SAT Competition 2016: Recent Developments

Tomáš Balyo
Karlsruhe Institute of Technology
Karlsruhe, Germany

Marijn J.H. Heule
Department of Computer Science
The University of Texas at Austin, USA

Matti Järvisalo
HIIT, Department of Computer Science
University of Helsinki, Finland

Abstract

We give an overview of SAT Competition 2016, the 2016 edition of the famous competition for Boolean satisfiability (SAT) solvers with over 20 years of history. A key aim is to point out "what's hot" in SAT competitions in 2016, i.e., new developments in the competition series, including new competition tracks and new solver techniques implemented in some of the award-winning solvers.

Introduction

Boolean satisfiability (SAT) is one of the great success stories of modern computer science. Despite—and to an extent in conflict with—the traditional view of NP as a true sign of 'intractability', decision procedures for SAT, i.e., *SAT solvers*, are today frequently used as the core workhorses in tackling real-world instances of a wide range of NP-hard search and optimization problems, from AI applications to industrial use such as in tools for hardware and software verification. SAT Competitions—the history of which dates back to early 90s (Buro and Bühning 1993; Johnson and Trick 1996), variants of which have been yearly organized since 2002 (Simon, Berre, and Hirsch 2005; Le Berre and Simon 2004; 2005; Balint et al. 2015; Järvisalo et al. 2012; Balyo et al. 2016)—have been a driving force of SAT solver development, to the point that the performance of contemporary SAT solvers is noticeably better than that of those from one to two decades ago. A further central role of the SAT Competition series is to provide the research community with systematically constructed benchmark sets on a yearly basis, which nowadays have become a key ingredient in empirical evaluation of SAT solver techniques in published SAT research.

The 2016 SAT solver competition was the 10th edition organized under the name "SAT Competition". We give a brief overview of SAT Competition 2016 (SC 2016), with a specific aim of pointing out "what's hot" in SAT solver competitions in 2016, i.e., new developments in the competition series, including new competition tracks and new solver techniques implemented in some of the award-winning solvers.

Competition Tracks

The 2016 competition featured six tracks. Additionally to the Main, Parallel, and Random tracks, which have a long tradition in SAT competitions we had an Incremental Library track (introduced in the 2015 SAT Race (Balyo et al. 2016)) and two new tracks—Agile and No-Limits. In contrast to earlier editions, there were no longer special tracks for only satisfiable and only unsatisfiable benchmarks.

Solvers in all tracks were ranked based on the number of solved instances within the timeout[1]. The Agile track had a 60-second timeout, while the other tracks used a 5000-second timeout. Satisfiable benchmarks were considered solved if the solver produced a correct solution, except for the No-Limit track in which only a claim of satisfiability was required. In the Main track, we considered unsatisfiable benchmarks solved only if the solver produced a valid DRAT proof (Heule 2016), which was checked by the DRAT-trim checker (Wetzler, Heule, and Jr. 2014). In the other tracks, only a claim of unsatisfiability was required. Within each track, claiming satisfiable on an unsatisfiable benchmark or the other way around resulted in disqualification in the track.

The Agile track was established for SAT solvers with low overhead, suitable for rapidly solving large numbers of not too hard SAT instances. For a wide range of applications, SAT solvers are used as "core NP solvers" that are called iteratively, potentially thousands or even hundreds of thousands of times, as part of a single execution of a more complex procedure. Concrete examples of such use cases of SAT solvers come from Satisfiability Modulo Theories (SMT) (De Moura and Bjørner 2011), SAT-based counterexample-guided abstraction refinement (Clarke et al. 2003), hardware model checking (Clarke et al. 2001), Boolean optimization, etc.

The No-Limits track was introduced to remove all the restrictions for participation enforced in the other tracks, such as allowing portfolios and closed source solvers. The other tracks were restricted to so-called "single-engine" SAT solvers. Only brand new benchmarks were used in this track to prevent aggressive over-tuning to known benchmarks. By removing the source code requirement we wanted to increase non-academic participation, but this was futile.

[1]Except for the Incremental Library track, see the SC 2016 homepage (Balyo, Heule, and Järvisalo 2016a) for more details.

Table 1: Top-3 solvers of the SC 2016 tracks. The numbers of solved benchmark instances are shown in parenthesis.

Track	1st Place	2nd Place	3rd Place
Main (500)	MapleCOMSPS (203)	Riss (202)	Lingeling (201)
Parallel (500)	Treengeling (315)	Plingeling (302)	CryptoMiniSat (297)
Random (240)	Dimetheus (95)	CSCCSat (89)	DCCAlm (88)
Incremental	CryptoMiniSat	Glucose	Riss
No-Limits (350)	BreakIDCOMSPS (178)	Lingeling (162)	abcdSAT (161)
Agile (5000)	Riss (3284)	TB_Gluc. (3187)	CHBR_Gluc. (3179)

Winners: What's Hot in 2016

The top-3 solvers of each of the SC 2016 tracks are listed in Table 1. The difference between the top solvers tended to be very small in the Main track: the number one solved only two instances more than the number three (203 versus 201). As a consequence, the ranking could have been different with a slightly different timeout or slightly different benchmark selection procedure. We briefly describe some of the winners in the following.

MapleCOMPSP (Liang et al. 2016) Variations of the VSIDS (Moskewicz et al. 2001) decision heuristic are used by essentially all top solvers in the recent competitions. MapleCOMPSP uses a new branching heuristic called *learning rate branching (LRB)* for the first 2500 seconds of the search (then switching to VSIDS). The aim of LRB is to maximize the quantity of learned clauses by selecting variables with a high learning rate, i.e., variables that after being assigned are likely to drive the search into a conflict (where clauses are learned). Since learning rate values are difficult and expensive to compute they are estimated using multi-armed bandits, a special case of reinforcement learning.

BreakIDCOMiniSatPS (Devriendt et al. 2016) Many SAT problems exhibit symmetries but top solvers rarely exploit this. BreakIDCOMiniSatPS combines the symmetry breaking preprocessor BreakID with the SAT solver COMiniSatPS. Symmetries are detected using Saucy (Katebi, Sakallah, and Markov 2010), broken via BreakID, and symmetry-breaking predicates are added to the input formula. Although symmetry-breaking techniques can be expressed in the DRAT proof format (Heule, Hunt Jr., and Wetzler 2015), BreakIDCOMiniSatPS does not support unsatisfiability proof generation. It therefore participated only in the No-Limits track, in which it won very decisively.

Dimetheus (Gableske 2013; Gableske, Müelich, and Diepold 2013) The Random track used to be dominated by relatively simple local search solvers implementing some variant of WalkSAT. This changed in 2014, when the solver Dimetheus outperformed all such solvers during the SAT competition. Dimetheus combines message passing algorithms with more conventional local search techniques and was again very successful.

Treengeling (Biere 2016) Solvers in the Parallel track ran on nodes with 24 cores (48 with hyper-threading) and 96 gigabytes of memory. This setup appeared tricky for some solvers and resulted in many Out of Memory errors. The most successful parallel solver was Treengeling based on the Cube-and-Conquer paradigm (Heule et al. 2012), which uses look-ahead techniques to partition the problem and applies CDCL solvers to the subproblems. Cube-and-conquer solvers hardly share clauses between threads, which reduces the memory footprint. This is probably one of the factors that facilitated its strong performance.

Further Considerations

The benchmark selection procedure has a big impact on the results of the competition. In the last couple of years, benchmarks were selected by taking the performance of the prior year's top solvers into account: the benchmark suite was constructed in such a way that these top solvers would perform comparably. The rationale of such a selection is that 1) the competition is expected to be competitive and 2) progress compared to the prior year is awarded. There is also a clear disadvantage: instances that only one top solver can handle tend to be discarded from the competition suite.

This year we decided to select benchmarks based on their hardness. We distinguished between *easy*, *medium*, and *hard*. Benchmarks were considered *easy* if an established sequential solver, MiniSAT 2.2 (Eén and Sörensson 2004), required less than 600 seconds to solve it. The non-easy benchmarks were labeled *medium* if at least one top parallel solvers of SAT Race 2015, Plingeling, Treengeling, and Glucose-Syrup, could solve it in 600 seconds (wall-clock time), and *hard* otherwise. The far majority of submitted benchmarks were *easy*. As a consequence, we ended up selecting most of the submitted *medium* and *hard* benchmarks. This of course introduces a bias as well, since the submitted benchmarks were not a good representation of a wide spectrum of applications that SAT solvers are used for.

The core focus of the competition is to stimulate innovation of SAT solving techniques. In the future we will aim to further improve the benchmark selection procedure by avoiding any bias and with this focus in mind.

Beyond the brief overview provided here, we encourage the reader to visit the SC 2016 website (Balyo, Heule, and Järvisalo 2016a) for full details on the competition, including the competition rules, list of participated solvers, and full runtime logs for all participating solvers. Furthermore, the website offers the 2016 edition of the SAT Competition benchmark suites partitioned in terms of the traditional main tracks: application, crafted, and random instances.

A further valuable resource for more up-to-date information on "what's hot" in 2016 in SAT solving is the SC 2016 proceedings (Balyo, Heule, and Järvisalo 2016b), a collection of short (1–2 page) unedited descriptions of each SAT solver that participated in SC 2016, provided by the authors of the solvers. Together with the solver descriptions, the proceedings provide descriptions of new benchmarks submitted to the 2016 competition. In fact, providing a solver description was mandatory for being eligible to compete in SC 2016, following the tradition started by SAT Challenge 2012, the main SAT solver competition of 2012. The proceedings hence provide a detailed documentation of new developments in SAT solvers and benchmarks, for both present and future purposes.

References

Balint, A.; Belov, A.; Järvisalo, M.; and Sinz, C. 2015. Overview and analysis of the SAT Challenge 2012 solver competition. *Artificial Intelligence* 223:120–155.

Balyo, T.; Biere, A.; Iser, M.; and Sinz, C. 2016. SAT Race 2015. *Artificial Intelligence* 241:45–65.

Balyo, T.; Heule, M.; and Järvisalo, M. 2016a. Homepage of 2016 sat competition. `http://baldur.iti.kit.edu/sat-competition-2016`.

Balyo, T.; Heule, M.; and Järvisalo, M., eds. 2016b. *Proceedings of SAT Competition 2016: Solver and Benchmark Descriptions*, volume B-2016-1 of *Department of Computer Science Series of Publications B*. University of Helsinki.

Biere, A. 2016. Splatz, Lingeling, Plingeling, Treengeling, YalSAT entering the SAT Competition 2016. In Balyo, T.; Heule, M.; and Järvisalo, M., eds., *Proceedings of SAT Competition 2016; Solver and Benchmark Descriptions*, volume B-2016-1 of *Department of Computer Science Series of Publications B*, 44–45. University of Helsinki.

Buro, M., and Bühning, H. K. 1993. Report on a SAT competition. *Bulletin of the European Association for Theoretical Computer Science* 49:13–151.

Clarke, E.; Biere, A.; Raimi, R.; and Zhu, Y. 2001. Bounded model checking using satisfiability solving. *Formal Methods in System Design* 19(1):7–34.

Clarke, E.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2003. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM* 50(5):752–794.

De Moura, L., and Bjørner, N. 2011. Satisfiability modulo theories: Introduction and applications. *Communications of the ACM* 54(9):69–77.

Devriendt, J.; Bogaerts, B.; Bruynooghe, M.; and Denecker, M. 2016. Improved static symmetry breaking for SAT. In Creignou, N., and Berre, D. L., eds., *Proc. SAT*, volume 9710 of *Lecture Notes in Computer Science*, 104–122. Springer.

Eén, N., and Sörensson, N. 2004. An extensible sat-solver. In Giunchiglia, E., and Tacchella, A., eds., *SAT 2002 Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, 502–518. Springer.

Gableske, O.; Müelich, S.; and Diepold, D. 2013. On the performance of CDCL-based message passing inspired decimation using Rho-Sigma-PMP-i. In *Pragmatics of SAT Workshop*.

Gableske, O. 2013. On the interpolation between product-based message passing heuristics for SAT. In Järvisalo, M., and Gelder, A. V., eds., *Proc. SAT*, volume 7962 of *Lecture Notes in Computer Science*, 293–308. Springer.

Heule, M.; Kullmann, O.; Wieringa, S.; and Biere, A. 2012. Cube and Conquer: Guiding CDCL SAT solvers by lookaheads. In Eder, K.; Lourenço, J.; and Shehory, O., eds., *Proc. HVC 2011*, volume 7261 of *Lecture Notes in Computer Science*, 50–65. Springer.

Heule, M. J. H.; Hunt Jr., W. A.; and Wetzler, N. D. 2015. Expressing symmetry breaking in DRAT proofs. In Felty, A. P., and Middeldorp, A., eds., *Proc. CADE*, volume 9195 of *Lecture Notes in Computer Science*, 591–606. Springer.

Heule, M. J. H. 2016. The DRAT format and DRAT-trim checker. https://arxiv.org/abs/1610.06229.

Järvisalo, M.; Le Berre, D.; Roussel, O.; and Simon, L. 2012. The international SAT solver competitions. *AI Magazine* 33(1):89–92.

Johnson, D., and Trick, M., eds. 1996. *Second DIMACS implementation challenge: Cliques, coloring and satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society.

Katebi, H.; Sakallah, K. A.; and Markov, I. L. 2010. Symmetry and satisfiability: An update. In Strichman, O., and Szeider, S., eds., *Proc. SAT*, volume 6175 of *Lecture Notes in Computer Science*, 113–127. Springer.

Le Berre, D., and Simon, L. 2004. The essentials of the SAT 2003 competition. In Giunchiglia, E., and Tacchella, A., eds., *SAT 2003 Selected Papers*, volume 2919 of *Lecture Notes in Computer Science*, 452–467. Springer.

Le Berre, D., and Simon, L. 2005. Fifty-five solvers in vancouver: The SAT 2004 competition. In Hoos, H. H., and Mitchell, D. G., eds., *SAT 2004 Selected Papers*, volume 3542 of *Lecture Notes in Computer Science*, 321–344. Springer.

Liang, J. H.; Ganesh, V.; Poupart, P.; and Czarnecki, K. 2016. Learning rate based branching heuristic for SAT solvers. In Creignou, N., and Le Berre, D., eds., *Proc. SAT*, volume 9710 of *Lecture Notes in Computer Science*, 123–140. Springer.

Moskewicz, M. W.; Madigan, C. F.; Zhao, Y.; Zhang, L.; and Malik, S. 2001. Chaff: Engineering an efficient SAT solver. In *Proc. DAC*, 530–535. ACM.

Simon, L.; Berre, D. L.; and Hirsch, E. A. 2005. The SAT2002 competition. *Annals of Mathematics and Artificial Intelligence* 43(1):307–342.

Wetzler, N.; Heule, M.; and Jr., W. A. H. 2014. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In Sinz, C., and Egly, U., eds., *Proc. SAT*, volume 8561 of *Lecture Notes in Computer Science*, 422–429. Springer.