SL-COMP: competition of solvers for separation logic

Mihaela Sighireanu

STTT manuscript No.
(will be inserted by the editor)

SL-COMP: Competition of Solvers for Separation Logic

Report on the Third Edition

Mihaela Sighireanu

Received: date / Accepted: date

Abstract SL-COMP is a competition bringing together researchers and users interested in automated reasoning methods for separation logic (SL). The competition provides a snapshot of the state of the art in the area through a set of problems that put forward the strengths and challenges of the existing solvers and a comparative and replicable evaluation of participating solvers. The third edition of SL-COMP took place in April 2019, as part of the TOOLympics event at TACAS 2019. It collected more than 1K satisfiability and entailment problems, had seen the adoption of the new input format based on SMT-LIB and had doubled the number of participant solvers compared with the first edition in 2014. This report relates the history and the context of SL-COMP competition and accounts of its third edition. It also discusses the issues related with its organization and the challenges for the next editions.

Keywords Separation Logic · SAT Modulo Theory · SMT-LIB

1 Introduction

SMT solvers play an important role in the success of verification tools. The SL-COMP competition was inspired by the success of SMT-COMP for solvers on first order theories. It aims at the development and promotion of solvers for separation logic (SL), an established and fairly popular extension of Hoare logic for imperative, heap-manipulating programs [22]. A rather exhaustive list of the past and present tools using separation logic may be found at [21]. An SL-COMP event consists of two stages. The first stage collects satisfiability and entailment problems for SL, translates them into a common format and finally assigns them to categories depending on the fragment of SL they belong to. The problems originate from two sources: the verification tools employing SL or the theoretical studies on separation logic. The later source provides problems corresponding to corner cases (e.g., high complexity cases) in the decidable fragments, or problems in undecidable fragments. The second stage of SL-COMP is the comparative evaluation of registered solvers performed by the organizer. SL-COMP uses StarExec [30], the platform also used by SMT-COMP, to store the benchmark and the binaries of participating solvers as well as to run the experiments until the final run.

The first [26] and second editions of SL-COMP took place as part of the Federated Logic Conferences (FLoC) in 2014 respectivement 2018, and were affiliated to SMT-COMP respectivement Automated Deduction for Separation Logics (ADSL) workshops. The third edition took place as part of TOOLympics event [5] at TACAS 2019. Each event was an opportunity to exchange with researchers interested in SL or with developers of tools based on separation logic.

This paper provides an overview of SL-COMP and highlights the main achievements of the third edition and the open issues for the next editions. Section 2 describes the stages of the competition and the reasons leading to its current organization. The next sections detail the set of problems (Section 3), the running infrastructure (Section 4) and the results of the third edition (Section 5). Section 6 concludes with the achievements of SL-COMP and its perspectives.

Mihaela Sighireanu
IRIF, Université de Paris, France
E-mail: mihaela.sighireanu@irif.fr

2 Competition Organization

2.1 Rise and progress

The competition started in 2014, as an outcome of the emergence of solvers for SL which were independent from the verification tools. These solvers considered mainly the symbolic heap fragment (see Section 3.3) which was successfully used by static analyzers (e.g., SmallFoot [29]) and deductive verification tools (e.g., Hip [10] or VeriFast [18]). The call for problems and tools was addressed first to the teams developing these solvers and then sent to the SMT-COMP community. We collected 600 problems on different fragments of SL from the six participating solvers. The format was an ad hoc extension of the SMT-LIB format. This choice for the input format allowed SL-COMP to be considered an unofficial event associated to SMT-COMP at the FLoC Olympic Games. SL-COMP benefited from the experience of SMT-COMP's organizer, David Cok, in setting the competition's rules and the execution platform StarExec, as well as in running the competition and in the publication of final results. The organization details and the achievements of this edition are presented in [26]. The discussion following this event had two outcomes. First, a working group was charged to improve the input format by taking into account the new features included on the draft of the new version (2.6) of SMT-LIB. Second, the participants chose a sparse rhythm for the competition, roughly aligned with the FLoC's venues, in order to avoid editions with insignificant increments and to promote tools with strong theoretical foundations.

The second edition took place as expected at FLoC 2018 and was associated with the first workshop on Automated Deduction for Separation Logics (ADSL). The competition's organization followed the stages described in the next section and was disconnected from SMT-COMP, but it used the StarExec platform. The call for benchmark was very fruitful: the number of problems was doubled as well as the number of categories. Eleven solvers enrolled in the first round of the competition. The problems have been translated into the new input format (see Section 3.2) which required to update the translators to the input format of each solver. All these novelties were dealt by the teams of the registered solvers and the organizer during the two months before FLoC 2018. This short time was not enough to fix the issues in some newcomer solvers, so one solver abandoned before the final run. The competition was run on all problems (from the 2014 edition and the new ones) several times before an agreement was met on the results. The competition's results [28] have been presented during a full session of the ADSL workshop, and gave the opportunity of a prolific discussion on the different aspects of the organization. The participants agreed to rerun the competition next year as a pre-event of TOOLympics 2019, in order to gain visibility and to fix the identified problems in the benchmark, the scoring and the solvers.

The third edition took place a few days before TACAS 2019. Its organization was planned and described in [27]. The competition considered a slightly modified set of problems. The main changes were fixing the expected results of some problems and changing the division of others. The solver which abandoned at the second edition was enrolled successfully. We tried a new way to score the results and, as an indicator, we nominated a virtually best solver for each job and division. The results have been presented at TACAS and are available on the SL-COMP web site [2]. They are summarized in Section 5. The podium was different from the 2018 edition in several categories since the solvers were improved in the meanwhile. The results pointed out some weaknesses in the organization process, mainly concerning the choice of problems used for scoring and the scoring itself. We discuss these points in Section 5. The next edition of SL-COMP is supposed to take place at FLoC 2022 and its organization will be planned at ADSL 2021.

2.2 Organization process

The competition has a short running period, three months on average. This is possible due to the availability of the material used in the competition (the benchmark set, the definition of the input format, the parsers for the input format and the translators to the input of solvers) on StarExec and on a shared development repository [1] maintained by the organizer and the participants. Starting from its second edition, the organization of the event is discussed during the ADSL workshop of the previous year.

The call for problems and participants launches the competition and fixes the competition timeline. The call is sent on the competition mailing list `sl-comp@googlegroups.com` and on some other mailing lists specialized on automated reasoning.

New solvers are invited to send a short presentation (up to two pages) including the team, the SL-COMP categories joined in, a bibliographic reference and a website. Each team nominates a corresponding person who is responsible for preparing the solver to comply to the competition's constraints. This preparation ensures that the input format is supported, the solver is uploaded on the execution platform and it is registered

at the right categories with the correct configuration. The organizer creates a subspace for each solver on the space SL-COMP of StarExec. The solver's correspondent receives the necessary permissions for this subspace.

The benchmark problems are collected from the community and participants. Until now, we did not limit the number of problems proposed by participants in each category because we targeted the growing of our benchmark set. However, this may change in the future in order to increase the diversity of problems and to avoid fine tuning of tools (see Section 3). A problem may change during the competition if an issue is signaled. However, the benchmark set is fixed starting with the pre-final run.

The competition is run in three stages:

1. The *training period* takes around two weeks during which the solver's correspondent runs the solver on the execution platform and the existing benchmark set. During this step, the benchmark set may evolve if issues are detected; the solver's binary may be also changed.
2. The *pre-final run* is launched by the organizer using the binaries of solvers uploaded on the execution platform. The results of a pre-final run are available for all solvers' representatives, in order to compare results and have a first view on competitors' achievements. The organizer communicates with each correspondent to be sure that the results of this run are accepted. The benchmark set is fixed, but solvers' binaries may change until the acceptance of results.
3. The *final run* takes place one week before the event at which the results are presented. The final results are available as soon as possible on the competition's web site and the podium is presented during the event.

3 Benchmark Set

The benchmark set of SL-COMP contains 1286 problems which cover several fragments of Separation Logic. A quarter of these problems are satisfiability checking problems, the reminder are entailment problems. This section outlines the main features of this benchmark set, including the fragments covered, the input format, and the divisions established for this edition. The input theory and format are formally presented in [17].

3.1 Separation logic theory

The input theory is a multi-sorted second order logic over a signature $\Sigma = (\Sigma^{\mathrm{s}}, \Sigma^{\mathrm{f}})$, where the set of sorts Σ^{s} includes two (non necessarily disjoint) subsets of sorts: $\Sigma^{\mathrm{s}}_{\mathsf{Loc}}$ represents locations of the heap and $\Sigma^{\mathrm{s}}_{\mathsf{Data}}$ represents heap's data. For each sort Loc in $\Sigma^{\mathrm{s}}_{\mathsf{Loc}}$, the set of operations includes a constant symbol $\mathsf{nil}^{\mathsf{Loc}}$ modeling the null location. The heap's type τ is an injection from location sorts in $\Sigma^{\mathrm{s}}_{\mathsf{Loc}}$ to data sorts in $\Sigma^{\mathrm{s}}_{\mathsf{Data}}$. We also assume that the signature Σ includes the Boolean signature and an equality function for each sort.

Let *Vars* be a countable set of first-order variables, each $x^\sigma \in \mathit{Vars}$ having an associated sort σ. The *Ground Separation Logic* SL^g is the set of formulae generated by the following syntax:

$$\varphi := \phi \mid \mathsf{emp} \mid \mathsf{t} \mapsto \mathsf{u} \mid \varphi_1 * \varphi_2 \mid \varphi_1 \mathrel{-\!\!*} \varphi_2 \quad (1)$$
$$\neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \exists x^\sigma \,.\, \varphi_1(x)$$

where ϕ is a Σ-formula, and t, u are Σ-terms of sorts in $\Sigma^{\mathrm{s}}_{\mathsf{Loc}}$ and $\Sigma^{\mathrm{s}}_{\mathsf{Data}}$ respectively, such that they are related by the heap's type τ. We omit specifying the sorts of variables and functions when they are clear from the context.

The special atomic formulas of SL^g are the so-called *spatial atoms*: emp specifies an empty heap, $\mathsf{t} \mapsto \mathsf{u}$ specifies a heap consisting of one allocated cell whose address is t and whose value is u. The operator "$*$" is the separating conjunction denoting that the sub-heaps specified by its operands have disjoint locations. The operator "$\mathrel{-\!\!*}$" is the separating implication operator, also called magic wand. A formula containing only spatial atoms combined using separating conjunction and implication is called *spatial*. Formulas without spatial atoms and separating operators are called *pure*.

The full separation logic SL contains formulas with spatial predicate atoms of the form $P^{\sigma_1 \cdots \sigma_n}(\mathsf{t}_1, \ldots, \mathsf{t}_n)$, where each t_i is a first-order term of sort σ_i, for $i = 1, \ldots, n$. The predicates $P^{\sigma_1 \cdots \sigma_n}$ belong to a finite set $\mathbb{P}$ of second-order variables and have associated a tuple of parameter sorts $\sigma_1, \ldots, \sigma_n \in \Sigma^{\mathrm{s}}$. Second-order variables $P^{\sigma_1 \cdots \sigma_n} \in \mathbb{P}$ are defined using a set of rules of the form:

$$P(x_1, \ldots, x_n) \leftarrow \varphi_P(x_1, \ldots, x_n), \quad (2)$$

where φ_P is a formula possibly containing predicate atoms and having free variables in $x_1, \ldots, x_n$. The semantics of predicate atoms is defined by the least fixed point of the function defined by these rules.

An example of a formula specifying a heap with at least two singly linked list cells at locations x and y is:

$$x \mapsto \mathtt{node}(1, y) * y \mapsto \mathtt{node}(1, z) * \mathtt{ls}(z, \mathsf{nil}) \wedge z \neq x \quad (3)$$

where $\Sigma^{\mathrm{s}} = \{\mathtt{Int}, \mathsf{Loc}, \mathsf{Data}\}$ and the function $\mathtt{node}$ has parameters of sort $\mathtt{Int}$ and Loc and its type is Data. The predicate $\mathtt{ls}$ is defined by the following rules:

$$\mathtt{ls}(h, f) \leftarrow h = f \wedge \mathsf{emp} \quad (4)$$
$$\mathtt{ls}(h, f) \leftarrow \exists x, i \,.\, h \neq f \wedge x \mapsto \mathtt{node}(i, x) * \mathtt{ls}(x, f) \quad (5)$$

and specifies a possible empty heap storing a singly linked list of Data starting at the location denoted by h and whose last cell contains the location denoted by f. More complex examples of formulas and predicate definitions are provided in [26,17].

3.2 Input format

The input format of SL-COMP changed between the first and the second edition, but it was always based on the SMT-LIB format [4]. The first version of the format was built upon the version 2.5 of SMT-LIB and therefore introduced ad hoc constructs for inductive predicate definitions (not allowed in SMT-LIB 2.5) or datatypes; it also had the drawback of introducing a special type for spatial formulas. The new format fixed these issues and was built upon version 2.6 of SMT-LIB. The syntax and the semantics of this format were discussed and approved using the public mailing group.

Signature encoding: Following this format, the new functions of SL theory are declared in a "theory" file `SepLogicTyped.smt2` as follows:

```
(theory SepLogicTyped
 :funs ((emp Bool)
        (sep Bool Bool Bool :left-assoc)
        (wand Bool Bool Bool :right-assoc)
        (par (L D) (pto L D Bool))
        (par (L) (nil L))
        )
)
```

The functions `pto` and `nil` are polymorphic, with sort parameters `L` (for location sort) and `D` (for data sort). There is no restriction on the choice of location and data sorts. However, each problem shall fix them using a special command, not included in SMT-LIB, `declare-heap`. For example, to encode the formula given in Equation 3, we declare an uninterpreted sort `Loc` and a sort `Data` as a datatype as follows:

```
(declare-sort Loc 0)

(declare-datatype Data
                ((node (d Int) (next  Loc))))

(declare-heap (Loc Data))
```

The last declaration fixes the type of the heap model.

The predicate definitions are written into SMT-LIB format using the recursive function definition introduced in version 2.6. For instance, the definition of the list segment from Equations 4 and 5 is written into SMT-LIB as follows (based on the above declarations of `Loc` and `Data`):

```
(define-fun-rec ls ((h Loc) (f Loc)) Bool
    (or (and emp (= h f))
        (exists ((x Loc) (d Int))
                (and (distinct h f)
                     (sep (pto h (node d x))
                          (ls x f))))
    )
)
```

Problem format: Each benchmark file is organized as follows:

- Preamble information required by the SMT-LIB format: the sub-logic of SL theory (see Section 3.3), the team which proposed the problem, the kind (crafted, application, etc.) and the status (sat or unsat) of the problem.
- A list of declarations for the sorts of locations and data, for the type of the heap (the `declare-heap` command), for the second order predicates, and for the free variables used in the problem's formulae. Notice that the input format is strongly typed. At the end of the declarations, a checking command `check-sat` may appear to trigger for some solvers the checking for models of predicate declarations.
- One or two assertions (command `assert`) introducing the formulas used in the satisfiability respectively entailment problem.
- The file ends with a checking satisfiability command `check-sat`. Notice that checking the validity of the entailment $A \Rightarrow B$ is encoded by satisfiability checking of its negation $A \wedge \neg B$.

3.3 Divisions

The main difficulty that faces automatic reasoning using SL is that the logic, due to its expressiveness, does not have very nice decidability properties [3]. For this reason, most solvers use incomplete heuristics to solve the satisfiability and entailment problems in SL or restrict the logic employed to decidable fragments. Overviews of decidable results for SL are available in [26,11].

Each problem of SL-COMP's benchmark refers to one of the sub-logics of the multi-sorted Separation Logic. These sub-logics identify fragments which are handled by at least two participants or have been identified to be of interest by the jury (organizer with solvers's representatives).

The sub-logics are named using groups of letters, in a way similar to the SMT-LIB format. These letters have been chosen to evoke the restrictions used by the sub-logics:

Table 1 Divisions at SL-COMP and the participants enrolled

Division	*size*	*Solvers enrolled*
qf_bsl_sat	46	CVC4-SL
qf_bsllia_sat	24	CVC4-SL
qf_shid_entl	312	Cyclist-SL, Harrsh, S2S, Sleek, Slide, Songbird, Spen
qf_shid_sat	99	Harrsh, S2S, Sleek, SlSat
qf_shidlia_entl	75	ComSPEN, S2S
qf_shidlia_sat	33	ComSPEN, S2S
qf_shlid_entl	60	ComSPEN, Cyclist-SL, Harrsh, S2S, Spen
qf_shls_entl	296	Asterix, ComSPEN, Cyclist-SL, Harrsh, S2S, Spen
qf_shls_sat	110	Asterix, ComSPEN, Cyclist-SL, Harrsh, S2S, Spen
shid_entl	73	Cyclist-SL, S2S, Sleek, Songbird
shidlia_entl	181	S2S, Songbird

- QF for the restriction to quantifier free formulas;
- SH for the "symbolic heap fragment" where formulas are restricted to (Boolean and separating) conjunctions of atoms and do not contain magic wand; moreover, pure atoms are only equality or dis-equality atoms;
- LS where the only predicate allowed is the acyclic list segment, ls, defined in Equations 4 and 5;
- ID for the fragment with user defined predicates;
- LID for the fragment allowing linear definitions of user predicates, i.e., only one recursive call for all rules of a predicate is allowed;
- B for the ground fragment allowing any Boolean combination of atoms.

Moreover, the existing fragments defined in SMT-LIB are used to further restrict the theory signature. For example, LIA denotes the signature for linear integer arithmetic.

Currently, the competition has eleven categories, called divisions and named by the concatenation of the logic's name with the kind of problem solved (sat or entl). Table 1 provides the names of these divisions, their size and the solvers enrolled:

- qf_bsl_sat and qf_bsllia_sat divisions include satisfiability problems for quantifier free formulas in the ground logic using respectively none or LIA logic for pure formulas.
- qf_shid_entl and qf_shid_sat divisions include entailment respectively satisfiability problems for the symbolic heap fragment with user defined predicates. The fragment is undecidable for general predicate definitions, but restrictions on these definitions lead to decidability [15].
- qf_shidlia_entl and qf_shidlia_sat divisions include entailment respectively satisfiability problems for the quantifier free, symbolic heap fragment with user defined predicates and linear arithmetic included in pure formulas even in the predicate definitions.
- qf_shlid_entl division includes a subset of problems of division qf_shid_entl where the predicate definitions are linear and compositional [13]. This fragment is of interest because the entailment problem is decidable and has lower complexity.
- qf_shls_entl and qf_shls_sat divisions include entailment respectively satisfiability problems for the quantifier free symbolic heap fragment with only singly linked list predicate atoms. The inductive predicate, called ls, is defined as above but without integer data in each cell.
- shid_entl division contains entailment problems for quantified formulas in the symbolic heap fragment with general predicate definitions and no other logic theories than Boolean.
- shidlia_entl divisions extends the problems in shid_entl with constraints from linear integer arithmetic.

Table 2 gives the contribution of each solver to the benchmark set of the third edition. The changes done on this set may be tracked in the SL-COMP repository [1]. For the third edition, these changes concern the expected status of some problems (5%) and their division (2%). They have been done at the request of participants and after the validation of the organizer.

4 Running the Third Edition

4.1 Participation

The third edition brought together eleven solvers, like in the second edition, but all passed the final round. A detailed presentation of each solver may be found in [27] and on their web site which is reachable from the competition's web site [2]. The binaries of these solvers are available on StarExec. Table 2 provides an overview of the participating solvers and lists the features and technologies which they are using, as well as their contribution to the benchmark set.

Table 2 Details on participating solvers and their contribution to the benchmark set

Solver	*Ref.*	*Representative*	Contribution	Reduction to SMT	Abstraction based	Model-based	Theorem proving	Lemma predefined	Lemma synthesis	Graph isomorphism	Heap automata	Tree automata
Asterix	[23]	Juan Navarro Perez	30%	✓		✓						
ComSPEN	[14]	Chong Gao	3%	✓	✓					✓		
Cyclist-SL	[8]	Nikos Gorogiannis	6%				✓					
CVC4	[24]	Andrew Reynolds	5%	✓								
Harrsh	[19]	Jens Katelaan	2%								✓	
S2S	[25]	Quang Loc Le	8%	✓			✓		✓			
Sleek	[10]	Benedict Lee	6%	✓			✓		✓			
Slide	[16]	Adam Rogalewicz	2%									✓
SlSat	[6]	Nikos Gorogiannis	–	✓	✓							
Songbird	[31]	Quang-Trung Ta	31%	✓			✓	✓				
Spen	[12]	Mihaela Sighireanu	7%	✓				✓		✓		✓

Concerning the techniques employed, observe that most of the solvers combine several techniques. For the decidable fragments, the techniques based on reduction to SMT, graph isomorphism and tree automata are predominant. For the fragments including general inductive definitions of predicates, the solvers employ techniques coming from the mechanized proofs domain enriched with heuristics for lemma synthesis.

Most participants of this edition contributed to the benchmark set, but two teams distinguish themselves. The Asterix team provided most of the problems ($>$ 95%) in the divisions `qf_shls_entl` and `qf_shls_sat`. These problems have their origin in the queries generated by the SmallFoot [29] analyzer or have been crafted in a random way. The Songbird team provided most of problems in the divisions `qf_shid_entl` ($>$ 40%) and `shidlia_entl` ($>$ 95%), originating from the verification conditions of algorithms on data structures. This massive participation to the benchmark set influenced the podium at the second edition since Asterix maintained its first place and Songbird won the first place in the division `shidlia_entl` and the second place in the division `qf_shid_entl`. The origin of the problem lost its importance at the third edition because the S2S solver won the first or the second place for all categories.

The binaries used for the solvers evolved during this edition, mainly to fix the parsing of the new input format and to deal with bugs revealed by the new problems. It is worth noticing that these new problems have been proposed by teams which studied the binary and the public code repository of their competitors. This is an interesting effect of making available the binary and the code of solvers. Three solvers used the binary stored on StarExec at the first edition of SL-COMP: Asterix Cyclist-SL and SlSat.

One solver, CVC4, was the only participant in its divisions. In the second edition of SL-COMP, it had one competitor which was pulled out before the final run and did not join the third edition. The organizer maintained these divisions to encourage the development of solvers for these fragments and to keep track of CVC4's performances at this edition.

4.2 Training period

This period of the competition offers the opportunity to interact closely with the solvers' teams. In the third edition, the training was used to fix some benchmarks, to adapt some rules and to find an agreement on the scoring scheme. This is possible due to the fact that the SL-COMP community is still small. Having flexible rules stimulates the participation and keeps a collegial atmosphere.

The corresponding person has to ask for a sub-space in the SL-COMP space on the StarExec platform. The SL-COMP space has a subspace for each edition which contains a subspace with an up-to-date version of the benchmark, a subspace for the binaries corresponding to the latest version of each solver, and several examples of configuring StarExec tasks, i.e., execution of solvers on benchmark's categories.

The organizer applies for resources to the StarExec manager. For the first edition, the training was done with few resources, during the off-peak hours of SMT-COMP. For the second edition, SL-COMP had used six StarExec nodes for the training and the competition. Given the increase in the number of problems

and solvers, the third edition used ten nodes. These resources were enough even during the training period when several solvers were tested. For this reason, the organizer increased the memory limit for each task (pair of solver and problem) from 4 GB used at the second edition to 10 GB. This amount of memory was requested by one solver using JVM; however, the solvers compiled to native code are using less than 4 GB. The timeout for each task ranged from 600 to 2400 seconds, depending on the timeout rate experienced in each division during the training period.

4.3 Scoring scheme

The third edition adopted a score-based system to designate the best solver in each division. In the previous editions, the nomination of the winner was based on the scoring scheme of SMT-COMP, i.e., the best solver was the one with, in order: (a) the least number of incorrect answers, (b) the largest number of correctly solved problems, and (c) the shortest time taken in solving the correctly solved problems. We changed this system for the third edition in order to limit the penalty for sound but imprecise results (false positives). For each division, the score was computed by:

$$10 \cdot \textit{solved} - \textit{false-positive} - 10 \cdot \textit{false-negative} \quad (6)$$

where a *false-positive* means a problem solved with result "sat" instead of the expected "unsat" (this result is sound for program verification but not precise), a *false-negative* denotes a problem solved with result "unsat" instead of the expected "sat" (this result is not sound for program verification). The CPU time is the tiebreaker. The scoring scheme was discussed with the corresponding persons during the training period. Such a scheme deals uniformly with entailment and satisfiability problems, which may be unfair if the satisfiability problem is used in the context of symbolic execution or program testing. This point should be considered for the next editions.

In addition to the score, the *Virtually Best Solver* contribution (VBS, the solver which would be the best for the division) was computed by taking the minimum time for each correctly solved problem over all solvers. The contribution of each solver to VBS is the number of problems where the solver is the fastest. However, the VBS was given only as an indicative information and not used for the podium.

4.4 Pre-final and final runs

The pre-final stage took nine days for the third edition compared with four days for the second edition which used a similar number of problems and benchmarks. More pre-final runs were needed to converge to the final result because of frequent changes in the solver's binaries and configurations. This is still possible with eleven solvers, but should be regulated if the size of the benchmark or the number of participants increase.

The final run took two days to be run by the organizer. After that, the organizer produced the official results. Unfortunately, the StarExec platform did not provide means to automatically extract the results from the CSV files given a scoring scheme. It could be interesting to share such tools. The ones used for SL-COMP are available on the GIT repository [1]. The final results were inspected and approved by the participating teams before the presentation at TOOLympics.

5 Results and Discussion

For each division, the best solver obtained five stars, the fifth one received one star. The global podium was computed by sorting the solvers in the decreasing order of the total number of stars obtained in all divisions.

Table 3 presents the podium (from first to 5th place) for each division. For some divisions, the podium was not completely occupied either by lack of participants or because some solvers obtained negative scores. The table also presents the final podium computed from the results of all divisions. The website of the competition [2] provides detailed results for each division and solver in the form of a CSV file.

The results revealed the domination of the S2S solver. It participated in 9 divisions (over a total of 11), was able to solve all the problems without errors and with the best score in 7 over 9 divisions. These results are very different from the first and the second edition, where the podium varied between divisions. To avoid the feeling of a special tuning of the winner for the current benchmark, the rules of the competition should include a new way of choosing the set of problems used for evaluation. Several competitions use such rules consisting of a mixture of problems from the public benchmark scrambled or not and problems chosen by the jury for the final run. Another possibility is to ask solvers to produce a certificate of the result as a list of proof rules applied by the solver, like it is proposed for SMT-COMP. This certificate may be checked off-line using proof assistants like Coq.

Looking at the rest of the podium, we observe a general improvement of results obtained compared with the second edition. More problems were solved in less time. However, some divisions remain difficult to solve for most of solvers. This is not surprising be-

Table 3 Overall podium and podium for each division from first to fifth place

Participant	*Podium*	qf_bsl_sat	qf_bsllia_sat	qf_shid_entl	qf_shid_sat	qf_shidlia_entl	qf_shidlia_sat	qf_shlid_entl	qf_shls_entl	qf_shls_sat	shid_entl	shidlia_entl
Asterix									1	1		
ComSPEN	Bronze					3	2		5	3		
Cyclist-SL				4				4			3	
CVC4		1	1									
Harrsh				3	4			3				
S2S	Gold			1	1	1	1	1	2	2	1	1
Sleek				5	2		4	5		5		
Slide											4	
SlSat					3							
Songbird	Silver			2	5	2	3	2	4		2	2
Spen									3	4		

cause these divisions correspond to undecidable fragments. In the divisions corresponding to decidable fragments, most solvers produce correct results in relatively short time. Two newcomers obtained good general results, ComSPEN and Songbird. Notice that Asterix maintains its first place in the `qf_shls` divisions from the first edition.

6 Conclusion and Perspectives

The third edition of SL-COMP achieved the following objectives:

1. It attracted eleven solvers (2014: 6 solvers, 2018: 10 solvers) from seven countries.
2. It provided a high-performance view of SL solvers.
3. The benchmark set was improved and reached more than 1K of problems (600 in 2014). The repository of problems is publicly available [1] for free use as standard benchmark suite for evaluating solvers.
4. The input format for problems proposed in [17] was largely adopted by the competing solvers. It is now a standard for submitting problems to SL-COMP.
5. The competition won visibility and the researchers interested in developing SL tools send us positive feedback.

For the next edition, planed for 2022, several points have to be improved in the organization of SL-COMP. Firstly, the rules shall be enforced to obtain a fair and transparent evaluation. During the last edition, we identified two sources of bias: the lack of diversity in the origin of the benchmark's problems and an evaluation process that allows the fine-tuning of solvers on the existing benchmark. The scoring scheme should also be changed take into account the kind of problems dealt with (satisfiability or entailments). Secondly, the period between two events should be used to maintain the link between competitors and a collegial atmosphere. The annual workshop ADSL may be an opportunity to discuss the issues related with the competition's rules and content. A special session could be dedicated to present the participating solvers or a new benchmark set. Finally, the increasing number of participants and problems requires more automatization for the tasks allocated to the jury, for example the problem classification, the validation of results and their presentation. Appropriate tools could be developed for such tasks. Apart from these improvements, SL-COMP should continue to enrich its benchmark with problems coming from the fragments shown to have decidable satisfiability and entailment problems. One of these fragments, which has also an interesting application in program analysis [9], is the separation logic with pointer arithmetic and arrays [7,20]

Acknowledgements The author thanks the representative of each participating solver for their willing collaboration during the running of the competition, especially Andrew Reynolds, Jens Katelaan, Le Quang Loc, Benedict Lee, Quang-Trung Ta, Adam Rogalewicz, Chong Gao and Zhilin Wu. Cristina Şerban provided the first version of the parser and the typechecker for the new format. The reviewers provided interesting comments and suggestions for the improvement of this paper. This work was partially supported by the ANR project CoLiS, contract number ANR-15-CE25-0001.

References

1. SL-COMP repository. https://github.com/sl-comp
2. SL-COMP website. https://sl-comp.github.io/
3. Antonopoulos, T., Gorogiannis, N., Haase, C., Kanovich, M.I., Ouaknine, J.: Foundations for decision problems in separation logic with general inductive predicates. In: FOSSACS, *LNCS*, vol. 8412, pp. 411–425. Springer (2014). DOI: 10.1007/978-3-642-54830-7_27
4. Barrett, C., Fontaine, P., Tinelli, C.: The SMT-LIB Standard: Version 2.6. Tech. rep., Department of Computer Science, The University of Iowa (2017). URL www.SMT-LIB.org
5. Beyer, D., Huisman, M., Kordon, F., Steffen, B. (eds.): Tools and Algorithms for the Construction and Analysis of Systems - 25 Years of TACAS: TOOLympics, Held as Part of ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part III, *LNCS*, vol. 11429. Springer (2019). DOI: 10.1007/978-3-030-17502-3
6. Brotherston, J., Fuhs, C., Pérez, J.A.N., Gorogiannis, N.: A decision procedure for satisfiability in separation logic with inductive predicates. In: CSL-LICS, pp. 25:1–25:10. ACM (2014). DOI: 10.1145/2603088.2603091
7. Brotherston, J., Gorogiannis, N., Kanovich, M.: Biabduction (and related problems) in array separation logic. In: CADE, vol. 10395, pp. 472–490. Springer (2017). DOI: 10.1007/978-3-319-63046-5_29
8. Brotherston, J., Gorogiannis, N., Petersen, R.L.: A generic cyclic theorem prover. In: APLAS, *LNCS*, vol. 7705, pp. 350–367. Springer (2012). DOI: 10.1007/978-3-642-35182-2_25
9. Calcagno, C., Distefano, D., O'Hearn, P.W., Yang, H.: Beyond Reachability: Shape Abstraction in the Presence of Pointer Arithmetic. In: SAS, *LNCS*, vol. 4134, pp. 182–203. Springer (2006). DOI: 10.1007/11823230_13
10. Chin, W.N., David, C., Nguyen, H.H., Qin, S.: Automated verification of shape, size and bag properties via user-defined predicates in separation logic. Science of Computer Programming **77**(9), 1006–1036 (2012). DOI: 10.1016/j.scico.2010.07.004
11. Demri, S., Deters, M.: Separation logics and modalities: a survey. Journal of Applied Non-Classical Logics **25**(1), 50–99 (2015). DOI: 10.1080/11663081.2015.1018801
12. Enea, C., Lengál, O., Sighireanu, M., Vojnar, T.: Compositional entailment checking for a fragment of separation logic. In: APLAS, *LNCS*, vol. 8858, pp. 314–333. Springer (2014). DOI: 10.1007/978-3-319-12736-1_17
13. Enea, C., Sighireanu, M., Wu, Z.: On automated lemma generation for separation logic with inductive definitions. In: ATVA, *LNCS*, vol. 9364, pp. 80–96. Springer (2015). DOI: 10.1007/978-3-319-24953-7_7
14. Gu, X., Chen, T., Wu, Z.: A complete decision procedure for linearly compositional separation logic with data constraints. In: IJCAR, *LNCS*, vol. 9706, pp. 532–549. Springer (2016). DOI: 10.1007/978-3-319-40229-1_36
15. Iosif, R., Rogalewicz, A., Simácek, J.: The tree width of separation logic with recursive definitions. In: CADE, *LNCS*, vol. 7898, pp. 21–38. Springer (2013). DOI: 10.1007/978-3-642-38574-2_2
16. Iosif, R., Rogalewicz, A., Vojnar, T.: Deciding entailments in inductive separation logic with tree automata. In: ATVA, *LNCS*, vol. 8837, pp. 201–218. Springer (2014). DOI: 10.1007/978-3-319-11936-6_15
17. Iosif, R., Serban, C., Reynolds, A., Sighireanu, M.: Encoding separation logic in SMT-LIB v2.5. https://github.com/sl-comp/SL-COMP18/input/Docs (2018)
18. Jacobs, B., Smans, J., Piessens, F.: A quick tour of the VeriFast program verifier. In: APLAS, *LNCS*, vol. 6461, pp. 304–311. Springer (2010). DOI: 10.1007/978-3-642-17164-2_21
19. Katelaan, J., Matheja, C., Noll, T., Zuleger, F.: Harrsh: A tool for unified reasoning about symbolic-heap separation logic. In: LPAR-22, *Kalpa Publications in Computing*, vol. 9, pp. 23–36. EasyChair (2018). DOI: 10.29007/qwd8
20. Kimura, D., Tatsuta, M.: Decidability for entailments of symbolic heaps with arrays. CoRR **abs/1802.05935** (2018). URL http://arxiv.org/abs/1802.05935
21. O'Hearn, P.: Separation logic. http://www0.cs.ucl.ac.uk/staff/p.ohearn/SeparationLogic/Separation_Logic/SL_Home.html
22. O'Hearn, P.: Separation logic. Commun. ACM **62**(2), 86–95 (2019). DOI: 10.1145/3211968
23. Pérez, J.A.N., Rybalchenko, A.: Separation logic modulo theories. In: APLAS, *LNCS*, vol. 8301, pp. 90–106. Springer (2013). DOI: 10.1007/978-3-319-03542-0_7
24. Reynolds, A., Iosif, R., Serban, C., King, T.: A decision procedure for separation logic in SMT. In: ATVA, pp. 244–261 (2016). DOI: 10.1007/978-3-319-46520-3_16
25. S2S: https://loc.bitbucket.io/s2s/
26. Sighireanu, M., Cok, D.: Report on SL-COMP'14. JSAT **9**, 173–186 (2014). DOI: 10.3233/SAT190107
27. Sighireanu, M., Pérez, J.A.N., Rybalchenko, A., Gorogiannis, N., Iosif, R., Reynolds, A., Serban, C., Katelaan, J., Matheja, C., Noll, T., Zuleger, F., Chin, W., Le, Q.L., Ta, Q., Le, T., Nguyen, T., Khoo, S., Cyprian, M., Rogalewicz, A., Vojnar, T., Enea, C., Lengál, O., Gao, C., Wu, Z.: SL-COMP: competition of solvers for separation logic. In: Beyer et al. [5], pp. 116–132. DOI: 10.1007/978-3-030-17502-3_8
28. SL-COMP'2018: https://www.irif.fr/~sighirea/sl-comp/18/
29. SmallFoot: http://www0.cs.ucl.ac.uk/staff/p.ohearn/smallfoot/
30. StarExec: http://www.starexec.org
31. Ta, Q.T., Le, T.C., Khoo, S.C., Chin, W.N.: Automated lemma synthesis in symbolic-heap separation logic. Proc. ACM Program. Lang. **2**(POPL), 9:1–9:29 (2017). DOI: 10.1145/3158097