satisfiability, Boolean Modeling, and Computation.

João Marques-Silva, University of Southampton, United Kingdom

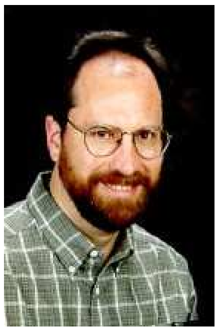
João Marques-Silva is a professor of Computer Science in the School of Electronics and Computer Science at the University of Southampton, United

Kingdom. He received a Ph.D. degree in Electrical Engineering and Computer Science in 1995 from the University of Michigan, Ann Arbor, USA, and the habilitation degree in Computer Science from the Technical University of Lisbon, Portugal, in 2004. His research interests include formal methods, namely system specification, verification and model checking, algorithms for Boolean satisfiability and extensions, and application of formal methods in artificial intelligence, computational biology, and design automation.

Victor Marek, University of Kentucky

Victor W. Marek is a Professor of Computer Science at the University of Kentucky. He received his Ph.D. from the University of Warsaw, Warsaw, Poland. Dr. Marek is an author of over 160 technical papers published in journals and reviewed conference proceedings. Dr. Marek is an author of three books, including the monograph of Nonmonotonic Logic co-authored with Dr. Truszczynski. Research interests of Dr. Marek include: knowledge representation, logic as a computational mechanism, logic programming, and constraint satisfaction. Dr. Marek serves on the editorial boards of four journals: Fundamenta Informaticae, Studia Logica, Journal of Logic and Computation and Central European Journal of Mathematics. Besides of teaching at the University of Kentucky, Dr. Marek spent several years in other universities, including Cornell University, University of of California, San Diego and other research institutions.

David Mitchell, Simon Fraser University



My research primarily is about aspects of the following general scheme for representing and solving problems: We axiomatize the properties of (solutions to) the problem in some logic, such that models of the axioms amount to solutions of the problem, and then we use a model-finder for the logic in question to obtain solutions.

Most of my time currently is devoted to the MX Project, for representing a solving search problems in logic. The project is joint work with E. Ternovska, and involves many students, post-docs, and other colleagues. A search problem is axiomatized in classical logic (extended with Inductive Definitions and possibly other operators), in such a way that an instances of problem is a finite structure, and solution are expansions of the structure that satisfy the formula.

On-going work on that project includes:

1. Extending the basic formal framework to include useful modeling features (such as "built-in" arithmetic, strings, and aggregate operators such as cardinality constraints), and resolving questions of the complexity and expressiveness of various fragments and extensions;
2. Design and implementation of tools for solving (NP) search problems, including a grounder for FO(ID) model expansion, and new solvers for SAT extended with extra connectives ; We currently have a solver implementation that is as good or better than existing tools based on related ideas.
3. Development of techniques for solving problems at the second level of the polynomial hierarchy;

4. Applying the approach to problems such as verification of hardware and software, planning, computation biology (e.g., phylogeny and haplotyping), finding Van der Waerden numbers, and others.

Katherine Moreland, University of Texas, Dallas

This past summer I began an internship in the information technology field as a computer programmer. Although I had excelled in the classroom, I was somewhat unprepared for the transition into the workplace. Suddenly I was given real-life problems to solve, not just those from a textbook. Also, I had to collaborate with others who each contributed a piece of a larger project. This experience was very new to me and it took time to adjust. I believe it is imperative that the curriculum be altered to better prepare engineering and computer science students for the transition to the workplace so that not only may they excel to their highest potential, but so they may also reach a higher level of productivity earlier in their careers to advance the companies they work for and the people that are benefited by the products they produce.

The Texas Engineering and Technical Consortium strives to achieve these goals by improving freshmen curriculums as well as offering the All Across Texas Database which helps to place computer science and engineering students in summer internship positions where they may begin to learn the valuable skills required to excel not only in the classroom, but also in a work environment. With funding we can better prepare the students of today to become the work force of tomorrow.

Alyson Reeves, Center for Computing Sciences, Bowie, Maryland

Alyson Reeves received a Ph.D. in Mathematics from Cornell University in 1992 under Michael Stillman and Peter Kahn. She has described a parallel implementation of the Gröbner bases algorithm based on the well-known sequential program Macaulay. Like Macaulay, the implementation computes Gröbner bases only over integers mod p, where p is a prime number no greater than 31991.

Anja Remshagen, University of West Georgia

Anja Remshagen received a M.S. in Mathematics from Universität Köln, Germany, in 1998 and a Ph.D. in Computer Science from the University of Texas at Dallas in 2001. Since then she has been on the faculty of the University of West Georgia where she currently holds the position of Associate Professor.

Remshagens interests include computational logic, quantified Boolean formulas, satisfiability, computer science education, data mining and intelligent systems, combinatorics, and graph theory.

Karem Sakallah, University of Michigan

Karem Sakallah received a bachelors degree in electrical engineering from American University of Beirut, Lebanon, in 1975. He earned a masters degree in electrical engineering and a doctorate in electrical and computer engineering from Carnegie Mellow University (CMU) in 1977 and 1981, respectively. He joined the Electrical Engineering Department at CMU in 1981 as a visiting professor. From 1982 to 1988, he worked for Digital Equipment Corporation in Hudson, Massachusetts, where he headed the Analysis and Simulation Advanced Development team. Since 1988, he has been a professor of electrical engineering and computer science.

Dr. Sakallah pioneered a rigorous methodology for the modeling, simulation and optimization of integrated circuits. The methodology has yielded fundamental insights about the relationship between the complementary discrete and continuous representations of digital ICs and has led to major innovations in simulation and timing verification technology. These include the SAMSON mixed analogy/digital event- driven simulator, the checkTc and minTc tools for verifying and optimizing the performance of level-clocked multiphase synchronous circuits, and the GRASP tool suite for functional timing analysis based on fast Boolean search. His most recent contribution has been the development of a symbolic functional logic waveform model centered on a continuous-time differential calculus. This "waveform calculus" allows the transient behavior of logic waveforms to be elegantly described in terms of appropriate time derivatives, and is amenable to precise functional and temporal abstractions that can be applied to generate compact high-level timing views with guaranteed accuracy bounds.

He was associate editor of the IEEE Transactions on CAD during 1995-1997 and has served on the program committees of ICCAD, DAC, ICCD, and numerous other workshops. He has published more than 90 papers and has presented seminars and tutorials at many professional meetings and various industrial sites. He is a member of ACM and Sigma Xi.

Roberto Sebastiani, Università di Trento, Italy

Professor Sebastiani is Associate Professor at the Faculty of Science, Department of Information Science and Engineering (DISI) at University of Trento, Italy. He received a M.S. in Electronic Engineering at University of Padova, Italy (1991), 110/110 cum Laude, and a Ph.D. in Computer Science Engineering at University of Genova, Italy (1997). His Ph.D. thesis dealt with SAT techniques for modal and description logics. During his PhD he has been twice a visiting scholar at Stanford University (1994 and 1996). He is associate editor of the Journal on Satisfiability, Boolean Modeling, and Computation (JSAT). He has been the co-Editor of the Journal of Symbolic Computation, Special Issue on the Integration of Automated Reasoning and Computer Algebra Systems, 2002, and of JSAT, Special issue on Satisfiability Modulo Theories (2007). He is/has been in the PC of many international conferences and workshops, including SAT, IJCAR, KR, AIMSA, FROCOS, PDPAR, SMT, CALCULEMUS.

Professor Sebastiani has published more than 60 scientific refereed papers. His current research focuses mostly on SATISFIABILITY MODULO THEORIES (SMT) and its applications to formal verification. He is one of the inventors of the lazy approach to SMT, which efficiently combines DPLL-based SAT solvers with theory-specific decision procedures. Professor Sebastiani has contributed to both Chapter 7 ("SAT techniques for modal and description logics") and 8 ("Satisfiability Modulo Theories") in part 4 of the upcoming Handbook of Satisfiability, and he has recently published a 84-page survey paper on SMT in JSAT. He has given tutorials and invited talks on SMT and SMT-related topics at many international schools (ESSLI'02, BIT'05, SFM'06) and conferences (CADE03, IJCAI03, STRATEGIES'04, FROCOS07, Daghstul'07, DOD-SAT'08). Professor Sebastiani and Dr. Alessandro Cimatti lead the development of the MathSAT SMT solver (www.mathsat4.unitn.it).

They have received a grant from Intel for using SMT techniques for the verification of RTL circuit designs.

Ewald Speckenmeyer, Universität zu Köln, Germany

Bio of Ewald Speckenmeyer.

Klaus Truemper, University of Texas, Dallas

Professor Truemper received a B.S. in Industrial Engineering from the Rheinische School of Engineering, Köln, Germany, in 1965, an M.S. in Industrial Engineering from the University of Iowa in 1968, and a Ph.D. in Operations Research from Case Western Reserve University in 1973. He joined the University of Texas at Dallas 1973 and is now Professor Emeritus at that university. In 1988, he received the Senior Distinguished U.S. Scientist Award by Alexander von Humboldt-Stiftung, Germany.

Professor Truemper is the author of three books on matroid theory, logic computation, and design of intelligent systems. He has created the Leibniz System for the construction of intelligent systems. The modules of the system carry out logic computation, learning from data, and combinatorial decomposition.

His current interest is focused on learning explanations from data that are comprehensible by humans. Application areas range from engineering, finance, medicine, molecular biology, to security.

Mirek Truszczynski, University of Kentucky

Miroslaw Truszczynski is a Professor of Computer Science at the University of Kentucky. He received a Ph.D. degree from the Warsaw University of Technology in 1980. From 1993 to 2007 he served as the department chair. Dr. Truszczynski's research interests include knowledge representation, non-monotonic reasoning, logic programming, and constraint programming. He authored and co-authored of over 130 technical papers. Jointly with Victor Marek he wrote a research monograph "Nonmonotonic Logic," which marked a milestone in the development of the field. In the late 1990s, he helped found answer-set programming as a computational paradigm for knowledge representation. Dr. Truszczynski is active in his research community. He is a member of the Executive Committee of the Association of Logic Programming, Steering Committee of the Knowledge Representation, Inc., and Chair of the Steering Committee of Nonmonotonic Reasoning Workshops. He also serves on editorial and advisory boards of several professional journals, including Journal of Artificial Intelligence Research, Theory and Practice of Logic Programming, and AI Communications.

Sean Weaver, U.S. Department of Defense

Sean Weaver received a Master of Computer Science from the University of Cincinnati (UC) in 2004. Since 2004, he has been pursuing a PhD in Computer Science from UC and working for NSA's Information Assurance Directorate where he researches techniques to aid NIAP on EAL 6/7/7+ Common Criteria evaluations of safety and security critical applications. He is the co-author and maintainer of SBSAT, a state-based Satisfiability solver. His current research interests include modeling and solving combinatorial problems related to formal verification.

Allen Van Gelder, University of California, Santa Cruz

Allen Van Gelder holds a B.S. in Mathematics from the Massachusetts Institute of Technology and a Ph.D. in Computer Science from Stanford University. He is current Professor of Computer Science at the University of California, Santa Cruz.

Van Gelder's research interests include development of algorithms for propositional satisfiability, methods for verifiable software, theorem proving, analysis of algorithms, parallel algorithms, computer graphics, and scientific visualization. He received a National Science Foundation Presidential Young Investigator Award in 1989 to investigate the use of logic programming for problems in database and artificial intelligence systems. He is best known for the ground-breaking development of the Well-founded semantics with J.S. Schlipf and K.A. Ross.

He is on the editorial board of the Journal on Satisfiability, Boolean Modeling, and Computation and has an Erdös number of 3.

Miroslav Velev

Miroslav Velev holds a B.S. and M.S. in Electrical Engineering and a B.S. in Economics from Yale University (1994) and a Ph.D. in Electrical and Computer Engineering from Carnegie Mellon University (2004) under mentorship of Prof. Randal E. Bryant. He has served as adjunct assistant professor in the Department of Electrical and Computer Engineering at the University of Illinois at Chicago since 2006, and was visiting assistant professor in the School of Electrical and Computer Engineering at the Georgia Institute of Technology from 2002 to 2003. During a summer internship at Motorola, Austin, TX, in 1998 he worked on the formal verification of the M.CORE microprocessor.

As a Ph.D. student he developed an efficient memory model for behavioral abstraction of memory arrays in symbolic ternary simulation. This model was adopted by Intel, NEC, and Motorola in internal tools, by Synopsys in a prototype of a commercial tool, and by Innologic Systems in a commercial tool. He also developed a tool flow and formal verification techniques that are highly automatic, and significantly outperform all previous methods for formal verification of pipelined microprocessors. This tool flow was used to formally verify a model of the M.CORE processor at Motorola, and detected bugs.

He was the winner of the EDAA Outstanding Dissertation Award for topic "New directions in logic and system design" in 2005 and won the Franz Tuteur Memorial Prize for the most outstanding senior project in Electrical Engineering at Yale in 1994.

Velev's research interests include formal verification of microprocessors, decision procedures, Boolean Satisfiability (SAT), SAT Solvers, computer-aided design for VLSI, constraint satisfaction problems (CSPs), computer architecture, computer engineering, and artificial intelligence.

He is a member of the editorial boards of the Journal on Satisfiability, Boolean Modeling, and Computation and the Journal of Universal Computer Science.

Lintao Zhang, Microsoft Research, Mountain View

Lintao Zhang earned a Ph.D. degree in Electrical Engineering from Princeton University in 2003. Since then he has been a researcher at Microsoft Research Silicon Valley in Mountain View California. His interests include verification and logic, distributed systems, storage, computer architecture and reconfigurable computing.