Software Tools for Technology Transfer manuscript No.
(will be inserted by the editor)

VerifyThis 2015

A Program Verification Competition

Marieke Huisman[1], **Vladimir Klebanov**[2], **Rosemary Monahan**[3], **Michael Tautschnig**[4]

[1] University of Twente, The Netherlands, e-mail: m.huisman@utwente.nl
[2] Karlsruhe Institute of Technology, Germany, e-mail: klebanov@kit.edu
[3] Maynooth University, Ireland, e-mail: Rosemary.Monahan@nuim.ie
[4] Queen Mary University of London, e-mail: michael.tautschnig@qmul.ac.uk

Received: date / Revised version: date

Abstract. VerifyThis 2015 was a one-day program verification competition which took place on April 12th, 2015 in London, UK as part of the European Joint Conferences on Theory and Practice of Software (ETAPS 2015). It was the fourth instalment in the VerifyThis competition series. This article provides an overview of the VerifyThis 2015 event, the challenges that were posed during the competition, and a high-level overview of the solutions to these challenges. It concludes with the results of the competition, and some ideas and thoughts for future instalments of VerifyThis.

1 Introduction

VerifyThis 2015 took place on April 12th, 2015 in London, UK, as a one-day verification competition in the European Joint Conferences on Theory and Practice of Software (ETAPS 2015). It was the fourth edition in the VerifyThis series after the competitions held at FoVeOOS 2011, FM2012 and Dagstuhl (Seminar 14171, April 2014).

The aims of the competition were:

- to bring together those interested in formal verification, and to provide an engaging, hands-on, and fun opportunity for discussion
- to evaluate the usability of logic-based program verification tools in a controlled experiment that could be easily repeated by others.

Typical challenges in the VerifyThis competitions are small but intricate algorithms given in pseudo-code with an informal specification in natural language. Participants have to formalise the requirements, implement a solution, and formally verify the implementation for adherence to the specification. There are no restrictions on the programming language and verification technology used. The time frame to solve each challenge is quite short (between 45 to 90 minutes) so that anyone can easily repeat the experiment. Examples of the verification challenges are available from the VerifyThis website http://www.verifythis.org/.

The correctness properties which the challenges present are typically expressive and focus on the input-output behaviour of programs. To tackle them to the full extent, some human guidance within a verification tool is usually required. At the same time, considering partial properties or simplified problems, if this suits the pragmatics of the tool, is encouraged. The competition welcomes participation of automatic tools as combining complementary strengths of different kinds of tools is a development that VerifyThis would like to advance.

Submissions are judged for correctness, completeness, and elegance. The focus includes the usability of the tools, their facilities for formalizing the properties and providing helpful output.

1.1 VerifyThis 2015

VerifyThis 2015 consisted of three verification challenges. Before the competition, an open call for challenge submissions was made. As a result, six challenges were submitted, of which one was selected for the competition (see also Section 5.5 for more details about this call and the selection criteria). The challenges (presented later) provided reference implementations at different levels of abstraction. For the first time, one of the challenges centered around concurrency.

Fourteen teams participated (Table 1). Teams of up to two people were allowed and physical presence on site was required. We particularly encouraged participation of:

- student teams (this includes PhD students)

– non-developer teams using a tool someone else developed
– several teams using the same tool

Teams using different tools for different challenges (or even for the same challenge) were welcome.

As in the VerifyThis 2012 competition, after the competition a post-mortem session was held, where participants explained their solutions and answered questions of the judges. In parallel, the participants used this half-day session to discuss details of the problems and solutions among each other.

The website of the 2015 instalment of VerifyThis can be found at http://etaps2015.verifythis.org/. More background information on the competition format and the rationale behind it can be found in [HKM12]. Reports from previous competitions of similar nature can be found in [KMS+11, BBD+11, FPS12], and in the special issue of the International Journal on Software Tools for Technology Transfer (STTT) on the VerifyThis competition 2012 (see [HKM15] for the introduction).

1.2 Rules

In order to ensure that the competition proceeded smoothly, the following rules were established:

1. The main rule of the competition is: *no cheating is allowed*. The judges may penalise or disqualify entrants in case of unfair competition behaviour and may adjust the competition rules to prevent future abuse.
2. Solutions are to be submitted by email.
3. Submissions must state the version of the verification system used (for development versions, internal revision, time-stamp, or similar unique id).
4. It is permitted to modify the verification system during the competition. This is to be noted in the solution(s).
5. All techniques used must be general-purpose, and are expected to extend usefully to new unseen problems.
6. Internet access is allowed, but browsing for problem solutions is not.
7. Involvement of other people beyond those on the team is not allowed.
8. While care is taken to ensure correctness of the reference implementations supplied with problem descriptions, the organisers do not guarantee that they are indeed correct.

2 Challenge 1: Relaxed Prefix (60 minutes)

This problem was submitted by Thomas Genet, Université de Rennes 1, in response to the open call for challenges.

2.1 Verification Task

Verify a function `isRelaxedPrefix` determining if a list `pat` (for pattern) is a *relaxed prefix* of another list `a`. The relaxed prefix property holds iff `pat` is a prefix of `a` after removing at most one element from `pat`.

Examples:

– `pat = {1,3}` is a relaxed prefix of `a = {1,3,2,3}` (standard prefix)
– `pat = {1,2,3}` is a relaxed prefix of `a = {1,3,2,3}` (remove 2 from pat)
– `pat = {1,2,4}` is not a relaxed prefix of `a = {1,3,2,3}`

Implementation notes: One may implement lists as arrays, e.g., of integers. A reference implementation is given below. It may or may not contain errors.

```
public class Relaxed {

public static boolean isRelaxedPrefix(int[] pat,int[] a){

     int shift = 0;
     for(int i=0; i<pat.length; i++) {
         if (pat[i]!=a[i-shift])
             if (shift==0) shift=1;
                 else return false;
     }
     return true;
}
public static void main(String[] argv) {
     int[] pat = {1,2,3};
     int[] a1 = {1,3,2,3};
     System.out.println(isRelaxedPrefix(pat, a1));
}
}
```

2.2 Comments on Solutions

Eleven teams (Verifast, Why3, Autoproof, KeY, Dafny (3 teams), mCRL2, F*, KIV, and VerCors) submitted a solution to this challenge. Difficulties that had been encountered by the participants were mainly at the specification level: getting the prefix definition correct, and making sure that all cases were covered in the postconditions. In particular, several teams forgot the case where the length of the array was less than the length of the prefix, or where the method returned false. In the overall evaluation, the solution provided by the Why3 team was the only one to obtain full marks from the judges.

During the verification, the main challenge was to find an appropriate instantiation for the existential quantifier. Different solutions for this were used: the Why3 team brought the specification into a particular syntactical shape that enabled the SMT solver to guess the instantiation (in a post-competition solution, this trick was replaced with an explicit assertion); the KeY team and the AutoProof team used an explicit return value, which avoided the need for existential quantification (witness computed by the program); Tim Wood,

Table 1. Teams participating in VerifyThis 2015 (alphabetically by tool)

#	Team members	Tool		Team attributes
1	Nadia Polikarpova, Carlo Furia	AutoProof	[TFNP15]	
2	Michael Tautschnig	CBMC	[KT14]	
3	Rustan Leino	Dafny	[Lei10]	
4	Tim Wood	Dafny	— " —	student, non-developer
5	Robert Kelly, Marie Farrell	Dafny	— " —	student, non-developer
6	Aleksey Schubert	Frama-C	[KKP+15]	non-developer
7	Simon Forest, Jean Karim Zinzindohoué	F*	[SCF+13]	student
8	Daniel Bruns, Michael Kirsten	KeY	[ABB+14]	student
9	Gidon Ernst, Jörg Pfähler	KIV	[EPS+14]	student
10	Jan Friso Groote	mCRL2	[CGK+13]	
11	Jonathan Hoyland	MoCHi	[KSU11]	non-developer
12	Stefan Blom, Saeed Darabi	VerCors	[BH14]	
13	Bart Jacobs	VeriFast	[PJP15]	
14	Jean-Christophe Filliâtre, Guillaume Melquiond	Why3	[FP13]	

using Dafny, used an explicit hint in form of a trigger annotation; the KIV team tried to address this by manual instantiation; while Robert Kelly and Marie Farrell, using Dafny, provided a recursive definition of a relaxed prefix.

2.3 Future Verification Tasks

For those who had completed the challenge quickly, the description included a further challenge, outlined below. No submissions attempting to solve the advanced challenge were received during the competition.

Verification task: Implement and verify a function `relaxedContains(pat, a)` returning whether `a` contains `pat` in the above relaxed sense, i.e., whether `pat` is a relaxed prefix of any suffix of `a`.

3 Challenge 2: Parallel GCD (60 minutes)

Various parallel algorithms for computing the greatest common divisor `GCD(a,b)` exist (cf. [Sed08]). In this challenge, we consider a simple Euclid-like algorithm with two parallel threads. One thread performs subtractions of the form `a:=a-b`, while the other thread performs subtractions of the form `b:=b-a`. Eventually, this procedure converges on the GCD.

In pseudo-code, the algorithm is described as follows:

```
(
 WHILE a != b DO
     IF a>b THEN a:=a-b ELSE SKIP FI
 OD
||
 WHILE a != b DO
      IF b>a THEN b:=b-a ELSE SKIP FI
 OD
);
OUTPUT a
```

3.1 Verification Task

Specify and verify the following behaviour of this parallel GCD algorithm:

Input: two positive integers `a` and `b`
Output: a positive integer that is the greatest common divisor of `a` and `b`

Synchronisation can be added where appropriate, but try to avoid blocking of the parallel threads.
Sequentialisation: If a tool does not support reasoning about parallel threads, one may verify the following pseudo-code algorithm:

```
WHILE a != b DO
   CHOOSE(
        IF a > b THEN a := a - b ELSE SKIP FI,
        IF b > a THEN b := b - a ELSE SKIP FI
   )
OD;
OUTPUT a
```

3.2 Comments on Solutions

Five teams (Verifast, mCRL2, KIV, CBMC and VerCors) submitted a solution to the concurrent version of this challenge; six teams (Why3, Autoproof, KeY, Dafny (2 teams) and F*) submitted a solution to the sequentialised variant of the challenge.

The solutions to the concurrent program were all very different in spirit. Bart Jacobs (VeriFast) developed a fine-grained concurrent solution for the fully

general problem, where individual load and stores were considered to be atomic operations. He assumed the necessary GCD properties, as well as sequential consistency, and then focused on the shared memory aspects of the problem, proving that the individual threads were correct. Despite its significant complexity, his solution to this problem using "shared boxes" (which integrate rely-guarantee reasoning into the separation logic of VeriFast) obtained the best marks from the judges. Closely following was a solution by Jan Friso Groote with mCRL2. He needed to provide a bound on the possible input parameters, because mCRL2 uses finite state model checking for verification. The CBMC and KIV teams developed partial solutions to this problem, assuming that each iteration of the while loop was executed atomically, i.e. it is assumed that there is no interference possible while executing the loop body. The KIV team got stuck on the necessary GCD properties (where later they realised that they actually could have used GCD properties from the KIV libraries). Finally, the VerCors team submitted a solution, making use of the recently added support for parallel blocks, but proving absence of data races only. Post-competition they extended this to a full solution.

The judges were also impressed by the attempt of the AutoProof team. In addition to proving correctness of the sequentialised algorithm, they almost succeeded in proving termination of the sequential version, assuming an appropriate fairness condition.

The KIV team was not the only team that experienced that well-developed libraries can help while solving a challenge. The Why3 team solved the sequentialised version of this problem within 15 minutes, because their tool has a powerful GCD library, while Rustan Leino (Dafny) struggled with this (sequentialised) challenge because of the lack of appropriate Dafny libraries.

After the competition, Rustan Leino developed a Dafny solution for the concurrent program, by writing a program that explicitly encoded all the possible interleavings between the different threads, while using explicit program counters for each thread.

4 Challenge 3: Dancing Links (90 minutes)

Dancing links is a search technique introduced in 1979 by Hitotumatu and Noshita [HN79] and later popularised by Knuth [Knu00]. The technique can be used to efficiently implement a search for all solutions of the exact cover problem, which in its turn can be used to solve Tiling, Sudoku, N-Queens, and other related problems.

Suppose `x` points to a node of a doubly linked list; let `L[x]` and `R[x]` point to the predecessor and successor of that node. Then the operations

```
L[R[x]] := L[x];
R[L[x]] := R[x];
```

remove `x` from the list. The subsequent operations

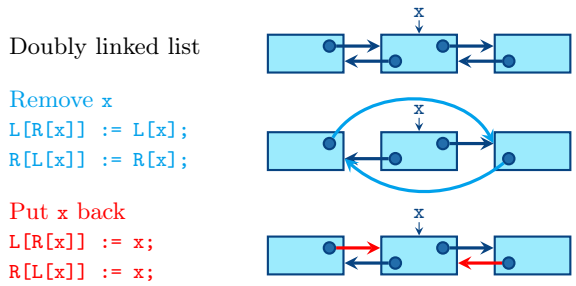

Fig. 1. Graphic illustration of dancing links operations (inspired by Wim Bohm)

```
L[R[x]] := x;
R[L[x]] := x;
```

will put `x` back into the list again. Figure 1 provides a graphical illustration of this process.

4.1 Verification Task

Implement the data structure with these operations, and specify and verify that they behave in the way described above.

4.2 Comments on Solutions

Several participants reported that this had been a difficult challenge, and in particular it had taken them time to understand the full details of the intended behaviour. Ten solutions (Verifast, Why3, AutoProof, KeY, Dafny (2 teams), mCRL2, F*, KIV and CBMC) to the challenge were submitted. During the competition the organisers clarified that the main challenge was in managing the remove and unremove – many elements can be removed and unremoved but they must be unremoved in reverse order (otherwise the references are not maintained).

Rustan Leino (using Dafny) was the only one to address this challenge completely within the allocated time. The AutoProof team became very excited about this challenge, because it was an ideal challenge for demonstrating a technique called *semantic collaboration*, which they had recently developed [PTFM14]. This technique helped them to express invariants over the circular dancing lists. The Why3 team also used circular lists, which helped to formulate and prove the desired properties easily. Other teams, which used non-circular lists explicitly had to mark nodes that had been removed, which complicated verification, and required the use of abstraction techniques to capture the actual list structure.

5 Results, Statistics, and Overall Remarks

We conclude this report with various data points and summaries of results.

5.1 Awarded Prizes and Statistics

Prizes were awarded in the following categories:

- Best team: team Why3 – Jean-Christophe Filliâtre and Guillaume Melquiond
- Best student team: team KIV – Gidon Ernst and Jörg Pfähler
- Distinguished user-assistance tool feature – awarded to two teams:
  - Why3 for the lemma library (as demonstrated by its use in the competition)
  - mCRL2 for a rich specification language in an automated verification tool
- Best challenge submission: Thomas Genet for the Relaxed Prefix problem, which was used as Challenge 1 in the competition
- Tool used by most teams: Dafny

The best student team received a 500 Euro cash prize donated by our sponsors while the best overall team received 150 Euros. Smaller prizes were also awarded for the best problem submission and the distinguished user-assistance tool feature.

5.2 Statistics per Challenge

- Relaxed Prefix: 11 submissions were received, of which only the submission by Jean-Christophe Filliâtre and Guillaume Melquiond (Why3) was judged as correct and complete.
- Parallel GCD: 11 submissions were received, of which only the submission by Bart Jacobs (Verifast) was judged as correct and complete. Six of the submitted solutions were restricted to the sequential version of the challenge.
- Dancing Links: 10 submissions were received, of which only the submission by Rustan Leino (Dafny) was judged as correct and complete.

5.3 Travel Grants

The competition had funds for a limited number of travel grants for student participants. A grant covered the incurred travel and accommodation costs up to EUR 250 for those coming from Europe and EUR 500 for those coming from outside Europe. Evaluation criteria were qualifications (for the applicant's career level), need (explained briefly in the application), and diversity (technical, geographical, etc.). Six travel grants were awarded.

5.4 Post-mortem Sessions

Two concurrent post-mortem sessions were held on the afternoon of the competition (stretching to the day after the competition, given the large number of participants). These sessions were much appreciated, both by the judges and by the participants. It was very helpful for the judges to be able to ask the teams questions in order to better understand and appraise their submissions. Concurrently, all other participants presented their solutions to each other. We would recommend such a post-mortem session for any on-site competition. In future editions of the competition we intend to extend this aspect of the event as participants reported the time used as invaluable, providing lively discussions about the challenges, gaining knowledge about tools through presenting challenge solutions to each other and exchanging ideas about future tool developments and solution strategies.

5.5 Soliciting Challenges

After much discussion at the previous competition, on how to extend the problem pool and tend better to the needs of the participants, we issued a call for challenges to extend the problem pool. The call stipulated that

- a problem should contain an informal statement of the algorithm to be implemented (optionally with complete or partial pseudo-code) and the requirement(s) to be verified
- a problem should be suitable for a 60-90 minute time slot
- submission of reference solutions is strongly encouraged
- problems with an inherent language- or tool-specific bias should be clearly identified as such
- problems that contain several subproblems or other means of scaling difficulty are especially welcome
- the organisers reserve the right (but no obligation) to use the problems in the competition, either as submitted or with modifications
- submissions from (potential) competition participants are allowed

We received six suggestions for challenges, and decided that one was suited for use during the competition. This challenge was practical, easy to describe to participants, suitable in duration for the competition and could be easily adapted to suit different environments. However, even though we decided not to use all of the submitted challenges directly[1], the call for submissions provided inspiration for further challenges as well as insight in what people in the community consider interesting, challenging and relevant problems for state-of-the-art verification tools.

5.6 Session Recording

This year, for the first time, the organizers encouraged the participants to record their desktop during the competition (on a voluntary basis). The recording would give

[1] primarily due to the time restrictions on challenge length

insight into the pragmatics of different verification systems and allow the participants to learn more from the experience of others deriving a solution. The organisers provided a list with recording software suggestions, though so far only a solution for Linux (Freeseer) could be successfully tested. The main criteria are free availability, ease of installation, and low CPU load.

In general, participants agreed that recording could provide useful information, but, as far as we know, only the KIV team actually made a recording.

5.7 Related Events

VerifyThis 2015 is the 4th event in the VerifyThis competition series. Related events are the Verified Software Competition (VSComp, http://vscomp.org) held online, the Competition on Software Verification (SV-COMP [Bey15], http://sv-comp.sosy-lab.org) focusing on evaluating systems in a way that does not require user interaction[2], and the RERS Challenge ([HIM+14], http://www.rers-challenge.org), which is dedicated to rigorous examination of reactive systems, using different technologies such as theorem proving, model checking, program analysis, symbolic execution, and testing.

VerifyThis is also a collection of verification problems (and solutions). Its counterpart is VerifyThus (http://verifythus.cost-ic0701.org/) – a distribution of deductive verification tools for Java-like languages, bundled and ready to run in a VM. Both were created with support from COST Action IC0701.

A workshop on comparative empirical evaluation of reasoning systems (COMPARE2012 [KBBS12]) was held at IJCAR 2012 in Manchester. Competitions were one of the main topics of the workshop.

5.8 Judging Criteria

Limiting the duration of each challenge assists the judging and comparison of each solution. However, this task is still quite subjective and hence, difficult. Discussion of the solution with the judges typically results in a ranking of solutions for each challenge. In future editions of the competition we envisage that each team would complete a questionnaire for each challenge on submission. This would assist the judging and would also encourage teams to reflect on their solutions.

Criteria that were used for judging were:

- Correctness: is the formalisation of the properties adequate and fully supported by proofs?
- Completeness: are all tasks solved, and are all required aspects covered?
- Readability: can the submission be understood easily, possibly even without a demo?
- Effort and time distribution: what is the relation between time expended on implementing the program vs. specifying properties vs. proving?
- Automation: how much manual interaction is required, and for what aspects?
- Novelty: does a submission apply novel techniques?[3]

A novelty for VerifyThis this year was the inclusion of a judge with a background in software model checking (the fourth author of this paper). He observed that the participants could have been more critical, reflecting on their solutions. To use the tools, often expert knowledge is necessary, and the tools are not very good at providing feedback when a proof attempt fails. For future competitions, he felt that the most interesting aspect would be new insights, leading to further improvements to the tool. This aspect was also mentioned by some of the competition participants. It will be worthwhile investigating what novelties have resulted from earlier competitions.

5.9 Post-Competition Discussion

Directly after the competition, before starting the post-mortem session, a plenary discussion was held to gather the opinion of the participants about the organisation of future competitions. The following topics were discussed:

Challenges: The participants agreed that it was interesting and timely to have a concurrency-related challenge. In general the feeling was that it is good to have modular challenges, which can be broken down into smaller subproblems. There was also a suggestion to have challenges in the form: given a verified program, extend it to...

Tool vs. user: An interesting aspect remains regarding what we are actually measuring: the tool or the user. To focus more on measuring the tool, the challenge descriptions could include an informal description of the necessary invariants. However, it was also remarked that this might restrict the variety of tools participating in the competition.

Another possibility, to help focus the competition on the tools, would be to create mixed teams, where you use a tool that you do not know in advance (possibly with a tutor). As a result of this discussion, in the next edition of this competition, we plan to start the day with a Dafny tutorial, followed by an out-of-competition challenge, open to anybody interested in participating.

Timing: The program as it is now, is very dense. A slightly larger break between the challenges would be welcome.

Since participants often continue working on their solutions after the competition, a post-competition deadline to submit solutions would also be welcomed.

[2] SV-COMP is associated with TACAS.

[3] Here, the judges would primarily like to cite the semantic collaboration technique demonstrated by the AutoProof team.

The possibility of providing all challenges to the competitors at the same time was discussed, such that participants can organise their own time to work on a challenge. In that case, to avoid two person teams having an advantage over single person teams, because they can distribute the work, all teams would be allowed the use of only one computer.

Reporting: There was much discussion about the possibility of publishing details from these competitions. There have been several competition report papers, and there has been a special issue of STTT on the VerifyThis competition in 2012. New publications need to provide new insights. One possibility is to encourage several participants to write a joint paper about one particular challenge, where they compare their different solutions. Another possibility is to reach an agreement with an editor to publish a series of competition reports, summarising the main facts of the competition.

In general, the participants agreed that it is important to make the (polished) solutions publicly available for others to inspect and compare. The solutions of the best student team prize winners, the KIV team, are available at https://swt.informatik.uni-augsburg.de/swt/projects/verifythis-competition-2015/, while solutions of the best overall team prize winners, the Why3 team, are available at http://toccata.lri.fr/gallery/why3.en.html. Further solutions to competition challenges may be found at http://etaps2015.verifythis.org/.

5.10 Final Remarks

The VerifyThis 2015 challenges have offered a substantial degree of complexity and difficulty. A new development compared to earlier editions of the competition was the introduction of a concurrency-related challenge. It is also remarkable that this year Jan-Friso Groote participated with an automated verification tool (mCRL2), and submitted solutions to all challenges. Typically, some bounds had to be added to the challenges for the model checking algorithm to work, however, this clearly shows that the gap between model checking and auto-active verification is becoming smaller.

Further, the main insight provided by the solutions this year was the importance of a good library, and of a good specification language.

A new edition of the VerifyThis competition will be held as part of ETAPS 2016.

Acknowledgments

The organisers would like to thank Wojciech Mostowski and Radu Grigore for their feedback and support prior to the competition. The organisers also thank the competition's sponsors: Formal Methods Europe, Galois, Inc., and Microsoft Research. Their contributions helped us to support participants with travel grants, and to finance the various prizes.

References

ABB+14. Wolfgang Ahrendt, Bernhard Beckert, Daniel Bruns, Richard Bubel, Christoph Gladisch, Sarah Grebing, Reiner Hähnle, Martin Hentschel, Mihai Herda, Vladimir Klebanov, Wojciech Mostowski, Christoph Scheben, Peter H. Schmitt, and Mattias Ulbrich. The KeY platform for verification and analysis of Java programs. In Dimitra Giannakopoulou and Daniel Kroening, editors, *6th International Conference on Verified Software: Theories, Tools and Experiments (VSTTE 2014)*, volume 8471 of *LNCS*, pages 55–71. Springer, 2014.

BBD+11. Thorsten Bormer, Marc Brockschmidt, Dino Distefano, Gidon Ernst, Jean-Christophe Filliâtre, Radu Grigore, Marieke Huisman, Vladimir Klebanov, Claude Marché, Rosemary Monahan, Wojciech Mostowski, Nadia Polikarpova, Christoph Scheben, Gerhard Schellhorn, Bogdan Tofan, Julian Tschannen, and Mattias Ulbrich. The COST IC0701 verification competition 2011. In Bernhard Beckert, Ferruccio Damiani, and Dilian Gurov, editors, *International Conference on Formal Verification of Object-Oriented Systems (FoVeOOS 2011)*, volume 7421 of *LNCS*, pages 3–21. Springer, 2011.

Bey15. Dirk Beyer. Software verification and verifiable witnesses - (report on SV-COMP 2015). In Christel Baier and Cesare Tinelli, editors, *21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2015)*, volume 9035 of *LNCS*, pages 401–416. Springer, 2015.

BH14. Stefan Blom and Marieke Huisman. The VerCors tool for verification of concurrent programs. In Cliff B. Jones, Pekka Pihlajasaari, and Jun Sun, editors, *19th International Symposium on Formal Methods (FM 2014)*, volume 8442 of *LNCS*, pages 127–131. Springer, 2014.

CGK+13. Sjoerd Cranen, Jan Friso Groote, Jeroen J. A. Keiren, Frank P. M. Stappers, Erik P. de Vink, Wieger Wesselink, and Tim A. C. Willemse. An overview of the mCRL2 toolset and its recent advances. In Nir Piterman and Scott A. Smolka, editors, *19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2013)*, volume 7795 of *LNCS*, pages 199–213. Springer, 2013.

EPS+14. Gidon Ernst, Jörg Pfähler, Gerhard Schellhorn, Dominik Haneberg, and Wolfgang Reif. KIV: overview and VerifyThis competition. *International Journal on Software Tools for Technology Transfer*, pages 1–18, 2014.

FP13. Jean-Christophe Filliâtre and Andrei Paskevich. Why3 - where programs meet provers. In Matthias

Felleisen and Philippa Gardner, editors, *22nd European Symposium on Programming (ESOP 2013)*, volume 7792 of *LNCS*, pages 125–128. Springer, 2013.

FPS12. Jean-Christophe Filliâtre, Andrei Paskevich, and Aaron Stump. The 2nd Verified Software Competition: Experience report. In Vladimir Klebanov, Armin Biere, Bernhard Beckert, and Geoff Sutcliffe, editors, *1st International Workshop on Comparative Empirical Evaluation of Reasoning Systems (COMPARE 2012)*, volume 873 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

HIM+14. Falk Howar, Malte Isberner, Maik Merten, Bernhard Steffen, Dirk Beyer, and Corina S. Păsăreanu. Rigorous examination of reactive systems. *Int. J. Softw. Tools Technol. Transf.*, 16(5):457–464, October 2014.

HKM12. Marieke Huisman, Vladimir Klebanov, and Rosemary Monahan. On the organisation of program verification competitions. In Vladimir Klebanov, Bernhard Beckert, Armin Biere, and Geoff Sutcliffe, editors, *1st International Workshop on Comparative Empirical Evaluation of Reasoning Systems (COMPARE 2012)*, volume 873 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

HKM15. Marieke Huisman, Vladimir Klebanov, and Rosemary Monahan. Verifythis 2012. *Int. J. Softw. Tools Technol. Transf.*, 17(6):647–657, November 2015.

HN79. Hirosi Hitotumatu and Kohei Noshita. A technique for implementing backtrack algorithms and its application. *Inf. Process. Lett.*, 8(4):174–175, 1979.

KBBS12. Vladimir Klebanov, Bernhard Beckert, Armin Biere, and Geoff Sutcliffe, editors. *Proceedings of the 1st International Workshop on Comparative Empirical Evaluation of Reasoning Systems (COMPARE 2012)*, volume 873 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

KKP+15. Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-C: A software analysis perspective. *Formal Asp. Comput.*, 27(3):573–609, 2015.

KMS+11. Vladimir Klebanov, Peter Müller, Natarajan Shankar, Gary T. Leavens, Valentin Wüstholz, Eyad Alkassar, Rob Arthan, Derek Bronish, Rod Chapman, Ernie Cohen, Mark Hillebrand, Bart Jacobs, K. Rustan M. Leino, Rosemary Monahan, Frank Piessens, Nadia Polikarpova, Tom Ridge, Jan Smans, Stephan Tobies, Thomas Tuerk, Mattias Ulbrich, and Benjamin Weiß. The 1st Verified Software Competition: Experience report. In Michael Butler and Wolfram Schulte, editors, *17th International Symposium on Formal Methods (FM 2011)*, volume 6664 of *LNCS*, pages 154–168. Springer, 2011.

Knu00. Donald E. Knuth. Dancing links. *arXiv preprint cs/0011047*, 2000.

KSU11. Naoki Kobayashi, Ryosuke Sato, and Hiroshi Unno. Predicate abstraction and CEGAR for higher-order model checking. In Mary W. Hall and David A. Padua, editors, *32nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2011)*, pages 222–233. ACM, 2011.

KT14. Daniel Kroening and Michael Tautschnig. CBMC - C bounded model checker - (competition contribution). In Erika Ábrahám and Klaus Havelund, editors, *20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2014)*, volume 8413 of *LNCS*, pages 389–391. Springer, 2014.

Lei10. K. Rustan M. Leino. Dafny: An automatic program verifier for functional correctness. In Edmund M. Clarke and Andrei Voronkov, editors, *16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2010)*, volume 6355 of *LNCS*, pages 348–370. Springer, 2010.

PJP15. Willem Penninckx, Bart Jacobs, and Frank Piessens. Sound, modular and compositional verification of the input/output behavior of programs. In Jan Vitek, editor, *24th European Symposium on Programming (ESOP 2015)*, volume 9032 of *LNCS*, pages 158–182. Springer, 2015.

PTFM14. Nadia Polikarpova, Julian Tschannen, Carlo A. Furia, and Bertrand Meyer. Flexible invariants through semantic collaboration. In Cliff B. Jones, Pekka Pihlajasaari, and Jun Sun, editors, *19th International Symposium on Formal Methods (FM 2014)*, volume 8442 of *LNCS*, pages 514–530. Springer, 2014.

SCF+13. Nikhil Swamy, Juan Chen, Cédric Fournet, Pierre-Yves Strub, Karthikeyan Bhargavan, and Jean Yang. Secure distributed programming with value-dependent types. *J. Funct. Program.*, 23(4):402–451, 2013.

Sed08. Sidi Mohamed Sedjelmaci. A parallel extended GCD algorithm. *J. Discrete Algorithms*, 6(3):526–538, 2008.

TFNP15. Julian Tschannen, Carlo A. Furia, Martin Nordio, and Nadia Polikarpova. AutoProof: Auto-active functional verification of object-oriented programs. In Christel Baier and Cesare Tinelli, editors, *21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2015)*, volume 9035 of *LNCS*, pages 566–580. Springer, 2015.