Designing a solver competition: the QBFEVAL'10 case study

Claudia Peschiera
University of Genoa, Italy, Luca Pulina
University of Genoa, Italy and Armando Tacchella
University of Genoa, Italy

Abstract

In this paper we report about QBFEVAL'10, the seventh in a series of events established with the aim of assessing the advancements in reasoning about quantified Boolean formulas (QBFs). The paper discusses the results obtained and the evaluation setup, from the criteria used to select QBF instances down to the hardware infrastructure. We also discuss the current state-of-the-art in light of past challenges and we envision future research directions that are motivated by the results of QBFEVAL'10.

1 Introduction

For almost a decade now, competitive events in the field of Boolean reasoning have influenced related research agendas and shaped the course of tool developments. Nowadays, organized evaluations are popular for several subfields of Boolean reasoning, including propositional satisfiability (SAT) [3, 27], quantified Boolean formula (QBF) [20], MAXSAT [1], pseudo-Boolean [18], and satisfiability modulo theory (SMT) solving [2]. While in all these events the organizers are trying to answer the question "Which solver should I use?" [15] using transparent and fair evaluation methods, researchers and practitioners look at the outcomes as the preferred way to understand the current state of the art in each subfield. Often, it is not just the best solver for the given application which is sought, but also some feeling about what makes a solver suitable for one kind of problem or another. It is therefore important, in any such event, to motivate and explain thoroughly the testing methods which led to the latest results. Whenever possible, the organizers should try to answer the question "Which solver is best, for which task and why?", which is far more relevant than just assessing the winner(s) of the event.

In this spirit, we report about the 2010 evaluation of QBF solvers (QBFEVAL'10), the seventh in a series of events established with the aim of assessing the advancements in QBF reasoning. An important expansion with respect to the last evaluation (QBFEVAL'08) is the introduction of five different tracks, each with its own rules, solvers and instances. Overall, QBFEVAL'10 received 13 solver submissions from 8 different developing teams. Also, a pool of 136 formulas was submitted about module extraction in description logics [13]. These formulas have been made available both with a prenex CNF encoding and a non-prenex non-CNF encoding.

Table 1 shows QBFEVAL'10 at a glance. The main competition (MAIN in the following) is comprised only of prenex CNF formulas obtained by encoding various automated reasoning tasks into QBF. This track is competitive, i.e., we declare a winner in the end, whereas all the remaining tracks are non-competitive. This is because evaluation of QBFs in prenex CNF is a fairly mature area, faring more than ten years of active research into algorithms, heuristics and optimizations. On the other hand, we preferred to have non-competitive tracks for formulas with a single $\forall\exists$ alternation (2QBF), "small but hard" instances (SH), and random generated formulas (RND), because these tracks are meant to experiment with specific features for which solvers may have (not) been specifically optimized. All of the above tracks feature formulas in prenex CNF, and the intersection between the test sets of these tracks is empty, i.e., there is no

	Track				
	MAIN	2QBF	SH	RND	NPNCNF
# Solvers	11	9	10	9	2
# Formulas	568	200	50	550	478

Table 1: QBFEVAL'10 at a glance. "# Solvers" and "# Formulas" denote the total amount of solvers and formulas involved in a given track, respectively.

formula tested in two different tracks. In the case of non-prenex non-CNF encodings (NPNCNF), since this is only the third time in which non-prenex non-CNF solvers are evaluated, and the research contributions in this area are currently still limited in number, we felt that the topic is not mature enough for a proper competition. We devote part of Section 2 to describe in some detail the five tracks, including solvers and formulas.

As in every event of this kind, it is important to choose carefully the test set(s) in order to get meaningful results while completing the evaluation in a reasonable time. In order to fulfil these two requirements, in the MAIN track we have extracted the final test set by sampling the pool of instances available to us – currently more than 15000 instances and 551 families – to extract a much smaller, yet representative, test set. In all the remaining tracks we chose the test set according to the topic of the track, and we tried to cover as much as possible the variety of the instances available to us, either by sampling (2QBF, SH) or by controlled generation (RND). In the NPNCNF track, we considered all the instances available to us. Finally, we ran the solvers on a farm of identical PCs, imposing different resource bounds according to the track. Section 2 is chiefly devoted to the description of the above selection methods and the computing infrastructure.

We present an analysis of QBFEVAL'10 data in two stages. In the first stage – Section 3 – we consider all the solvers and the instances in each track to give a rough, but complete, picture of the state of the art in QBF. By analyzing the results for problems and discrepancies among the solvers, we were able to isolate some solvers which turned out to be problematic, and we have removed them from subsequent analysis. In the second stage – Section 4 – we analyze the results with the aim of understanding the relative strengths and weaknesses of the various solvers, as well as the reasons why some formulas turn to be harder than others. Here we try to extract a narrow, but crisp, picture of the current state of the art. We also try to put things in perspective, by comparing the current results with past challenges [4] and envisioning possible future developments. For the MAIN track, we also provide the final ranking of the solvers according to the YASMv2 scoring method [19]. Finally, in Section 5 we conclude with an analysis of the evaluation and suggestions for future events of this kind.

2 Setup: solvers, instances and infrastructure

Table 2 summarizes the solvers submitted to QBFEVAL'10. The salient features of the participants are briefly described in the following. The input format description is assumed to be prenex CNF format, unless otherwise specified.

AIGSOLVE [22] uses And-Inverter Graphs (AIGs) as the main data structure, and AIG-based operations to reason about the input formula. The solver includes preliminary phases devoted to simplification, structure extraction and early quantification of the input formula.

AQME'10 [23] is a multi-engine solver, i.e., a tool using Machine Learning techniques to select among its reasoning engines the one which is more likely to yield optimal results. The reasoning engines of AQME'10 are a subset of those submitted to QBFEVAL'06, namely 2CLSQ, QUANTOR2.11,

Solver	Track	Author(s)
AIGSolve	main, sh	F. Pigorsch, C. Scholl
aqme'10	main, 2qbf, sh, rnd	L. Pulina, A. Tacchella
CirQit2.1	npncnf	A. Goultiaeva, V. Iverson, F. Bacchus
depqbf	main, 2qbf, sh, rnd	F. Lonsing
depqbf-pre	main, 2qbf, sh, rnd	F. Lonsing
nenofex	main, 2qbf, sh, rnd	F. Lonsing, A. Biere
qmaiga	main	S. Reimer, F. Pigorsch, M. Lewis, B. Becker, C. Scholl
qpro	npncnf	U. Egly, M. Seidl, S. Woltran
quantor3.1	main, 2qbf, sh, rnd	A. Biere
QuBE7	main, 2qbf, sh, rnd	E. Giunchiglia, P. Marin, M. Narizzano
QuBE7-c	main, 2qbf, sh, rnd	E. Giunchiglia, P. Marin, M. Narizzano
QuBE7-m	main, 2qbf, sh, rnd	E. Giunchiglia, P. Marin, M. Narizzano
StruQS'10	main, 2qbf, sh, rnd	L. Pulina, A. Tacchella

Table 2: The QBFEVAL'10 systems. The table is structured as follows. The first column ("Solver") reports the name of the solver, the second column ("Track") indicates the track in which the given solver is involved, while the last column ("Author(s)") reports solvers' authors.

QuBE3.1, sKizzo, and ssolve. Engine selection is performed according to the adaptive strategy described in [23].

CirQit2.1 [12] is a solver for non-prenex non-CNF formulas using a circuit-based data structure to represent and reason about the input formula. Search is performed on the internal representation using unit and Don't Care propagation. It performs term- and clause-learning, and it leverages a variable-state independent heuristic.

depqbf is a search-based solver leveraging compact dependency graphs to represent the prefix, instead of the standard linear representation – see [17]. The difference of depqbf-pre from the basic version is the use of quantor3.1 as a preprocessor.

nenofex [16] is an expansion-based solver which operates on negation normal form (NNF) formulas. NNF formulas are represented as structurally restricted trees, and expansions are scheduled based on expansion cost estimates.

qmaiga merges two "orthogonal" approaches. Its core is AIGSolve, but when AIGSolve is stuck in a sub-problem, the search-based solver QMiraXT [14] takes over the entire solution process.

qpro [8] is a search-based solver for non-prenex non-CNF formulas implementing dependency-directed backtracking.

quantor3.1 [5] is based on variable elimination and expansion, plus a number of features, such as equivalence reasoning, subsumption checking, pure literal detection, unit propagation, and also a scheduler for the elimination step.

QuBE7 is based on the composition of two reasoning tools: the preprocessor sQueezeBF [9], combining various techniques for reducing the size of the input QBF, and the search-based solver QuBE3.1 [11].

StruQS'10 [24] main feature is a dynamic combination of search – with solution- and conflict-backjumping – and variable-elimination. The key point in this approach is to implicitly leverage graph abstractions of QBFs to yield structural features which support an effective decision between search and variable elimination.

Further details about the solvers can be found in the short papers submitted by their authors and made available through the QBFEVAL'10 website [21]. Concerning the formulas, there was a single submission by Roman Kontchakov. The "Kontchakov" suite, as it is named in the following, consists of 136 formulas obtained by encoding minimal query inseparability module extraction in DL-Lite [13]. The formulas have been made available both with a prenex CNF encoding and their corresponding non-prenex non-CNF encoding. All the remaining formulas used in the evaluation have been extracted from QBFLIB [10]. These include the "Wintersteiger" suite, 372 encodings concerning ranking function synthesis problems [7] submitted after QBFEVAL'08 and thus not extensively tested so far. The complete test set used in QBFEVAL'10 can be downloaded from the QBFEVAL'10 website [21]. In the following, we describe in some detail the choice of the test sets for each of the five tracks. Tables 6 and 7 in Section 4 show further details about the selected formulas.

In order to construct a meaningful comparison between solvers, we constructed the test set for the MAIN track according to the following guidelines:

1. Balance the mix of different kinds of formulas, considering both syntactic features, e.g., number of variables and quantifier alternations, and structural features, e.g., density of the associated variable dependency graph.
2. Balance the empirical hardness of formulas so that the whole spectrum between easy and hard formulas is covered, where we consider QBFEVAL'08 results as a yardstick to assess formula hardness.
3. Balance between true and false formulas, again considering QBFEVAL'08 results.
4. Balanced the mix of formulas coming from different application domains, so that no application domain is neglected or overrepresented.
5. Finally, ensure that no more than 10% of the total test set comes from a single submitter whenever the formula submitter is also authoring a competing solver.

To satisfy requirement (1) above, we start with a pool of 2734 formulas extracted from QBFLIB, i.e., all the publicly available prenex CNF QBFs which have not been generated using some probabilistic model, excluding the ones eligible for 2QBF and SH tracks (see below). For each formula, we compute the same set of syntactic features used in AQME'10 [23] to automate engine selection, plus an approximation of quantified treewidth as described in [25]. Then, we extract a representative subset of the original pool by (*i*) clustering the formulas according to their features, and (*ii*) random sampling without repetitions the members of each cluster. The result of the above procedure is a pool of 432 formulas to which we add formulas from the "Kontchakov" suite to yield a final test set of 568 formulas. It turns out that such test set satisfies also requirements (2-5) above. In particular, the hardness mix turns out to be balanced – at least considering the formulas which were tested in QBFEVAL'08. If we consider requirement (3), we can see that the final selection consists of 166 and 170 formulas which are known to be true and false, respectively. Among the 232 formulas whose truth value has not been established, we have newly submitted formulas, hard formulas from QBFEVAL'08 (48), and formulas that were not used in the previous evaluation (48). As for requirement (4), we notice that the final test set is composed by 197 instances of formal verification problems, 101 instances of planning problems, 136 instances of module extraction problems and 134 instances of miscellaneous problems, so there is already some balance among application domains. In the following, for the sake of simplicity, we will consider module extraction problems as part of the miscellaneous category. Finally, also requirement (5) is implicitly satisfied by our selection of formulas.

2QBFs, i.e., formulas with a single $\forall\exists$ alternation in the prefix, arise frequently in applications such as conformant planning, formal property verification and, more in general, problems having Σ_2^p complexity. If we consider the pool of prenex CNF formulas that we used to extract the test set for the MAIN track, we see that about 40% of them are 2QBFs. Clearly, any general-purpose QBF solver can deal with 2QBFs, but they are also appealing for special-purpose approaches, e.g., cooperation of two state-of-the-art SAT solvers or encoding to other logics, like, e.g., Disjunctive Logic Programming. Moreover, it could be the case that even general-purpose solvers behave differently on 2QBFs with respect to the unbound-alternation case. Indeed, no formulas were submitted specifically for this track. Therefore, we selected 200 formulas from QBFLIB, trying to meet the same requirements mentioned previously for the MAIN track. Even if we followed the same procedure outline above, we did not manage to get an overall balance between the formulas. This is because two suites – "Basler" and "Wintersteiger", both made up by encoding of formal verification problems – are numerically dominating the category. However, we know from QBFEVAL'08 that most of the formulas in the suite "Basler" were solved by either 1 or 2 solvers, so they represent a reasonably difficult test set. The suite "Wintersteiger" is interesting in its own right, since it was submitted after QBFEVAL'08 and it has never been evaluated extensively.

Because solving QBFs is a hard combinatorial problem, we expect that while heuristic-based solvers can deal with QBFs of increasing "size", there will always be "small" instances that turn out to be extremely hard to solve in practice. The focus of this track is precisely on "small but hard" instances, i.e., relatively small QBFs that resisted solution attempts in previous QBF evaluations. Clearly, a key factor in selecting these formulas is deciding on a "size" parameter. The formulas composing the test set for the SH track have been picked up from QBFLIB according to the following procedure. We focus on formulas that no participant was able to solve in QBFEVAL'08. We rank such formulas in ascending order according to (*i*) number of variables, (*ii*) number of clauses and (*iii*) total number of literals. We consider only the first 100 entries in each ranking, and then each formula is scored using the Borda method [26]. The position of the formula in each of the rankings above is considered as a preference expressed by a voter, and the individual preferences are added up to yield the final score. The 50 highest ranking formulas are selected in the end.

As for the RND track, since no new formulas/generators were submitted specifically for this track, we composed the test set considering QBFEVAL'05 and '06 experience – the last two events in which were ran random formulas. In particular, we selected 55 instances with ten samples each, obtaining a total amount of 550 formulas. Our selection considers both generators, i.e., the "Chen-Interian" probabilistic model [6] and formulas, available in QBFLIB. Concerning formulas, we made a selection from NestedCounterFactual category, Miscellanea category (family "ASP_Program_Inclusion"), and Planning category (suite "Narizzano" and family "Strategic_Companies"). Finally, the test set of the NPNCNF is composed by 342 formulas already available in QBFLIB at the time of the submission, together with 132 newly submitted – the suite "Kontchakov" mentioned above.

In the MAIN track, the only competitive one, formulas have been preprocessed using a satisfiability-preserving shuffling of the variables in the prefix, literals in the clauses, and clauses in the matrix, respectively. For each track, the CPU time limit is set to 1200 seconds for all the tracks, except the SH track where we allowed 43200 seconds (12 hours) of runtime. To prevent memory swapping, we also set a memory limit at 2GB. If a solver exceeds the resource bounds while attempting to solve a formula, it is killed and the corresponding result is left undefined. In the MAIN track, we rank the solvers using the YASMv2 [19] in order to declare the winner. As shown in [19], YASMv2 is more robust than other common scoring methods, including simple criteria based on the number of problems solved and the CPU time spent solving them. The evaluation runs on a farm of 9 identical PCs locally available at the University of Genoa. The PCs are equipped with a processor Intel Core2Duo running at 2.13 GHz, with 4 GB of RAM, and running GNU Linux Debian 2.6.18.5.

Solver	MAIN		2QBF		SH		RND	
	#	Time	#	Time	#	Time	#	Time
AIGSolve	329	22786.60	NA	NA	**37**	1140.01	NA	NA
aqme'10	**434**	33346.60	128	2323.11	11	30132.40	**407**	20078.90
depqbf	370	21515.30	24	690.42	4	41448.00	342	12895.10
depqbf-pre	356	18995.90	51	877.02	4	33371.90	**343**	9438.62
nenofex	225	13786.90	50	3545.65	3	30194.20	149	34502.80
qmaiga	361	43058.10	NA	NA	NA	NA	NA	NA
quantor3.1	205	6711.37	48	3689.30	5	57960.90	134	2830.97
QuBE7*	**410**	52142.10	**173**	1981.88	9	80570.70	**359**	27092.90
QuBE7-c*	389	34926.80	**172**	3340.89	**16**	27207.10	339	29495.20
QuBE7-m*	**393**	40786.30	**171**	4109.16	**16**	21458.50	340	29393.30
StruQS'10	240	32839.70	132	1399.30	5	26257.30	117	15480.40

Table 3: First stage results for all prenex CNF solvers. For each track, we report the number of formulas solved within the time limit ("#") and the total CPU time ("Time") spent on the solved instances. Results in boldface are those of the best three solvers in each track – according to number of problems solved and total time only. NA means that the solver did not participate in a track. Solvers marked with an asterisk are those whose output gave rise to discrepancies.

3 Evaluation: first stage results

Table 3 presents the raw results of the evaluation, considering all the prenex CNF solvers and the four tracks where prenex CNF QBFs were tested. Looking at the results, we can see that in the MAIN track all the competitors were able to solve at least 33% of the test set. On the other hand, nenofex, quantor3.1 and StruQS'10 were not able to solve more than 50% of the instances. In terms of number of problems solved, the best solver is aqme'10, which can solve about 76% of the test set. This is interesting if we consider that aqme'10 combines solvers that used to be state-of-the-art four years ago. However, we can also see that some solvers that do not exploit a multi-engine paradigm like, e.g., QuBE7, depqbf and qmaiga are getting close to the performances of aqme'10. This indicates a clear progress in the field over the past four years. The performance of the solvers is also quite diverse: there are 229 instances – 40% of the test set – separating the strongest solver, from the weakest one.

Still looking at Table 3, in the track 2QBF we can see that QuBE7, QuBE7-m and QuBE7-c are able to solve more than 85% of the test set, while StruQS'10 and aqme'10 both solve slightly more than 60% of the test set. On the other hand, all the remaining solvers cannot do better than a mere 25%. This indicates that given the current state of the art in QBF reasoning, the performance demand of these encodings is still exceeding the capabilities of most solvers. As for the SH track, we see that AIGSolve was able to solve a considerable number of previously open problems, whereas all the remaining solvers did not perform particularly well. On one hand, this indicates that AIGSolve is particularly suited for this kind of problems – probably due to its internals which are fairly different from all the other solvers that participated in the track. On the other, it also shows a clear progress over previous QBFEVAL events. Looking at the results on random instances, we can see that all the solvers, with the only exception of StruQS'10 and quantor3.1, were able to conquer at least 25% of the instances in this category, and six solvers were able to conquer more than 50% of the instances. Only aqme'10 is able to solve 74% of the instances. Overall it seems that the choice of parameters for the generation of random instances yielded a performance demand within the capabilities of most solvers. Considering the contestants which solved more than 50% of the test set, but excluding the multi-engine solver, we see that the performance of the solvers is similar: only 19 instances separate the worst solver (QuBE7-m) from the best (QuBE7). Finally, if we consider the NPNCNF results – not shown in Table 3 – we have that CirQit2.1 solved 291 instances and

Solver	Points
DEPQBF	2896.68
DEPQBF-PRE	2508.96
AQME'10	2467.96
QMAIGA	2117.55

Solver	Points
AIGSOLVE	2037.22
QUANTOR3.1	1235.14
STRUQS'10	947.83
NENOFEX	829.11

Table 4: Final ranking of QBFEVAL'10. The first column ("Solver") reports the solvers participating in MAIN, while the second one ("Points") is filled with the points computed by using YASMv2.

QPRO solved 153 instances, i.e., 61% and 32% of the test set, respectively. Also if CIRQIT2.1 solves a noticeable amount of instances more than QPRO, it does not dominate it, because QPRO is able to solve 30 instances uniquely.

As we have anticipated in Section 1, a few discrepancies in the results of the solvers were detected during the analysis of the first stage results. A total of 3 discrepancies were detected in the MAIN track, and 4 discrepancies in the SH track. For each of the discrepancies we reran the solvers reporting a result different from the majority of the other solvers and/or the expected result of the instance. We also inspected the instances, looking for weird syntax and other pitfalls that may lead a correct solver to report an incorrect result. In particular, the instances which gave rise to discrepancies belong to three different suites: "Letombe", "Katz" and "Mneimneh-Sakallah". We ran on such instances state-of-the-art QBF certifiers in order to get a definite answer on them. The only instance for which we obtained a push-button result is `par8-4-90` (suite "Letombe"), certified by QBD [28] as FALSE. By performing some manually-assisted reasoning, we found out that `test3_quant2` is FALSE as well. At the end of this analysis we excluded from the second stage the following solvers: QUBE7, QUBE7-M and QUBE7-C, responsible for all discrepancies detected. Clearly, for instances that were conquered by just one solver, and for which we do not know the satisfiability status in advance, the possibility that the solver is wrong still exists, but we consider this as unavoidable given the current state of the art.[1]

4 Evaluation: second stage results

Table 4 shows the results obtained computing YASMv2 on the competitive track. Looking at the table, we declare DEPQBF as the winner of QBFEVAL'10, followed by DEPQBF-PRE, and AQME'10. Notice that, even if AQME'10 is able to solve more problems that both DEPQBF and DEPQBF-PRE, the setup time of the engine-selection strategy may considerably increase the runtime in most short-to-solve instances, and YASMv2 is designed to penalize this behavior. In the remainder of the section, we look in detail at each track, considering both solver- and instance-centric views. When we say that "solver A *dominates* solver B" we mean that the set of problems solved by B is a subset of those solved by A.

4.1 Solver-centric view

Table 5 shows the results of the MAIN track dividing the formulas into three categories. As we can see, in terms of number of problems solved, AQME'10 is the strongest solver: it leads the count in Formal Verification and Planning, and it is second best in Miscellanea category.

[1]Notice that the same problem exists in the SAT competition when a solver is the only one to report about an instance, the answer is "unsatisfiable", and the solver cannot produce a checkable proof.

Category	Solver	Total		True		False		Unique	
		#	Time	#	Time	#	Time	#	Time
Formal Verification (197)	AQME'10	134	9056.00	49	3938.94	85	5117.06	3	639.97
	QMAIGA	130	10611.28	55	5374.21	75	5237.07	–	–
	AIGSolve	123	7129.58	56	4189.91	67	2939.67	2	1006.30
	StruQS'10	92	16501.98	39	8722.64	53	7779.34	6	2205.45
	DEPQBF	89	4109.93	17	959.61	72	3150.31	5	505.70
	QUANTOR3.1	73	4236.95	31	3096.88	42	1140.07	–	–
	NENOFEX	70	7928.03	27	4744.43	43	3183.60	3	350.89
	DEPQBF-PRE	65	4195.32	20	1032.27	45	3163.05	–	–
Miscellanea (270)	DEPQBF-PRE	222	12179.63	117	10735.33	105	1444.31	–	–
	AQME'10	219	20210.17	95	11698.52	124	8511.64	1	5.22
	DEPQBF	208	14063.50	110	12234.42	98	1829.08	–	–
	QMAIGA	162	27280.91	89	13087.86	73	14193.05	1	859.00
	AIGSolve	145	13569.84	81	6386.08	64	7183.76	–	–
	StruQS'10	90	11597.60	39	3751.63	51	7845.97	–	–
	NENOFEX	88	4786.58	45	2985.23	43	1801.35	–	–
	QUANTOR3.1	62	1087.27	31	678.05	31	409.22	–	–
Planning (101)	AQME'10	81	2824.92	40	188.10	41	2636.81	2	892.65
	DEPQBF	73	3341.86	37	577.74	36	2764.12	1	434.81
	QUANTOR3.1	70	1387.15	38	355.70	32	1031.46	1	585.96
	DEPQBF-PRE	69	2620.92	35	686.16	34	1934.76	–	–
	QMAIGA	69	5165.87	36	2234.56	33	2931.31	–	–
	NENOFEX	67	1072.33	37	512.21	30	560.12	–	–
	AIGSolve	61	2087.13	34	1515.49	27	571.64	–	–
	StruQS'10	58	4740.17	31	1331.23	27	3408.94	–	–

Table 5: MAIN track second stage results. "Category" reports the application domain, and for each solver, the table shows the number of instances solved ("#") and the total CPU time spent to solve them ("Time"). Total number of formulas solved ("Total") is also split into true, false, and uniquely solved formulas ("True", "False" and "Unique", respectively). A dash means that a solver did not solve any instance in the related group. Solvers are sorted according to the number of instances solved, and, in case of a tie, according to CPU time.

Overall, the strongest solver in Formal Verification category is able to solve 68% of the total number of instances, while DEPQBF-PRE is able to solve about 82% of the Miscellanea category, and AQME'10 solves 80% of instances in Planning category. Focusing on Formal Verification category, 68 instances separate the strongest solver from the weakest one. If we consider the problems that are uniquely solved, and the five best solvers, then we see that no solver is dominated by the others, with the noticeable exception of QMAIGA (dominated by AQME'10). Finally, we report that DEPQBF yields the smallest average CPU time (about 46s). In the Miscellanea category, the first thing to be observed is that the strongest solver is DEPQBF-PRE, which ranks last in the Formal Verification category. The first three solvers are pretty much in the same capability ballpark: only 14 instances separate the first one from the third one. Finally, considering the Planning category, we can see that only 23 instances separate the strongest solver from the weakest one. We also report that QUANTOR3.1 seems to perform better in the Planning category w.r.t. the other ones, because in this category we find it in the group of the best three solvers.

Considering now the 2QBF[2] track, we have that the test set is mostly composed by Formal Verification formulas (177 out of 200). StruQS'10 turns out to be the strongest solver in this category in the 2QBF track, with 126 instances solved (71% of the test set). AQME'10 trails with 111 instances solved, and NENOFEX is third best with 44 instances solved. StruQS'10 is very close to dominate all remaining solvers: it uniquely solves 20 formulas, but both AQME'10

[2]For the lack of space, we do not show tables about the non-competitive tracks, and we made them available on-line at [21].

and NENOFEX are also able to uniquely solve 1 formula. If we compare this result with the main track, where AQME'10, QMAIGA and AIGSOLVE are stronger than STRUQS'10, then we may conjecture that this is due to STRUQS'10 becoming less effective when the number of alternations increases. As for the remaining formulas in the 2QBF track, we have that STRUQS'10, QUANTOR3.1, AQME'10, and NENOFEX were able to solve all the 6 instances in the "Miscellanea" category, whereas DEPQBF-PRE is able to solve all 17 formulas in the "Planning" category, and it also dominates all the other solvers. In particular, QUANTOR3.1, NENOFEX, and STRUQS'10 were not able to solve any formula in this category. These formulas are all `sortnetsortXX.AE` (Family "Sorting Network", suite "Rintanen"), for which is confirmed their hardness for non-DPLL based solver, as we can see from results of previous QBFEVALs. Also in the SH track Formal Verification formulas are prevalent (42 out of 50). The main result of this track is that AIGSOLVE is able to globally solve 37 open problems of QBFEVAL'08. In particular, 35 out of 37 fall in the Formal Verification category, mainly belonging in "Biere" and "Katz" suite.

Looking at the RND track, and focusing on the Chen-Interian family, we report that AQME'10 is the strongest solver, and it is able to solve about 60% of the test set. It also dominates all remaining solvers. It is followed by a group of three solvers, namely DEPQBF-PRE, DEPQBF, and STRUQS'10, which are able to solve the same set of 100 formulas. The remaining solvers are not able to deal with more than 25% of the test set. There are also 61 formulas uniquely solved by AQME'10 (by using SSOLVE), which are the ones supposedly close to the phase transition in the random generation model. No other solver is able to return a solution in this range of formulas. Concerning other random formulas, we report the following.

- In the family "ASP_Program_Inclusion", DEPQBF-PRE, AQME'10 (mainly using QUBE3.1), and DEPQBFare able to solve 100% of the test set, while the remaining three are not able to solve any formula.

- In the family "NestedCounterFactual" the strongest solver is DEPQBF-PRE, able to cope with all the test set. The other solvers able to solve at least 50% of the test set are DEPQBF, which solves 79 out of 80 instances, and AQME'10, with 58 out of 80 solved instances.

- Finally, regarding the Planning category, the picture is very close to the one described for the Chen-Interian category. The strongest solver is AQME'10, coping with 92% of the dataset, followed by DEPQBF-PRE and DEPQBF, having solved both 77% of the dataset. In this case, AQME'10 also dominates all the remaining solvers. The noticeable difference with "Chen-Interian" family is that now STRUQS'10 ranks last, solving only 15 formulas.

Concluding, in the NPNCNF track there are two categories of formulas, namely "Formal Verification", composed by 342 out of 478 instances, and "Miscellanea", composed by the suite Kontchakov. For both categories, the strongest solver is CIRQIT2.1. Concerning "Formal Verification", CIRQIT2.1 solves 205 formulas, against 72 solved by QPRO, which is also dominated by CIRQIT2.1. The picture slightly changes in the case of "Miscellanea" because, even if CIRQIT2.1 is still the strongest solver (86 solved instances), being the gap with QPRO of only 5 formulas. Finally, for the Miscellanea category we also report that CIRQIT2.1 and QPRO uniquely solve 35 and 30 formulas, respectively. Therefore, no solver dominates the other in this category.

Family	Overall		Time	Hardness		
	N	#		EA	ME	MH
Abduction	52	50	25.19	14	35	1
Adder	15	15	568.79	–	12	3
blackbox-01X-QBF	59	53	807.66	8	39	6
blackbox_design	2	2	82.35	–	1	1
Blocks	5	5	16.05	1	4	–
BMC	18	17	64.16	7	10	–
C432	4	4	0.52	1	3	–
C499	2	2	0.90	–	2	–
C5315	7	3	3.80	1	2	–
C6288	4	2	21.09	–	1	1
C880	1	1	0.20	–	1	–
Chain	1	1	0.02	–	1	–
circuits	3	3	18.00	1	2	–
comp	2	2	0.03	2	–	–
conformant_planning	15	10	1042.32	2	7	1
Connect4	11	9	125.77	6	3	–
Counter	4	4	937.36	–	3	1
Debug	5	4	462.59	–	4	–
evader-pursuer-4x4-logarithmic	3	2	2.86	–	2	–
evader-pursuer-4x4-standard	7	1	5.75	–	1	–
evader-pursuer-6x6-logarithmic	4	3	721.25	–	2	1
evader-pursuer-6x6-standard	2	2	892.65	–	–	2
evader-pursuer-8x8-logarithmic	6	4	40.12	–	4	–
FPGA_PLB_FIT_FAST	2	2	0.05	1	1	–
FPGA_PLB_FIT_SLOW	1	1	1.61	–	1	–
Impl	1	1	0.00	1	–	–
jmc_quant	2	2	1.15	–	2	–
jmc_quant_squaring	1	1	8.38	–	1	–
k_branch_n	4	4	870.06	–	3	1
k_branch_p	7	7	380.20	–	7	–
k_d4_n	10	10	15.63	–	10	–
k_d4_p	5	5	1.63	1	4	–
k_dum_n	2	2	0.04	–	2	–
k_dum_p	4	4	0.74	–	4	–
k_grz_n	4	4	4.86	2	2	–
k_grz_p	3	3	3.16	1	2	–
k_lin_n	5	5	283.95	–	5	–
k_lin_p	4	4	0.40	2	2	–
k_path_n	3	3	0.12	–	3	–
k_path_p	4	4	0.14	–	4	–
k_ph_n	6	6	7.86	6	–	–
k_ph_p	4	2	6.11	2	–	–
k_poly_n	4	4	0.09	–	4	–
k_poly_p	2	2	0.06	–	2	–
k_t4p_n	4	4	5.62	–	4	–
k_t4p_p	5	5	3.01	–	5	–
Logn	1	1	1.12	–	1	–
mqm	136	136	7953.95	–	136	–
s1196	1	–	–	–	–	–
s1269	1	–	–	–	–	–
s27	1	1	0.02	–	1	–
s298	4	4	158.09	–	4	–
s3330	2	–	–	–	–	–
s386	1	1	715.71	–	1	–
s499	2	2	160.23	–	2	–
s510	3	2	403.96	–	2	–
s713	2	1	60.28	–	1	–
s820	2	–	–	–	–	–
Sorting_networks	6	6	51.65	2	4	–
SzymanskiP	2	2	527.78	–	1	1
term1	3	3	0.34	1	2	–
tipdiam	14	14	102.15	1	13	–
tipfixpoint	24	22	2282.68	4	12	6
Toilet	4	4	0.49	3	1	–
ToiletA	10	10	0.72	7	3	–
ToiletC	23	23	1.32	22	1	–
ToiletG	4	4	0.01	4	–	–
VonNeumann	2	2	5.92	1	1	–
z4ml	1	1	0.00	1	–	–

Table 6: Classification of MAIN formulas considering second stage data. The table consists of seven columns where for each family of instances we report the name of the family in alphabetical order (column "Family"), the number of instances included in the family, and the number of instances solved (group "Overall", columns "N", "#", respectively), the CPU time taken to solve the instances (column "Time"), the number of easy, medium and medium-hard instances (group "Hardness", columns "EA", "ME", "MH").

4.2 Instance-centric view

In Table 6 we show the classification of formulas included in MAIN according to the solvers admitted to the second phase. In the table, the number of instances solved and the cumulative time taken for each family is computed considering the "SOTA solver", i.e., the ideal solver that always fares the best time among all the participants. An instance is thus solved if at least one of the solvers solves it, and the time taken is the best among the times of the solvers that solved the instances. The instances are classified according to their hardness with the following criteria: easy instances are those solved by all the solvers, medium instances are those non-easy instances that could still be solved by at least two solvers, medium-hard instances are those solved by one reasoner only, and hard instances are those that remained unsolved.

According to the data summarized in Table 6, MAIN consisted of 568 instances, 523 of which have been solved, resulting in 105 easy, 393 medium, 25 medium-hard, and 45 hard instances. These results indicates that the selected instances are not trivial for current state-of-the-art

Track	Family	Overall		Time	Hardness			Family	Overall		Time	Hardness		
		N	#		EA	ME	MH		N	#		EA	ME	MH
	irqlkeapclte	7	1	0.46	–	1	–	Sorting_networks	17	17	107.19	–	11	6
2QBF	MutexP	7	7	5.08	3	4	–	terminator	57	51	427.64	–	38	13
	Qshifter	6	6	2.51	2	4	–	wmiforward	46	27	159.21	11	15	1
	RankingFunctions	60	55	72.50	–	47	8							
	Abduction	2	2	142.85	–	2	–	k_branch_n	2	2	9.50	–	2	–
	C499	3	3	17.92	–	–	3	Sorting_networks	4	3	5133.28	–	2	1
	C880	6	4	54.19	–	–	4	tipdiam	1	1	0.44	–	–	1
SH	circuits	1	1	187.02	–	1	–	tipfixpoint	13	9	102.25	–	4	5
	Counter	4	4	429.15	–	1	3	uclid	2	1	12.09	–	–	1
	jmc_quant	5	5	6.61	–	–	5	wmiforward	2	2	0.67	–	–	2
	jmc_quant_squaring	5	5	32.44	–	–	5							
	ASP_Program_Inclusion	40	40	47.99	–	40	–	RobotsD3	30	29	259.31	–	29	–
RND	CounterFactual	80	80	205.06	2	77	1	RobotsD4	30	30	202.22	5	22	3
	q2k3k3	270	161	12317.15	30	70	61	RobotsD5	30	30	290	7	23	–
	RobotsD2	30	29	1292.02	–	26	3	Strategic_Companies	40	30	1300.85	3	12	15
NPNCNF	mqm	136	116	14943.18	51	–	65	QLTL_safety	250	116	38217.74	6	–	110
	NuSMV_diam	92	89	385.37	66	–	23							

Table 7: Classification of formulas related to 2QBF, SH, RND, and NPNCNF tracks, considering second stage data. The table is organized as Table 6, with the only difference that in the leftmost column ("Track") is reported the related track.

QBF solvers since there is about 18% of easy instances. At the same time, the test set is not overwhelming, since most of the non-easy instances (about 69%) are solved by at least two solvers. Some "old" instances are still pretty hard for current state-of-the-art solvers, like the ones coming from the Mneimneh-Sakallah suite (families s641, s1196, s1269, and s3330). As a final consideration, we report that the main contributors to the SOTA solver, in percentage, were depqbf, depqbf-pre, and quantor3.1, with 28%, 18%, and 15%, respectively. This is the main explanation of the differences in the ranking positions between Table 3 and Table 4.

Concerning the 2QBF track, looking at the Table 7, we can see that, out of 200 instances, 164 have been solved, resulting in 16 easy, 120 medium, 28 medium-hard, and 36 hard instances. In particular, we report that the families in the suite "Basler" are the ones with the highest number of open formulas, while 92% of the formulas in the new family "RankingFunctions" (suite "Wintersteiger") are solved. Notice that 100% of the formulas in "legacy" families, i.e., "MutexP", "Qshifter", and "Sorting_networks" are solved by the SOTA solver. Finally, the three main contributors to the SOTA solver are StruQS'10, aqme'10 and depqbf-pre with 54%, 21%, and 20%, respectively. As regards the SH track, we can see in Table 7 that, out of 50 instances, 42 have been solved, resulting in 12 medium, 30 medium-hard, and 8 hard instances. These 8 open instances are mostly located in the "tipfixpoint" family (4). Notice that 1 hard instance is in the "uclid" family, that was submitted to the first QBFEVAL in 2003. The SOTA solver is mainly composed by AIGSolve, while other contributions do not exceed 5%. Looking now at the classification of formulas included in RND, in the related section of Table 7 we can see that, out of 550 instances, 429 have been solved, resulting in 47 easy, 299 medium, 83 medium-hard, and 121 hard instances. We report that the main contributors to the SOTA solver are aqme'10 and depqbf-pre, with 42% for each one. Finally, concerning NPNCNF track, in Table 7 we can see that, out of 478 instances, 321 have been solved, resulting in 123 easy, 198 solved by only one solver, and 157 hard instances. The contribution of CirQit2.1 and qpro to the SOTA solver is 64% and 36%, respectively. Noticeably, there has been a leap forward in solving the "QLTL_safety". In QBFEVAL'08, only 6 instances were solved[3], while we now report 116 solved formulas with "only" a factor 2× in the CPU time limit. Another interesting observation is about "mqm" formulas. If we compare the total amount of solved formulas with the one reported in Table 6, we can see that, even if the results achieved using prenex CNF solvers are better than those achieved with non-prenex non-CNF solvers, the two values are comparable.

5 Conclusions

The final balance of QBFEVAL'10 can be summarized as follows:

- 13 solvers participated, 11 requiring QBFs in prenex CNF format.
- 136 formulas obtained by encoding minimal query inseparability module extraction in DL-Lite were submitted, both with an encoding in prenex CNF and their corresponding non-prenex non-CNF.
- State-of-the-art solvers for each track have been identified; also, a total of 367 challenging hard instances have been identified to set the reference point for future developments in the field.

All the information contained in this paper can be retrieved at the QBFEVAL'10 web portal [21]. Despite some long-standing limitations it is our opinion that the development of QBF solvers has reached its maturity, in the sense that QBF solvers are now dependable and effective tools which have a concrete potential for applications. The main question to be addressed by future research is whether the current state-of-the-art solvers, alone or in combinations among them, can solve industrial-sized and practically relevant problems. While QBF-based automation techniques can be regarded as a promising research direction, the lack of applications may discourage further developments, so it is important for researchers in the field to come out with "killer applications" wherein QBF solvers have an edge over competing technologies, e.g., SAT solvers or BDDs.

Acknowledgments

The authors would like to thank all the participants to the QBF competition and evaluations for submitting their work. The authors also wish to thank Uwe Bubeck, Oliver Kullmann, and Inês Lynce for their effort as judges of QBFEVAL'10.

References

[1] J. Argelich, C. Min Li, F. Manya, and J. Planes. Fourth Max-SAT Evaluation, 2009. `www.maxsat.udl.cat/09`.

[2] C. Barrett, M. Deters, A. Oliveras, and A. Stump. Design and results of the 4th annual satisfiability modulo theories competition (SMT-COMP 2008). *To appear*, 2008.

[3] D. Le Berre, O. Roussel, and L. Simon. The SAT2009 Competition, 2009. `www.satcompetition.org/2009`.

[3]We are not tacking into account PQBF, reported as problematic solver in QBFEVAL'08.

[4] D. Le Berre, L. Simon, and A. Tacchella. Challenges in the QBF arena: the SAT'03 evaluation of QBF solvers. In *Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT 2003)*, volume 2919 of *Lecture Notes in Computer Science*, pages 468–485. Springer Verlag, 2003.

[5] A. Biere. Resolve and Expand. In *Seventh Intl. Conference on Theory and Applications of Satisfiability Testing (SAT'04)*, volume 3542 of *LNCS*, pages 59–70. Springer Verlag, 2005.

[6] H. Chen and Y. Interian. A Model for Generating Random Quantified Boolean Formulas. In *Proceedings of 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 66–71, 2005.

[7] B. Cook, D. Kroening, P. Rümmer, and C. M. Wintersteiger. Ranking function synthesis for bit-vector relations. In *Proceedings of TACAS 2010*. Springer, 2010. To appear.

[8] U. Egly, M. Seidl, and S. Woltran. A solver for QBFs in negation normal form. *Constraints*, 14(1):38–79, 2009.

[9] E. Giunchiglia, P. Marin, and M. Narizzano. Preprocessing Techniques for QBFs. In *15th RCRA workshop*, 2008.

[10] E. Giunchiglia, M. Narizzano, L. Pulina, and A. Tacchella. Quantified Boolean Formulas satisfiability library (QBFLIB), 2001. `www.qbflib.org`.

[11] E. Giunchiglia, M. Narizzano, and A. Tacchella. Clause-Term Resolution and Learning in Quantified Boolean Logic Satisfiability. *Artificial Intelligence Research*, 26:371–416, 2006.

[12] A. Goultiaeva, V. Iverson, and F. Bacchus. Beyond CNF: A Circuit-Based QBF Solver. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing*, pages 412–426. Springer-Verlag, 2009.

[13] R. Kontchakov, L. Pulina, U. Sattler, T. Schneider, P. Selmer, F. Wolter, and M. Zakharyaschev. Minimal Module Extraction from DL-Lite Ontologies using QBF Solvers. In *21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 836–841, 2009.

[14] M. Lewis, T. Schubert, and B. Becker. QMiraXT–A Multithreaded QBF Solver. *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*, 2009.

[15] D. Long and M. Fox. The 3rd International Planning Competition: Results and Analysis. *Artificial Intelligence Research*, 20:1–59, 2003.

[16] F. Lonsing and A. Biere. Nenofex: Expanding nnf for qbf solving. In *11th International Conference on Theory and Applications of Satisfiability Testing*, volume 4996 of *Lecture Notes in Computer Science*, pages 196–210. Springer, 2008.

[17] F. Lonsing and A. Biere. Efficiently Representing Existential Dependency Sets for Expansion-based QBF Solvers. *Electronic Notes in Theoretical Computer Science*, 251:83–95, 2009.

[18] V. Manquinho and O. Roussel. Pseudo-Boolean Competition 2009, 2009. `www.cril.univ-artois.fr/PB09`.

[19] M. Narizzano, L. Pulina, and A. Tacchella. Ranking and Reputation Sytems in the QBF competition. In *10th Conference of the Italian Association for Artificial Intelligence (AI*IA 2007)*, volume 4733 of *Lecture Notes in Artificial Intelligence*, pages 97–108. Springer Verlag, 2007.

[20] C. Peschiera, L. Pulina, and A. Tacchella. Sixth QBF solvers evaluation (QBFEVAL), 2008. `http://www.qbfeval.org/2008`.

[21] C. Peschiera, L. Pulina, and A. Tacchella. Seventh QBF solvers evaluation (QBFEVAL), 2010. `http://www.qbfeval.org/2010`.

[22] Florian Pigorsch and Christoph Scholl. Exploiting structure in an aig based qbf solver. In *Design, Automation and Test in Europe (DATE 2009)*, pages 1596–1601. IEEE, 2009.

[23] L. Pulina and A. Tacchella. A self-adaptive multi-engine solver for quantified Boolean formulas. *Constraints*, 14(1):80–116, 2009.

[24] L. Pulina and A. Tacchella. A structural approach to reasoning with quantified Boolean formulas. In *21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 596–602,

2009.

[25] L. Pulina and A. Tacchella. An empirical study of QBF encodings: from treewidth estimation to useful preprocessing. *Journal of Algorithms in Cognition, Informatics and Logic*, 2009. To appear.

[26] D. G. Saari. *Chaotic Elections! A Mathematician Looks at Voting.* American Mathematical Society, 2001.

[27] C. Sinz. Sat-race 2008, 2008. `baldur.iti.uka.de/sat-race-2008`.

[28] Y. Yu and S. Malik. Verifying the correctness of quantified boolean formula(qbf) solvers: Theory and practice. In *Proceedings of the 2005 Conference on Asia South Pacific Design Automation ASP-DAC*, pages 1047–1051, 2005.