Better Evaluations by Analyzing Benchmark Structure

Norbert Manthey and Sibylle Möhle

Knowledge Representation and Reasoning Group
Technische Universität Dresden
Dresden, Germany

Abstract

We present a method for improving the efficiency of SAT solver benchmarking. The increase in efficiency is achieved by removing redundant formulae from the benchmark for the evaluation and then mapping back the results on the entire benchmark. Experiments confirm the accurateness of our method along with a computation time reduction. The redundancy contained in the benchmark causes a bias in the evaluation result as well and therefore has an impact on its significance, particularly if multiple benchmarks are combined. We illustrate this effect by an example and pinpoint the structure of the benchmarks used for the competitive events performed since 2002. We focus on SAT, however, the presented results are of interest for other competitions as well.

1 Introduction

Since 2002, annual competitions are organized and induced SAT researchers to submit their tools. The publicly available benchmarks of these events have been used to evaluate additions to the basic conflict driven clause learning (CDCL) algorithm. This evaluation requires computationally expensive experiments. However, not every research group has an appropriate hardware at its disposition and accessing publicly available resources may not be an option. The need for increasing the efficiency of solver evaluation while preserving the expressiveness of the results is therefore evident.

The interpretation of the evaluation outcome is problematic if duplicates of a formula are present notably in the combination of multiple benchmarks. Consider the following example: Two benchmarks A and B consist of 300 formulae each, with 100 formulae occurring in both benchmarks. Hence, in our example, the combined benchmark contains 100 duplicates. We denote the set of the remaining 500 formulae by *set of unique formulae* and the formulae contained in it simply by *uniques*, since they are contained only once in this set. Imagine a solver X solving 200 formulae contained in A and 200 formulae contained in B with 50 of these formulae occurring in both A and B. In total, X solved 400 formulae, and one might be inclined to conclude that when executed on other benchmarks, X will solve $400/600 = 66.67\,\%$ of all formulae as well. Actually, only 350 of them are uniques, and hence, X solved $350/500 = 70\,\%$ of the unique formulae, since 50 of the solved formulae are duplicates. Note that if a formula is solved, its duplicate will also be solved. This fact has an impact on the assessment of the evaluation result. We illustrate it by further elaborating our example. Let us now imagine that a novel extension of X is tested on the same data and, again, 200 formulae contained in A and 200 formulae contained in B are solved, while, this time, 100 of these formulae are duplicates. As in the first experiment, $66.67\,\%$ of all formulae are solved, but the percentage of solved unique formulae amounts to $300/500 = 60\,\%$, which is $10\,\%$ less than before. The tested solver extension evidently exploits the presence of duplicates, i.e., the *redundancy*, in the benchmark, but this property goes unnoticed if the benchmark structure is ignored. For this reason, it is not possible to predict how X will perform on a different combination of benchmarks C, unless the percentage of duplicates in C is comparable to that of the combination of A and B.

Knowledge concerning the structure of the benchmark – notably the redundancy contained in it – is therefore crucial. Furthermore, it is very hard to reason about an experimental evaluation on two benchmarks if the details on the intersection are not known. In this case, only one of the two benchmarks can be used for reasoning.

Broadening the notion of duplicates introduced in our example, *feature-equivalence* of formulae is defined as the equality of formulae with respect to their features. Based on feature-equivalence, we introduce the concept of *redundancy* in a benchmark, which in our example corresponds to the multiset of duplicates. These notions facilitate the investigation of the structure of a combination of benchmarks and thus allow for an unbiased assessment of the evaluation results. We analyze the structure of the benchmarks used in competitive SAT events so far and demonstrate its impact on the evaluation results.

Despite the need for increasing the efficiency of solver evaluation, to the author's best knowledge, no ongoing research exists in this field. Experiments presented in publications the authors are aware of are executed on a full benchmark, and in the interpretation of the results the structure of the benchmark is not considered. Usually, results are presented on a per year basis. We show how solver evaluation can be sped up by exploiting the redundancy present in the combination of benchmarks. Suitable tools as well as the features calculated for all formulae are provided. Experiments performed using Glucose 3.0[1], Lingeling baq[2], and Riss 6[3] show the feasibility of our approach.

In this work, we are focussing on SAT. However, adapting our results to other problems, e.g., QBF, MaxSAT, PB, CSP, and AIG, is straightforward. Therefore, the presented findings are relevant for the evaluation of solvers competing in these related fields as well.

The paper is structured as follows: After recalling some preliminary terms, presenting background information and introducing an adequate terminology, in Section 3 we analyze the benchmarks of all competitive events between 2002 and 2015. In Section 4, we present a method for performing solver evaluations on a reduced set of formulae by exploiting redundancy in the benchmarks. An experimental evaluation is reported in Section 5, before in Section 6 we conclude and point out future work in Section 7.

2 Preliminaries

2.1 The Satisfiability Problem

Let the set of Boolean variables $\mathcal{V}$ be the set of the positive natural numbers, i.e., $\mathcal{V} = \mathbb{N}^+$. A *literal* v is a positive Boolean variable v or a negated Boolean variable $\overline{v}$. A *clause* C is a set of literals, and a *formula* F in conjunctive normal form (CNF) is a multiset of clauses, since it can contain duplicate clauses.

We assume the reader to be familiar with the basic concepts of propositional logic and SAT solving. More details about the used concepts can be found in [5].

2.2 Benchmarks and CNF Features

The formulae on which we evaluated our approach were downloaded from `satcompetition.org`, choosing only packages that have been used in competitions.

[1]http://baldur.iti.kit.edu/sat-race-2015/descriptions/glucose-satrace15.pdf
[2]http://baldur.iti.kit.edu/sat-race-2015/descriptions/fmv-solvers-sat-race-2015.pdf
[3]Submitted to the SAT Competition 2016

For each CNF formula, various numerical features were computed using *classifier*, a tool which is built by a call to Riss 6 with the appropriate parameters. *classifier* first performs unit propagation and then removes all satisfied clauses and falsified literals from the formula, similarly to a simple preprocessor of a SAT solver. On the resulting formula, the features were computed. In this work, we use 196 features based on the clause size distribution, the symmetry breaking features, the binary implication graph, the recursive weight heuristics and the constraint extraction[4]. For details on the features used in our work see [1].

All features mentioned above are *rotation-invariant*, which makes them immune to the renaming of variables and reordering of clauses in formulae. However, the data in Table 1 confirm that the precision of the generated features is accurate enough to discriminate between formulae.

2.3 Redundancy in Benchmarks

In our approach, comparing formulae is essential. To this end, we introduce the notion of *feature-equivalence*. We call two formulae F and G *feature-equivalent* if the values of all their features are equal. A *duplicate* of a formula F in this context denotes a formula which is feature-equivalent to F. Duplicates account for the *redundancy* in a benchmark. We call a benchmark *redundancy-free* if it contains just a single representative formula of each equivalence class in the partitioning by feature-equivalence. A *redundancy-free extract* of a benchmark is a redundancy-free benchmark obtained from a given benchmark by picking a representative from each of its equivalence classes in the partitioning by feature-equivalence. We denote these representatives by *unique formulae* or simply by *uniques*.

3 Analyzing Benchmarks of Competitive Events

Since 2002, annual competitions are organized with the aim to evaluate SAT solvers on various sets of benchmarks. These competitions run tracks for application formulae as well as for hard combinatorial (also called handmade or crafted) problems. Each year, the organizers of the competition issue a call for novel formulae for these tracks and generate a benchmark based on some criteria, e.g., [2, 3].

3.1 Distribution of Formulae Among Benchmarks

Most competition benchmarks contain novel formulae as well as problems already included in earlier benchmarks. We examined the composition of the benchmarks for the *application* track since 2002. Our main objective was to determine the portion of novel formulae on the one hand and to identify redundancy on the other hand. For determining the number of feature-equivalent formulae, the order of the clauses was retained fixed, and the literals in a clause were sorted – as this procedure is performed in most SAT solvers. Unit propagation was performed in the order of the occurrence of units in the formula and with respect to the introduction of new units during propagation. Finally, falsified literals were removed from the formula. The results are visualized in Table 1.

As an example, the 2004 competition benchmark (shown in row **04**) comprises 18 formulae introduced in 2002 (column **02**), 8 formulae submitted for the first time in 2003 (column **03**), and 298 new problems (column **04**). In total, the benchmark contains 324 formulae (Σ), 91.97% of which are new (`N`), and there are 315 uniques (`U`) and 9 duplicates (`D`). 98 formulae introduced

[4]The exact call is *./classifier -no-clausesf -no-resf -no-xor -no-varf -no-derivativef -big -rwhf -const.*

Table 1: Every year's benchmark (rows **02** to **15**) consists of novel formulae as well as formulae already contained in earlier competition benchmarks (columns **02** to **15**). Since 2008, the portion of novel formulae (column `N`) amounts to approximately 50 %. The number of uniques and duplicates is shown in columns `U` and `D`, respectively. Row `R` indicates the number of formulae used in subsequent benchmarks.

	02	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	$\sum$	N	U	D
02	829														829	100 %	827	2
03	0	119													119	100 %	119	0
04	18	8	298												324	91.97 %	315	9
05	0	0	0	176											176	100 %	175	1
06	5	5	14	5	71										100	71 %	100	0
07	0	0	0	15	20	140									175	80 %	175	0
08	7	7	30	10	26	11	109								200	54.5 %	187	13
09	4	11	12	9	28	54	13	161							292	55.1 %	292	0
10	0	1	0	3	7	17	6	23	43						100	43 %	100	0
11	12	16	7	10	7	19	11	28	11	179					300	59.67 %	299	1
12	28	28	35	19	31	69	28	82	9	89	182				600	30.33 %	558	42
13	0	9	0	8	10	18	3	33	3	32	26	158			300	52.67 %	300	0
14	1	14	0	4	3	14	9	38	4	29	20	12	152		300	50.67 %	299	1
15	1	2	0	3	0	9	5	19	1	13	8	3	69	167	300	55.67 %	291	9
R	76	101	98	86	132	211	75	223	28	163	54	15	69					

in 2004 are reused in subsequent competition benchmarks (row `R`). Note that in this table we do not discriminate between single and multiple reuse of formulae.

From 2008 on, the novel formulae (column `N`) make up approximately one half of the benchmark. We found that in the benchmarks of 2002, 2004, 2005, 2011, 2012, 2014, and 2015, feature-equivalent formulae are contained[5].

3.2 Redundancy in Combined Benchmarks

Usually, researchers evaluate their solver extensions on a benchmark of a single competition, e.g., the most recent one. One reason might be that research groups lack computational resources to execute solvers on a larger benchmark. On the other hand, reviewers request evaluations on a wider range of benchmarks. In Section 4, we present a method for speeding up solver evaluation by exploiting the redundancy present in combined benchmarks. This redundancy has a direct impact on the achievable speedup as the main idea of the proposed method consists in restricting the experiment to the redundancy-free extract of the combined benchmarks. As an example, the data in Table 2 represent the portion of the formulae which can be ignored when performing an evaluation based on the benchmarks of two competitive events. Indicating the numbers for the combination of more events is beyond the scope of this paper[6].

The data in Table 2 may be interpreted in two different ways: On the one hand, the data

[5]In 2008, there were two competition phases where formulae were used in both phases. This lead to the reported duplicates.

[6]For this purpose, we provide a collection of scripts that can be adapted to extract the set of unique formulae from an arbitrary multiset of formulae.

Table 2: Redundancy in the combination of the benchmarks of two competitive events. The presented values refer to the relative amount (in percent) of duplicates in the combination of the benchmarks of the year indicated in the row and column head, respectively.

	02	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**
03	0.21												
04	2.52	3.84											
05	0.30	0.34	2										
06	0.75	2.28	6.84	2.17									
07	0.20	0	1.80	4.56	9.09								
08	2.04	5.96	10.88	5.85	16.33	10.40							
09	0.54	2.68	5.36	2.14	10.97	16.49	11.59						
10	0.22	0.46	2.36	1.45	4.50	9.45	9	9.69					
11	1.33	4.06	3.53	2.52	3	5.26	6.2	9.80	6				
12	4.48	9.04	10.70	7.22	13.43	17.03	16.25	22.65	12.71	22.56			
13	0.18	2.15	1.76	1.89	3.50	5.68	5.40	8.78	3.75	10	18.89		
14	0.35	3.58	2.56	1.26	2	4	5.80	10.64	4.75	9.67	18.22	15.33	
15	1.06	2.63	3.21	2.73	2.25	4.42	5.80	6.59	4.50	6.33	12	8.17	23.83

represent the portion of computational resources which can be saved. As an example, if a solver is evaluated on the combination of the 2009 and 2012 benchmarks, this saving amounts to 22.65 %. Hence, it is sufficient to perform the computation on 77.35 % of the benchmark data, since it provides the information needed for inferring the results for the redundant portion of the benchmark. On the other hand, results presented for two competitive events have to be read with a grain of salt. In the combination of the 2009 and 2012 benchmarks, 22.65 % of the formulae actually are duplicates. A solver modification exploiting this knowledge might be emphasized on this combination of benchmarks, as this modification is especially effective on duplicate formulae, while failing on another combination (as discussed in Section 1). The data in Table 2 point this weakness out, encouraging developers to present evaluations on benchmarks containing distinct formulae only, or to include a discussion on the evaluation on the intersection of benchmarks.

An extreme case in our data is represented by the combination of the benchmarks of 2014 and 2015. In this data set, 76.17 % percent of the formulae are unique, thus the saving of computation amounts to 23.83 %. Hence, solvers applying appropriate techniques might perform best on this combination and worst on the combination of the 2003 and 2007 benchmarks, since the latter combination lacks redundancy. Therefore, it is essential to analyze the benchmark structure in order to make significant predictions concerning the solver performance on new benchmarks.

An even more important fact has to be considered: Benchmarks are combined with the aim to obtain a larger and more representative data set to perform an evaluation on. With this intention in mind, one would expect to obtain more meaningful results. However, due to the redundancy in the benchmark, the results are biased and a clear conclusion cannot be drawn if they are reported only per year.

Table 3: Redundancy in the composition of benchmarks for intervals. For each year (**02** – **15**), the number of formulae (Σ), the number of unique formulae (`U`) and the percentage of unique formulae (`U` (%)) rounded to one decimal place, and the respective average values of their increase (øΔ) are listed for the combination of all benchmarks from 2002 to the year referred to in the column header.

	02	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	øΔ
Σ	829	948	1272	1448	1548	1723	1923	2215	2315	2615	3215	3515	3815	4115	252.77
`U`	827	946	1235	1410	1481	1621	1728	1889	1932	2111	2293	2451	2603	2761	148.77
`U` (%)	99.8	99.8	97.0	97.4	95.7	94.1	89.9	85.3	83.5	80.7	71.3	69.7	68.2	67.1	-2.52

4 Speeding up Evaluations

The weakness pointed out in Section 3.2 can also be turned into a benefit: In order to be able to present an evaluation on multiple competitive events, the actual computation can be restricted to a subset of the benchmark, namely to its redundancy-free extract. From the results obtained on this set of uniques, the results for the duplicates can be inferred. As an example, for comparing a novel solver to existing systems on the combination of the benchmarks of 2014 and 2015, by adopting this approach the computation is reduced by 23.83 %[7]. To this end, we propose the following tool chain:

1. select the benchmarks to be used for the evaluation;
2. collect all formulae;
3. compute the redundancy-free extract of the combined benchmarks;
4. perform the evaluation on this redundancy-free benchmark; and
5. duplicate the results for duplicate formulae.

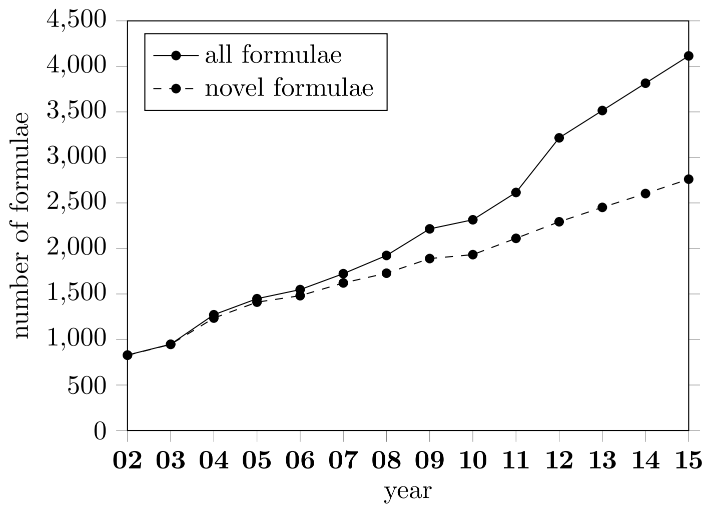
Along with this publication we provide tools that, given a benchmark, perform steps 3 to 5. By means of this procedure, computational resources are saved. The reduction depends on the portion of redundancy contained in the benchmark combination and may be significant. We investigated its effect on the combination of benchmarks for all intervals ranging from 2002 to 2015. The results are presented in Table 3.

The data show that both the number of novel formulae (represented by the increase of Σ) and the number of unique formulae (`U`) increase with time, the mean increase of the latter being about 60 % of the increase of the former (øΔ). The percentage of the unique formulae (`U` (%)) therefore decreases with time, resulting in the fact that the more competitive events are added to the series, the higher is the potential for saving computational resources. As an example, when evaluating a tool for all instances, 67.1 % of the formulae have to be considered. The computation saved therefore amounts to 32.9 %. These trends are visualized in Figure 1.

Summing up, solver evaluation should be carried out on a redundancy-free benchmark. This method eliminates side effects stemming from redundancy in the benchmarks on which the evaluation is performed, and, thus, allows for a meaningful assessment of the robustness of an approach with respect to the given benchmark[8]. As discussed before, if the evaluation is reported on a per year basis, its outcome on the intersection of the presented benchmarks has to be discussed as well.

[7] Assuming a uniformly distributed run time of the tools.

[8] Discussing how a good benchmark should be selected for a representative evaluation is beyond the scope of this work. See [6] for more details.



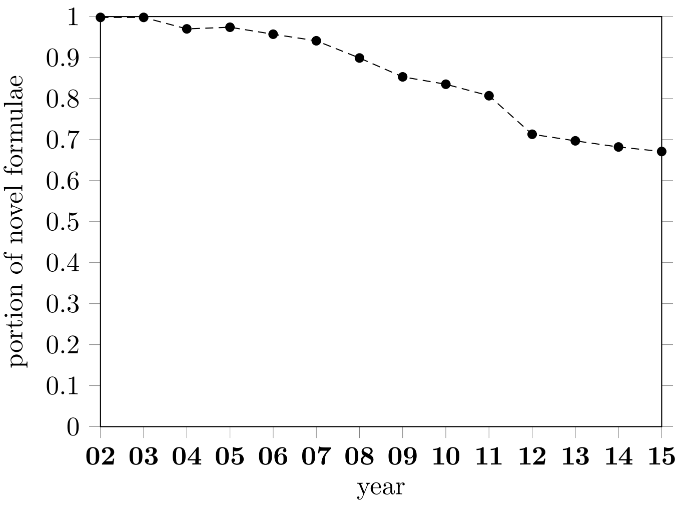


Figure 1: Trend regarding the composition of benchmarks for intervals from 2002. In the upper figure, the trend with respect to the number of formulae and the number of novel formulae is visualized for the combination of all benchmarks from 2002 until the year indicated on the x-axis. The number of novel formulae shows a slower increase than the number of all formulae, hence the the portion of novel formulae decreases with time, as shown in the figure below.

5 Results

The procedure presented in Section 4 was tested using Glucose 3.0, Lingeling baq, and Riss 6. As test data, we chose all application benchmarks used in the competitive events from 2002 to 2015. The experiment was run on a cluster where each node is equipped with 2 Intel Xeon CPU E5-2680 v3 with 12 cores running at 2.50 GHz. Each process was run with a CPU time limit of 1 hour and a main memory limit of 6.5 GB using every 4th core.

In a first step, we executed steps 3 to 5 of our tool chain obtaining evaluation results on the redundancy-free extract of the combined benchmarks and the results mapped onto the entire

Table 4: The entries show the results of the evaluation executed on the redundancy-free extract of the combined benchmarks, after mapping the results to the entire benchmark, and the evaluation conducted on the entire benchmark. usc, # solved, total time and PAR denote unique solver contribution, number of instances solved, total run time and penalized average run time, respectively.

experiment	solver	usc	# solved	total time [s]	PAR [h]
redundancy-free extract	Glucose 3.0	14	2117	453120	769.87
	Lingeling baq	88	2213	521379	692.83
	Riss 6	44	2155	540352	756.10
mapped formulae	Glucose 3.0	31	3261	899943	1103.98
	Lingeling baq	152	3393	1029790	1008.05
	Riss 6	69	3293	1024330	1106.54
full benchmark	Glucose 3.0	31	3256	890734	1106.43
	Lingeling baq	151	3388	1007180	1006.77
	Riss 6	68	3297	1035320	1105.59

formula multiset. In a second step, we ran the solvers on all formulae, including the ones that were not evaluated in the first step. Recall from Table 3 that the combination of all benchmarks used in competitive events since 2002 comprises 4115 formulae, 2761 of which are unique. The portion of unique formulae hence amounts to 67.1 %. One would therefore expect run time savings in the size of 32.9 % compared to the evaluation on the entire benchmark. The results are presented in Table 4.

The *unique solver contribution (ucs)* refers to the number of formulae solved only by the respective solver. The *total time* refers to the run time used for the instances which were solved. The *penalized average run time (PAR)* is obtained by adding to the total run time the number of unsolved formulae multiplied by the timeout. It therefore accounts for unsolved instances as well, and it should also be affected by the run time saving due to the exploitation of redundancy in the benchmark. From the results obtained from the run on the redundancy-free extract of the combined benchmarks, the run times for the entire benchmark are computed by duplicating the appropriate run time for the duplicate formulae. In the same manner, the number of solved instances was calculated for the entire benchmark. The data for *mapped formulae* therefore represent what we expect to find when the experiment is run on the entire benchmark.

Let us compare the results for the mapped formulae and the full benchmark. The usc is very accurate, it differs by 1 for Lingeling and Riss and is equal for Glucose. The differences in the number of solved instances amount to 0.15 % for Glucose and Lingeling, and 0.13 % for Riss, respectively. The differences for Glucose, Lingeling, and Riss concerning the total time amount to 1.02 %, 2.2 %, and 1.06 %, respectively. Riss was slower than expected while Glucose and Lingeling were faster than expected. The PAR is significantly more accurate: The differences amount to 0.22 %, 0.13 %, and 0.13 % for Glucose, Lingeling, and Riss, respectively. For each of the instances not solved, 1 h is added to the PAR. Glucose shows a higher PAR than expected, contrarily to Lingeling and Riss. Differences of this scale may also be due to the run time environment.

The PAR value of the solvers represents a meaningful value for the given evaluation. As we use a penalty factor of 1 (contrary to the commonly used PAR10 score), the PAR value of a solver refers to the time required by this solver to perform the evaluation. By comparing the values for the unique formulae and the full benchmark, calculating the saving is straightforward: We save 336.56, 313.94 and 349.49 hours for the three solvers. The amount of computational resources saved can be used for evaluating an additional solver on the same benchmark. The computation time for the features on all crafted and application formulae that are available from 2002 to 2015, including the unused formulae, amounts to about 60 hours. Hence, for the selected benchmark, the usage of the presented procedure already pays when one single solver is evaluated. Note that, in this work we considered application formulae only.

6 Conclusions

Evaluation and benchmarking are important and are affected by the setup and the used benchmarks. We therefore analyzed the structure of the application benchmarks used in the competitive SAT events between 2002 and 2015. We introduced the notion of *feature-equivalence* in order to identify duplicate formulae accounting for the *redundancy* in a benchmark. If a solver is evaluated on a benchmark containing redundancy, it is not possible to predict its performance on a different benchmark, unless either this benchmark contains the same portion of redundancy or the solver does not exploit this redundancy. Therefore, the solver's performance on the redundant formulae has to be taken into account in the discussion on evaluation results. We showed that from 2008, the benchmarks contained about 50 % novel formulae. For this reason, the most promising approach to be placed well in the current year's competition is to optimize a solver for last year's benchmark.

We propose a method for saving computational resources when a solver is evaluated on a combination of benchmarks. This saving is achieved by performing the evaluation on the redundancy-free extract of the combined benchmarks and amounts to the portion of the redundancy contained in the combined benchmarks. For the combination of all application benchmarks used in the competitive SAT events between 2002 and 2015, by this method about 30 % of the run time is saved. We provide appropriate tools to

1. extract features;
2. match duplicate formulae (based on features); and
3. scale the evaluation on the redundancy-free extract up to the full benchmark again.

These tools will be available for download on our website[9] and facilitate the addition of novel benchmarks in the future. With our method, developers can obtain meaningful evidence regarding a novel extension while saving computational resources. We present experimental data confirming the suitability of the chosen features and the feasibility of our approach.

From our method, various data result, e.g., formula features, feature calculation period, and run time of the solver. These data can be used for diverse learning tasks, e.g., determine feature correlation or predict suitable solver configurations.

When reporting evaluation results for benchmarks of two (or more) years independently, then a discussion on the formulae contained in the intersection of all used benchmarks has to be presented. Otherwise, no clear conclusion can be drawn. We pointed out that for obtaining an unbiased assessment, the evaluation results with respect to the redundancy have to be taken into account. The higher the percentage of solved duplicate formulae, the worse the solver will perform on novel formulae lacking redundancy. We emphasize that for this reason redundancy-free benchmarks should be used for evaluating novel solvers and solver extensions. Evaluations

[9]http://tools.computational-logic.org/content/evaluation.php

should follow ideas proposed in publications, such as benchmark construction [6], or report effects per family, as done in [4].

It would be best, of course, to provide a redundancy-free benchmark. Be it that enough new formulae are available, the ideal solution consists in including only newly submitted formulae in the benchmark. If formulae from old benchmarks have to be included, e.g., due to the lack of benchmark data, redundancy-free benchmarks can be built by considering for every year formulae submitted for the first time only. If old formulae are contained in a competition benchmark, we suggest that a list of all included formulae per year is provided. Taking into account the discussion of Section 3, this approach facilitates the determination of the redundancy in the benchmark allowing for an unbiased solver evaluation.

As a final remark, we want to make both authors of experimental data as well as reviewers aware, that whenever data for several independent benchmarks are or should be provided, the question whether the benchmarks are actually independent has to be considered. For the application track of the SAT competition benchmarks, we showed that most benchmarks are not independent.

7 Future Work

There are several aspects which should be addressed in the future. In this work, we focus on SAT solving. However, our results can be adapted to other problems, e.g., QBF, MaxSAT, PB, CSP, and AIG, and are therefore of relevance for other competitions as well. To this end, an analogon to the CNF features has to be identified for the respective problems. The definition of such an analogon should be straightforward, since most of the features used in our work are based upon graphs which can be constructed in most symbolic languages. In a next step, the classifier applied for the computation of the formula features has to be adapted accordingly. Evaluation method and assessment of the results principally remain unmodified. Our results are therefore of interest for authors participating in other competitions as well, e.g., in HWMCC, PB, MaxSAT, SMT, and QBF as well as Planning.

In this work, we addressed the application track of the SAT competition benchmarks. The structure of the crafted benchmarks can be investigated by means of the tools we provide on our website.

We used the features presented in [1]. However, different features could prove adequate as well for identifying the redundancy contained in the combination of benchmarks.

Acknowledgments. The authors thank the DFG for supporting this work under the grant HO 1294/11-1 and the ZIH of TU Dresden for providing the computational resources to produce the experimental data for the empirical evaluation.

References

[1] Enrique Alfonso and Norbert Manthey. New CNF features and formula classification. In Daniel Le Berre, editor, *POS'14*, volume 27 of *EPiC Series*, pages 57–71. EasyChair, 2014.

[2] Adrian Balint, Anton Belov, Marijn Heule, and Matti Järvisalo, editors. *Proceedings of SAT Competition 2013; Solver and Benchmark Descriptions*, volume B-2013-1. University of Helsinki, Department of Computer Science Series of Publications B, 2013.

[3] Anton Belov, Daniel Diepold, Marijn J.H. Heule, and Matti Järvisalo, editors. *Proceedings of SAT Competition 2014*, volume B-2014-2 of *Department of Computer Science Series of Publications B*. University of Helsinki, Helsinki, Finland, 2014.

[4] Armin Biere and Andreas Fröhlich. Evaluating CDCL restart schemes. In *POS'15*, 2015.

[5] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[6] Holger H. Hoos, Benjamin Kaufmann, Torsten Schaub, and Marius Schneider. Robust benchmark set selection for boolean constraint solvers. In Giuseppe Nicosia and Panos M. Pardalos, editors, *LION 7, Revised Selected Papers*, volume 7997 of *LNCS*, pages 138–152. Springer, 2013.