Design and Results of the Second International Competition on Computational Models of Argumentation

Sarah A. Gaggl[a], Thomas Linsbichler[b], Marco Maratea[c,*], Stefan Woltran[b]

[a] *Faculty of Computer Science, TU Dresden, Germany*
[b] *Faculty of Informatics, TU Wien, Austria*
[c] *Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi, Università di Genova, Italy*

Abstract

Argumentation is a major topic in the study of Artificial Intelligence. Since the first edition in 2015, advancements in solving (abstract) argumentation frameworks are assessed in competition events, similar to other closely related problem solving technologies. In this paper, we report about the design and results of the Second International Competition on Computational Models of Argumentation, which has been jointly organized by TU Dresden (Germany), TU Wien (Austria), and the University of Genova (Italy), in affiliation with the 2017 International Workshop on Theory and Applications of Formal Argumentation. This second edition maintains some of the design choices made in the first event, e.g. the I/O formats, the basic reasoning problems, and the organization into tasks and tracks. At the same time, it introduces significant novelties, e.g. three additional prominent semantics, and an instance selection stage for classifying instances according to their empirical hardness.

Keywords: Abstract Argumentation, Solver Competition, Computational Logic

[☆] This paper is an extended and revised version of a paper presented at the First International Workshop on Systems and Algorithms for Formal Argumentation (Gaggl et al., 2016), which included the design of the event before the competition was run. A brief survey of the competition is to be published in AI Magazine (Gaggl et al., 2018).

[*] Corresponding author

Email addresses: sarah.gaggl@tu-dresden.de (Sarah A. Gaggl), linsbich@dbai.tuwien.ac.at (Thomas Linsbichler), marco@dibris.unige.it (Marco Maratea), woltran@dbai.tuwien.ac.at (Stefan Woltran)

1. Introduction

Computational Argumentation is a multidisciplinary area at the intersection of Philosophy, Artificial Intelligence (AI), Linguistics, Psychology, and several application domains (Bench-Capon and Dunne, 2007). Within AI, several subfields are particularly relevant to – and benefit from – studies of argumentation. These include decision support, knowledge representation, nonmonotonic reasoning, and multiagent systems. Moreover, computational argumentation provides a formal investigation of problems that have been studied informally only by philosophers, and which consequently allow for the development of computational tools for argumentation, see (Atkinson et al., 2017).

Since its invention by Dung (1995), abstract argumentation based on argumentation frameworks (AFs) has become a key concept for the field. In AFs, argumentation scenarios are modeled as simple directed graphs, where the vertices represent arguments and each edge corresponds to an attack between two arguments. Besides its simplicity, there are several reasons for the success story of this concept: First, a multitude of semantics (Baroni et al., 2011, 2018) allows for tight coupling of argumentation with existing formalisms from the areas of knowledge representation and logic programming; indeed, one of the main motivations of Dung's work (Dung, 1995) was to give a uniform representation of several nonmonotonic formalisms including Reiter's Default Logic, Pollock's Defeasible Logic, and Logic Programming (LP) with default negation; the latter lead to a series of works that investigated the relationship between different LP semantics and different AF semantics, see e.g. (Wu et al., 2009; Caminada et al., 2015). Second, abstract argumentation is employed as a core method in advanced argumentation formalisms like ASPIC+ (Modgil and Prakken, 2014) or the ABA framework (Cyras et al., 2018); in particular, semantics for such formalisms are often defined via a representation that makes use of AFs, and moreover, some of the systems implementing ASPIC+ or ABA rely on efficient solvers for abstract argumentation. Consequently, an increasing amount of work has been focused on the development of efficient algorithms and systems for AFs, see (Charwat et al., 2015) for a survey.

Given this development, it was soon recognized that there is a need for systematic benchmarking in order to have a solid comparison of the different methods and systems that have been proposed. This is witnessed by a number of papers on the topic, e.g. (Bistarelli et al., 2015; Cerutti et al., 2016b; Bistarelli et al.,

2018; Vallati et al., 2018) and cumulated in the creation and organization of the International Competition on Computational Models of Argumentation (ICCMA). The first edition took place in 2015 and focused on four prominent semantics; 18 solvers were competing in this event, see (Thimm et al., 2016; Thimm and Villata, 2017) for details.

In this report, we present the design and results of the Second International Competition on Computational Models of Argumentation (ICCMA'17)[1], which has been jointly organized by TU Dresden (Germany), TU Wien (Austria), and the University of Genova (Italy), in affiliation with the 2017 International Workshop on Theory and Applications of Formal Argumentation (TAFA'17). ICCMA'17 has been conducted in the first half of 2017, and comes two years after the first edition.

The general goal of this competition is to consolidate and strengthen the ICCMA series, which in its first edition had very good outcomes in some respects, e.g. in terms of the number of submitted solvers (18, as already mentioned above). The second edition maintains some of the design choices previously made, e.g. the I/O formats and the basic reasoning problems. With a slight modification to the first edition, the competition is organized into *tasks* and *tracks*, where a *task* is a reasoning problem under a particular semantics, and a *track* collects different tasks over a semantics. ICCMA'17 also introduces several novelties: (i) a new scoring scheme is implemented for better reflecting the solvers' behavior, (ii) three new semantics are included, namely semi-stable, stage and ideal semantics, (iii) a special "Dung's Triathlon" track is added, where solvers are required to deal with different problems simultaneously, with the goal of testing the solvers' capability of exploiting interrelationships among semantics, and (iv) a "call for benchmarks" has been performed, to enrich the suite of instances for the competition, followed by a novel instance selection stage.

In addition to the report of the competition, we also compare in this article the performance of the ICCMA'15 winning systems to the current leaders.

Besides its importance for the argumentation community, the ICCMA series is also of interest for researchers beyond this field. This is due to the following two reasons:

- Solvers need to handle a variety of different semantics which range over different levels of complexity; in ICCMA'17 we put even more emphasis on this rather unique feature by the introduction of the Dung's triathlon,

[1] `http://argumentationcompetition.org/2017/`

where the systems are required to solve problems situated at three different complexity layers, preferably exploiting interrelationships between these problems. (We note that problems of different complexity are also present in other competitions, e.g. in Quantified Satisfiability (QBF) or in Answer Set Programming (ASP) competitions, see (Pulina, 2016; Calimeri et al., 2016; Gebser et al., 2017)); however, the situation is more challenging in argumentation since the diverse complexity actually stems from the different semantics which require different computational tasks including subset-maximization, fixed-point computations, etc.)

- Given the range of submitted solvers, we see a great variety of approaches. In particular, various methods including (different forms of) reductions to SAT, ASP, constraint satisfaction, and circumscription are employed in the submitted systems. Thus, ICCMA also provides (to a certain extent) an interdisciplinary comparison between different reasoning paradigms in AI.[2].

The report is structured as follows. Section 2 introduces preliminaries about abstract argumentation, with focus on the semantics evaluated in the competition. Then, Section 3 presents the design of the competition. Section 4 and 5 are devoted to the description of the benchmark suite employed in the competition, and the instance selection process, respectively. Section 6 then presents the participating solvers. The results of the competition, with respective award winners, are then presented in Section 7. The report ends in Section 8 with a discussion on how the novelties introduced are treated in related competitions, and in Section 9 with conclusions and final remarks.

2. Background

An *abstract argumentation framework* (AF, for short) is a tuple $F = (\mathsf{A}, \rightarrow)$ where A is a set of arguments and $\rightarrow$ is a relation $\rightarrow \subseteq \mathsf{A} \times \mathsf{A}$ (Dung, 1995). For two arguments $a, b \in \mathsf{A}$ the relation $a \rightarrow b$ means that argument *a attacks* argument b. An argument $a \in \mathsf{A}$ is *defended* by $S \subseteq \mathsf{A}$ (in F) if for each $b \in \mathsf{A}$ such that $b \rightarrow a$ there is some $c \in S$ such that $c \rightarrow b$. A set $E \subseteq \mathsf{A}$ is *conflict-free* (in F) if and only if there are no $a, b \in E$ with $a \rightarrow b$. E is *admissible* (in F) if and only if it is

[2]It has to be mentioned that this not a completely new phenomenon. For instances, SAT-based approaches competed in ASP competitions, see, e.g. (Giunchiglia et al., 2006), and likewise, an ASP-based approach for 2-QBF solving participated (Amendola et al., 2016) to the 2016 QBF evaluation.

conflict-free and each $a \in E$ is defended by E. Finally, the range of E (in F) is given by $E_F^+ = E \cup \{a \in \mathsf{A} \mid \exists b \in E : b \to a\}$.

Semantics are used to determine sets of jointly acceptable arguments by mapping each AF $F = (\mathsf{A}, \to)$ to a set of *extensions* $\sigma(F) \subseteq 2^{\mathsf{A}}$. The extensions under complete (**CO**), preferred (**PR**), stable (**ST**), semi-stable (**SST**) (Caminada et al., 2012), stage (**STG**) (Verheij, 1996), grounded (**GR**) and ideal (**ID**) (Dung et al., 2007) semantics are defined as follows. Given an AF $F = (\mathsf{A}, \to)$ and a set $E \subseteq \mathsf{A}$,

- $E \in \mathbf{CO}(F)$ iff E is admissible in F and if $a \in \mathsf{A}$ is defended by E then $a \in E$,
- $E \in \mathbf{PR}(F)$ iff $E \in \mathbf{CO}(F)$ and there is no $E' \in \mathbf{CO}(F)$ s.t. $E' \supset E$,
- $E \in \mathbf{ST}(F)$ iff $E \in \mathbf{CO}(F)$ and $E_F^+ = \mathsf{A}$,
- $E \in \mathbf{SST}(F)$ iff $E \in \mathbf{CO}(F)$ and there is no $E' \in \mathbf{CO}(F)$ s.t. $E_F'^+ \supset E_F^+$,
- $E \in \mathbf{STG}(F)$ iff E is conflict-free in F and there is no E' such that E' is conflict-free in F and $E_F'^+ \supset E_F^+$,
- $E \in \mathbf{GR}(F)$ iff $E \in \mathbf{CO}(F)$ and there is no $E' \in \mathbf{CO}(F)$ s.t. $E' \subset E$,
- $E \in \mathbf{ID}(F)$ iff E is admissible in F, $E \subseteq \bigcap \mathbf{PR}(F)$ and there is no $E' \subseteq \bigcap \mathbf{PR}(F)$ s.t. E' is admissible in F and $E' \supset E$,

For more discussion on these semantics we refer to Baroni et al. (2011).

Note that both grounded and ideal extensions are uniquely determined and always exist (Dung, 1995; Dung et al., 2007). Thus, they are also called *single-status* semantics. The other semantics introduced are *multi-status* semantics. That is, there is not always a unique extension induced by the semantics. For all semantics except stable semantics, there always exists at least one extension, whereas the set of stable extensions can be empty. If the set of stable extensions is non-empty, it coincides with the set of semi-stable extensions and with the set of stage extensions, i.e. $\mathbf{ST}(F) = \mathbf{SST}(F) = \mathbf{STG}(F)$ whenever $\mathbf{ST}(F) \neq \emptyset$.

Example 1. To illustrate the semantics, consider the following AF:

$$F = (\{a,b,c,d,e,f,g,h\},\\ \{(a,b),(b,a),(b,c),(c,d),(d,e),(d,g),(e,c),(e,f),(f,f),(g,g),(g,h),(h,g)\}).$$

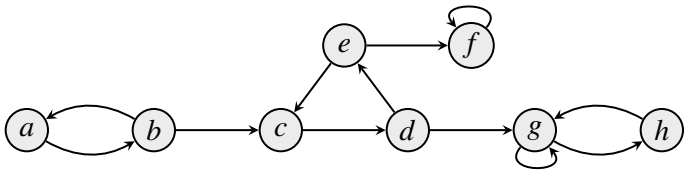

Figure 1: An argumentation framework.

F is depicted in Figure 1, where nodes represent arguments and directed edges represent attacks. First, the conflict-free sets of F are as follows:

$$\{\emptyset,\{a\},\{b\},\{c\},\{d\},\{e\},\{h\},\{a,c\},\{a,d\},\{a,e\},\{a,h\},\{b,d\},\{b,e\},$$
$$\{b,h\},\{c,h\},\{d,h\},\{e,h\},\{a,c,h\},\{a,d,h\},\{a,e,h\},\{b,d,h\},\{b,e,h\}\}.$$

Note that no set containing f or g can be conflict-free, since both f and g are self-attacking. Among the conflict-free sets, the following sets are admissible:

$$\{\emptyset,\{a\},\{b\},\{h\},\{a,h\},\{b,d\},\{b,h\},\{b,d,h\}\}.$$

The conflict-free set $\{a,d\}$, for instance, is not admissible since d is attacked by c in F, but $\{a,d\}$ does not attack c, i.e. it does not defend d.

For stable semantics, it can be checked that there is no conflict-free set of arguments in F attacking all other arguments, hence:

$$\mathbf{ST}(F) = \emptyset.$$

The complete extensions of F are those admissible sets which do not defend any argument not contained in the set:

$$\mathbf{CO}(F) = \{\emptyset,\{a\},\{h\},\{a,h\},\{b,d,h\}\}.$$

For instance, the admissible set $\{b,d\}$ is not complete since it defends h. As no argument of F is unattacked, the grounded extension is empty:

$$\mathbf{GR}(F) = \{\emptyset\}.$$

The preferred extensions are just the $\subseteq$-maximal admissible sets, which always coincide with the $\subseteq$-maximal complete extensions:

$$\mathbf{PR}(F) = \{\{a,h\},\{b,d,h\}\}.$$

The semi-stable and stage extensions of F are given as follows:

$$\mathbf{SST}(F) = \{\{b,d,h\}\}.$$
$$\mathbf{STG}(F) = \{\{a,e,h\},\{b,e,h\},\{b,d,h\}\}.$$

Finally, $\{h\} = \bigcap \mathbf{PR}(F)$ and $\{h\}$ is admissible, hence

$$\mathbf{ID}(F) = \{\{h\}\}.$$

In order to reason with multi-status semantics, usually, one takes either a credulous or skeptical perspective.

Given a semantics[3] $\sigma \in \{\mathbf{CO}, \mathbf{PR}, \mathbf{ST}, \mathbf{SST}, \mathbf{STG}, \mathbf{GR}, \mathbf{ID}\}$, we thus define the following decision problems:

- $Cred_\sigma$: Given an AF $F = (\mathsf{A}, \rightarrow)$ and argument $a \in \mathsf{A}$, a is *credulously accepted* in F under semantics σ if there is a σ-extension $E \in \sigma(F)$ with $a \in E$;

- $Skept_\sigma$: Given an AF $F = (\mathsf{A}, \rightarrow)$ and argument $a \in \mathsf{A}$, a is *skeptically accepted* in F with semantics σ if for all σ-extensions $E \in \sigma(F)$ it holds that $a \in E$.

Recall that stable semantics is the only case where an AF might possess no extension. In such a situation, each argument is defined to be skeptically accepted.

Further reasoning problems for any semantics σ are defined as follows:

- Ver_σ: Given an AF $F = (\mathsf{A}, \rightarrow)$ and a set of arguments $S \subseteq \mathsf{A}$, decide whether $S \in \sigma(F)$.

- $Exists_\sigma$: Given an AF $F = (\mathsf{A}, \rightarrow)$, decide whether there exists an $S \in \sigma(F)$.

- $Exists_\sigma^{\neg\emptyset}$: Given an AF $F = (\mathsf{A}, \rightarrow)$, decide whether there exists an $S \in \sigma(F)$ with $S \neq \emptyset$.

- $Enum_\sigma$: Given an AF $F = (\mathsf{A}, \rightarrow)$, enumerate the set $\sigma(F)$.

[3] For the sake of uniformity, we include here also the single-status semantics **GR**, **ID**; clearly, in this case credulous and skeptical acceptance coincides.

Table 1: Complexity of reasoning with AFs. $\mathscr{C}$-c means that the problem is complete for class $\mathscr{C}$.

σ	$Cred_\sigma$	$Skept_\sigma$	Ver_σ	$Exists_\sigma$	$Exists_\sigma^{\neg\emptyset}$	$Enum_\sigma$
CO	NP-c	P-c	in L	trivial	NP-c	nOP
PR	NP-c	Π_2^P-c	coNP-c	trivial	NP-c	nOP
ST	NP-c	coNP-c	in L	NP-c	NP-c	nOP
GR	P-c	P-c	P-c	trivial	in L	in DelayP
STG	Σ_2^P-c	Π_2^P-c	coNP-c	trivial	in L	nOP
SST	Σ_2^P-c	Π_2^P-c	coNP-c	trivial	NP-c	nOP
ID	in Θ_2^P	in Θ_2^P	in Θ_2^P	trivial	in Θ_2^P	nOP

Complexity of reasoning problems under the various semantics has been studied in (Dimopoulos and Torres, 1996; Dunne and Bench-Capon, 2002; Caminada et al., 2012; Dvořák and Woltran, 2010; Dunne et al., 2013; Kröll et al., 2017). The most recent survey can be found in (Dvořák and Dunne, 2018). Table 1 provides an overview. We thereby assume familiarity with basic concepts such as completeness and the polynomial hierarchy (see (Arora and Barak, 2009) for more details). The class Θ_k^P is a refinement of the class Δ_k^P: it contains the problems that can be decided in polynomial time by a deterministic Turing machine with at most $\mathscr{O}(\log m)$ calls to a Σ_{k-1}^P oracle, where m is the input size. By nOP we denote that the enumeration problem is not contained in the class OutputP (also called TotalP), i.e. it is not solvable in polynomial time in the size of the input and the output (Johnson et al., 1988; Strozecki, 2010)[4]. Containment in DelayP on the other hand means that the extensions can be enumerated with a delay which is polynomial in the size of the input.

3. Format of ICCMA'17

This section presents the main design of the competition. The competition is organized into tracks, which are divided into tasks. Two sub-sections are devoted to their definitions. A third sub-section then presents the scoring system, which changed from ICCMA'15 in order to focus more on correctness of answers. Related to this issue, a fourth sub-section outlines how we verified correctness of

[4]Note that the result for **ID** is not published, but immediate by the fact that $Ver_{\mathbf{ID}}$ is coNP-complete (Dunne, 2009) and therefore the ideal extension is not computable in polynomial time.

answers. Finally, information about I/O requirements is given.

3.1. Tasks

A *task* is a reasoning problem under a particular semantics. We consider the semantics **CO**, **PR**, **ST**, and **GR** which have already been employed in the first edition, and additionally the semantics **SST**, **STG**, and **ID**; the motivation to add these three semantics is due to the fact that their complexity differs from the semantics already considered. Following ICCMA'15 we consider four different problems:

DC-σ: *Given* $F = (\mathsf{A}, \rightarrow)$ and $a \in \mathsf{A}$, *decide* whether a is credulously accepted in F under σ,

DS-σ: *Given* $F = (\mathsf{A}, \rightarrow)$ and $a \in \mathsf{A}$, *decide* whether a is skeptically accepted in F under σ,

SE-σ: *Given* $F = (\mathsf{A}, \rightarrow)$, *return* some set $E \subseteq \mathsf{A}$ that is a σ-extension of F,

EE-σ: *Given* $F = (\mathsf{A}, \rightarrow)$, *enumerate* all sets $E \subseteq \mathsf{A}$ that are σ-extensions of F,

for the seven semantics $\sigma \in \{\mathbf{CO}, \mathbf{PR}, \mathbf{ST}, \mathbf{SST}, \mathbf{STG}, \mathbf{GR}, \mathbf{ID}\}$.

For single-status semantics (**GR** and **ID**) some problems collapse, i.e. **SE** and **EE** require to compute the unique extension; and **DC** and **DS** are equivalent. Thus, for **GR** and **ID** only the problems **SE** and **DC** are considered. At this point, we also recall the well known fact that **DS**-**CO** coincides with **DC**-**GR** and **DC**-**PR** coincides with **DC**-**CO**.

The combination of problems with semantics amounts to a total number of 24 tasks.

3.2. Tracks

All tasks for a particular semantics constitute a *track*. Therefore, there is one track for each semantics.

Moreover, the competition features an eighth special track, the Dung's Triathlon. It is named after Phan Minh Dung, and involves enumerating three of the main semantics (grounded, stable, and preferred) from his seminal paper (Dung, 1995). The aim of this track is to evaluate solvers also with respect to their capability of exploiting interrelationships between different semantics.

More concretely, the problem to solve in this track is defined as follows:

D3: *Given* $F = (\mathsf{A}, \rightarrow)$, *enumerate*

- all sets $E \subseteq \mathsf{A}$ that are **GR**-extensions[5] of F, followed by
- all sets $E \subseteq \mathsf{A}$ that are **ST**-extensions of F, followed by
- all sets $E \subseteq \mathsf{A}$ that are **PR**-extensions of F.

3.3. Scoring system

Each solver can compete in an arbitrary set of tasks. If a solver supports all tasks of a track, it also participates in the track.

To compute the score for a solver, we start by defining the number of points a solver can get for each instance:

- 1 point, if it delivers a *correct* result;
- -5 points, if it delivers an *incorrect* result; or
- 0 points otherwise.

The precise understanding of what is a *correct*, or an *incorrect*, answer will be given in the next sub-section. Here, we focus on explaining how the solvers are ranked.

But before going into these details, we would like to stress a difference to ICCMA'15: in this edition wrong answers are penalized, while in ICCMA'15 they were treated as being neither correct nor incorrect, and got 0 points. The objective, as already stated before, is to put focus on solvers' correctness.

The *score* of a solver for a particular task is the sum of points over all instances. The ranking of solvers for a task is then based on the scores in descending order. Ties between solvers with the same score are broken by the total time it took the solver to return correct results.

The ranking of solvers for a track is based on the sum of scores over all tasks of the track, where each task is guaranteed to have the same impact on the evaluation of the track by all having the same number of instances (see Section 5 for details about the number of instances). Again, ties are broken by the total time it took the solver to return correct results.

As far as the Dung's triathlon in concerned, scoring and ranking follow the same method as for the single tasks.

[5] Although grounded semantics is a single-status semantics, we treat it here like a multi-status semantics for the sake of uniformity.

3.4. Verification of answers

In this sub-section we discuss how the solvers' answers have been verified. Before going into the details, in the following we precisely define the concepts of *correct* and *incorrect* answers:

- **DC**-σ (resp. **DS**-σ): if the queried argument is credulously (resp. skeptically) accepted in the given AF under σ, the result is *correct* if it is `YES` and *incorrect* if it is `NO`; if the queried argument is not credulously (resp. not skeptically) accepted in the given AF under σ, the result is *correct* if it is `NO` and *incorrect* if it is `YES`.

- **SE**-σ: the result is *correct* if it is a σ-extension of the given AF and *incorrect* if it is a set of arguments that is not a σ-extension of the given AF. If the given AF has no σ-extensions, then the result is *correct* if it is `NO` and *incorrect* if it is any set of arguments.

- **EE**-σ: the result is *correct* if it is the set of all σ-extensions of the given AF and *incorrect* if it contains a set of arguments that is not a σ-extension of the given AF.

- **D3**: the result is *correct* if it is the set of all **GR**-extensions, followed by the set of all **ST**-extensions, followed by the set of all **PR**-extensions, and *incorrect* if the first set contains a set of arguments that is not the **GR**-extension, the second set contains a set of arguments that is not a **ST**-extension, or the third set contains a set of arguments that is not a **PR**-extension.

Intuitively, a result is neither correct nor incorrect (and therefore gets 0 points) if (i) it is empty (e.g. the timeout was reached without answer) or (ii) it is not parsable with respect to the required output format (e.g. due to some unexpected error message). For **EE**-σ there is also the case that the result (iii) contains σ-extensions, but not all of them. Case (iii) applies also to the Dung's triathlon, recursively on the three sub-problems.

To verify the correctness of results, we employ the following checking procedure. First, we generate reference solutions by running ASPARTIX-D (Egly et al., 2010; Gaggl et al., 2015), extended by the encodings for the new semantics,[6] on all benchmarks selected for the competition (see Section 5). For the

[6]The ICCMA'15 version can be found at `https://iccl.inf.tu-dresden.de/web/`

instances that ASPARTIX-D is able to solve, we compare the solutions with the reference solutions in order to assess correctness. For the other instances, we then use dedicated ASP encodings to check single extensions (available at `http://argumentationcompetition.org/2017/SE_encodings.zip`) to verify answers for the **SE** and **EE** reasoning problems. These ASP encodings are directly derived from the ASPARTIX encodings – the part for guessing an extension is replaced by the given extension which is to be checked. For the other tasks as well as these cases where also checking all single extensions was not feasible, we then consider the solution provided by the majority of solvers as correct (other solutions could always be checked to be wrong though). The detailed number of uniquely solved instances by a certain solver will be given in Section7, also including the number of instances for each track and solver which could not be verified. In total only approx. 0.1% out of the 105350 solutions could not be verified and thus have been rated with 1 point. In none of the tracks these had an influence on the ranking of the solvers.

3.5. Solver requirements

Participant systems were required to support the same input-output format as used in 2015. Details on the input and output formats can be found in (ICCMA'17-Solreq, 2017).

4. Benchmark Suite

In this section we outline the benchmark suite available for ICCMA'17, which has been the starting point for the selection phase (described in the next section). The suite is composed both by domains employed in ICCMA'15 and by new domains, the latter received in response to a dedicated *call for benchmarks*. The next two sub-sections are devoted to the presentation of these two sets of domains.

4.1. Previous domains

ICCMA'15 introduced three new AF generators, called GroundedGenerator, StableGenerator, and SccGenerator, each of them aiming to produce challenging

`Sarah_Alice_Gaggl/ASPARTIX-D`; the additional encodings are available at `https://www.dbai.tuwien.ac.at/proj/argumentation/systempage`. The choice of this particular solver is due to (i) its declarative nature, (ii) its good results in 2015, (iii) the fact that it is "third-party" in 2017 given that it does not participate, and (iv) its reputation in the community ("state of the art of ASP-based solvers" Bistarelli et al. (2014)).

AFs addressing certain aspects of computational difficulty. They have been implemented (Cerutti et al., 2014b) and employed to generate the AFs that constituted the benchmark suite of ICCMA'15. In the following, we briefly describe the generators, but refer to (Thimm and Villata, 2017) for more details.

GroundedGenerator This generator aims at producing AFs with large grounded extensions. It takes the number of arguments n and probability `probAttacks` as parameters, linearly orders the arguments and adds an attack from argument a to argument b in case $a < b$ with probability `probAttacks`. Finally, it adds random attacks between the arguments not yet connected and the graph component obtained in the first part.

SccGenerator This generator aims at producing AFs such that the graph features many Strongly Connected Components (SCCs). It first partitions the arguments (the number of which is given by parameter n) into `nSCCs` (also given as parameter) components which are linearly ordered. Within each component, attacks between any pair of arguments are added with probability given by parameter `innerAttackProb`. Among arguments of different components, attacks are added with probability given by parameter `outerAttackProb`, but under the condition that the component of the attacking arguments is ranked lower with respect to the linear order on components than the component of the attacked argument.

StableGenerator This generator aims at producing AFs with a large number of stable extensions. It first identifies a set of arguments to form an acyclic subgraph of the AF and, consequently, to contain the grounded extension. Among the other arguments, subsets are iteratively singled out to form stable extensions by attacking all other arguments. Besides the parameter n for the number of arguments, the algorithm is further guided by the parameters `minNumExtensions`, `maxNumExtensions`, `minSizeOfExtensions`, `maxSizeOfExtensions`, `minSizeOfGroundedExtension`, and `maxSizeOfGroundedExtension`, which determine heuristic values for the minimum and maximum number of stable extensions, the minimum and maximum size of stable extensions, and the minimum and maximum size of grounded extensions, respectively.

4.2. New Domains

ICCMA'17 takes advantage, for the first time, of a dedicated *call for benchmarks*, which is customary in other competitions. The goal of this call has been to

enlarge the set of domains that are considered in the competition, and thus possibly having a more heterogeneous set of benchmarks in the evaluation. Contributors were asked to provide an instance set for the benchmark they submitted, and/or an instance generator, possibly with an indication about the estimated difficulty of the instances. We have received 6 submissions, among them AF generators as well as concrete sets of AFs, thus meeting our desiderata to have a heterogeneous set of benchmarks, i.e. random, crafted, and application-oriented, as a benchmark suite of the competition.

Herewith we briefly describe the domains that were submitted:

"ABA2AF" by Tuomo Lehtonen (University of Helsinki, Finland), Johannes P. Wallner (TU Wien, Austria), Matti Järvisalo (University of Helsinki, Finland), are assumption-based argumentation (ABA) benchmarks translated to AFs. ABA problems are one of the prevalent forms of structured argumentation in which, differently from AFs, the internal structure of arguments is made explicit through derivations from more basic structure (Toni, 2014). The translation employed is described in (Lehtonen et al., 2017). The original ABA set contains randomly generated cyclic and acyclic ABAs that, after a selection from the authors, resulted in a total of 426 instances.

AdmBuster by Martin Caminada (Cardiff University, UK), Mikolaj Podlaszewski (Talkwalker), is a crafted benchmark example for (strong) admissibility. It is made of a fixed structure composed of 4 sets of arguments and predetermined sets of attacks. The number n is a parameter of the generator. Two "starting" and "terminal" sets are composed of only one element, one having only outgoing edges and the other only incoming edges. The two "intermediate" sets have cardinality $n-2$, and their attack relations are constructed in order to have only one complete labelling. Details can be found in (Caminada, 2014). At the competition, 13 instances generated with different values of n are considered.

AFBenchGen2 by Federico Cerutti (Cardiff University, UK), Mauro Vallati (University of Huddersfield, UK), Massimiliano Giacomin (University of Brescia, Italy), is a generator of random AFs of three different graph classes, with a configurable number of arguments (Cerutti et al., 2016a). The three classes correspond to Erdös-Rényi (Erdös and Rényi, 1959), which selects attacks randomly, Watts-Strogatz (Watts and Strogatz, 1998), which aims for a *small-world* topology of networks being not completely random nor

regular, and Barabasi-Albert (Barabasi and Albert, 1999) for large networks. For each graph class, the generator takes the number of arguments n as parameter. 1400 instances have been generated, of which 500 are from Barabasi-Albert class, 500 are from Erdös-Rényi class, and 400 are from Watts-Strogatz class. In the following, we provide some more details for such three classes:

- Barabasi-Albert: This graph class is motivated by a common property of many large networks, i.e. that the node connectivities follow a scale-free power-law distribution. Therefore, the generator of a Barabasi-Albert graph iteratively connects a new node by preferring sites that are already well connected. In addition, a postprocessing procedure adds attacks in order to ensure a certain amount of cycles in the graph. This amount is controlled by the parameter `probCycles`. An attack is added as long as the number of SCCs of the AF is higher than $n \cdot (1 - \texttt{probCycles})$.
- Erdös-Rényi: Graphs are generated by randomly selecting attacks between arguments. For any two distinct arguments, the probability of an attack between them is given by the parameter `probAttacks`. The direction of the attack is chosen randomly.
- Watts-Strogatz: First, a ring of n arguments is generated where each argument is connected to its k (a parameter of the generator) nearest neighbors in the ring. Then, each argument is connected to the remaining arguments with a probability β (another parameter of the generator). Finally, as in Barabasi-Albert, random attacks are added as long as the number of SCCs of the AF is higher than $n \cdot (1 - \texttt{probCycles})$.

"Planning2AF" by Federico Cerutti (Cardiff University, UK), Massimiliano Giacomin (University of Brescia, Italy), Mauro Vallati (University of Huddersfield, UK), are AFs obtained from translating the well-known Blocksworld and Ferry planning domains. Each planning instance is first encoded as a propositional formula, by using the method in (Sideris and Dimopoulos, 2010); then, each clause is transformed into a material implication; and, finally, to each material implication the transformation in (Wyner et al., 2015) is applied. This domain comprises 385 instances.

SemBuster by Martin Caminada (Cardiff University, UK), Bart Verheij (Rijksuniversiteit Groningen, Netherlands), is a crafted benchmark example for

semi-stable semantics. It has a fixed structure composed by 3 sets of arguments of equal cardinality, and predetermined sets of attacks. Given a parameter n, attack relations are defined in a way that each instance has exactly $n+1$ complete labellings that correspond also to preferred labellings, but only one among those corresponds to a semi-stable extension. Details can be found in (Caminada and Verheij, 2010). At the competition, 16 instances generated with different values of n are considered.

"Traffic" by Martin Diller (TU Wien, Austria), are graphs obtained from real world traffic networks data available at `https://transitfeeds.com/` expressed as AFs. Given a graph, the corresponding AF contains the same set of vertices as the graph, and the attack relation is defined as follows: Given an existing edge, and a probability for the attack of being symmetric, the generator decides whether there are both attacks, or randomly selects the attack. A total of 600 instances are provided, 200 for each of the probabilities 0.2, 0.5, and 0.8. Although these instances do not directly relate to argumentation applications, we decided to include them in the competition, in order to have an orthogonal class of sparse graphs with certain structural features.

More detailed descriptions for such domains can be found in the ICCMA'17 home page at (ICCMA'17-Soldes, 2017).

Table 2 gives details on the collected benchmarks by stating, for each domain, the number of instances as well as the parameters used for generating the instances. If the benchmark submission consists of a set of instances, we simply considered them all. For domains emerging from submissions of benchmark generators, we produced instances randomly with the aim of covering a possibly broad range of difficulty. The exact parameters used for generating the instances can be read off from Table 2. In some cases, parameters are chosen randomly from an interval. This is denoted by $\texttt{random}[a,b]$. In other cases, all values in a set are considered, denoted by $\{v_1, v_2, \ldots, v_n\}$.

Thus, the benchmark suite of ICCMA'17 is finally composed of 3990 instances over 11 domains. This yields a healthy mixture of benchmarks ranging from random instances to more structured AFs which are either handcrafted or instantiated from different application domains.

Domain	**Inst.**	**Parameters**
ABA2AF	426	all submitted instances
AdmBuster	13	n in $\{1000, 2000, 4000, \ldots, 10000, 20000, 50000, 100000, 200000, 500000, 1000000, 2000000\}$
Barabasi-Albert	500	5 random instances for each (n, `probCycles`) in $\{20, 40, \ldots, 200\} \times \{0, 0.1, \ldots, 0.9\}$
Erdös-Rényi	500	10 random instances for each (n, `probAttacks`) in $\{100, 200, \ldots, 500\} \times \{0.1, 0.2, \ldots, 1.0\}$
GroundedGenerator	50	$n =$ `random`$[100, 1500]$; 10 random instances for each `probAttacks` in $\{0.01, 0.02, \ldots, 0.05\}$
Planning2AF	385	all submitted instances
SccGenerator	600	$n =$ `random`$[100, 1500]$; `nSCCs` $=$ `random`$[1, 50]$; 25 random instances for each (`innerAttackProb`, `outerAttackProb`) in $\{0.3, 0.4, \ldots, 0.7\} \times \{0.05, 0.1, 0.15, 0.2\}$. $n =$ `random`$[5000, 10000]$; no. SCCs `random`$[40, 50]$; 5 random instances for each (`innerAttackProb`, `outerAttackProb`) in $\{0.3, 0.4, \ldots, 0.7\} \times \{0.05, 0.1, 0.15, 0.2\}$.
SemBuster	16	n in $\{60, 150, 300, 600, \ldots, 1800, 2400, 3000, 3600, 4200, 4800, 5400, 6000, 7500\}$
StableGenerator	500	$n =$ `random`$[100, 800]$; 500 random instances with parameters `minNumExtensions` $= 5$, `maxNumExtensions` $= 30$, `minSizeOfExtensions` $= 5$, `maxSizeOfExtensions` $= 40$, `minSizeOfGroundedExtension` $= 5$, `maxSizeOfGroundedExtension` $= 40$
Traffic	600	all submitted instances
Watts-Strogatz	400	(n, k, β, `probCycles`) in $\{100, 200, \ldots, 500\} \times \{\log_2(n), 2 \cdot \log_2(n), 3 \cdot \log_2(n), 4 \cdot \log_2(n)\} \times \{0.1, 0.3, \ldots, 0.9\} \times \{0.1, 0.3, 0.5, 0.7\}$

Table 2: Description of (generated) benchmarks constituting the benchmark suite.

5. Benchmark Selection

With the benchmark suite described in the previous section, the goal of this phase is to select the instances that are indeed run in the competition. In order to guide this selection, the instances are classified into hardness categories according to the performance of a set of solvers from the previous competition. Finally, the instances to be run at the competition are selected based on this classification, following a predefined distribution over hardness categories.

As the tasks of the competition span over a wide range of complexity (cf. Table 1), a single set of benchmarks for the whole competition might not be suitable. Therefore we aim to adjust the benchmarks to the complexity of the tasks, while keeping the total amount of different benchmarks manageable. To this end, we introduce a grouping of tasks according to their difficulty, such that each of the groups gets a dedicated set of benchmarks. The classification into groups A to E is based on known complexity results and corroborated by the analysis of the results of ICCMA'15. The applied grouping is the following:

Group A: **DS-PR**, **EE-PR**, **EE-CO**.

Group B: **DC-ST**, **DS-ST**, **EE-ST**, **SE-ST**, **DC-PR**, **SE-PR**, **DC-CO**.

Group C: **DS-CO**, **SE-CO**, **DC-GR**, **SE-GR**.

Group D: **DC-ID**, **SE-ID**.

Group E: **DC-SST**, **DS-SST**, **EE-SST**, **SE-SST**, **DC-STG**, **DS-STG**, **EE-STG**, **SE-STG**.

Hence, the classification and selection has to be done for each group. However, since there are no reference solvers for the tasks of groups D and E (these are the ones newly employed in this edition), we do not perform a dedicated selection for these groups. Instead, the tasks of these groups are assigned the same benchmark set as group A, because they are of high complexity and we expect solvers to be less mature since ICCMA'15 did not feature these tasks yet.

The following sub-sections present how instances are classified, how instances are selected, and, finally, how the query arguments for the **DC** and **DS** tasks are selected.

5.1. Benchmark Classification

To classify the hardness of instances, competitions in other research fields such as SAT (SAT-Comp, 2009; Järvisalo et al., 2012; Balint et al., 2015), ASP (Gebser et al., 2017), and IPC for automated planning (Vallati et al., 2015), employ best solvers from the most recent competition in the series. We follow this idea by also doing a classification of benchmarks based on the performance of solvers from ICCMA'15. However, in ICCMA the situation shows two significant differences. On the one hand, the number of tasks and tracks employed in ICCMA (significantly) exceeds the number of tasks and tracks in other competitions. On the other hand, ICCMA'17 features new semantics (and, consequently, new tasks and tracks), so no reference results are at disposal.

Due to the second point, the option of selecting the best solvers from the previous edition for each task is not feasible. But, even considering only tasks which are being conducted for the second time, this option would lead to a very high number of solvers to run for the classification. Instead, we identify "representative" tasks for each task group A, B, and C which have also been conducted in ICCMA'15. Moreover, as mentioned earlier, we abstain from classifying instances for tasks in groups D and E, but merge these tasks with the ones from group A and employ the same set of benchmarks. We identify the following representative tasks which will be used for classification:

- Group A: **EE-PR**
- Group B: **EE-ST**
- Group C: **SE-GR**

All task groups contain enumeration as well as decision tasks. We select enumeration tasks as representative, as the performance of solvers on decision tasks highly depends on the argument for which acceptance is to be decided. Therefore, enumeration tasks can give a better estimate of the difficulty of instances.

(Best) Solver selection. For each representative task we aim to select "representative" solvers from ICCMA'15, to get a proper estimate of the instances' hardness. Solvers to run for each group are thus selected by (i) considering best performing solvers from 2015 for the tasks, and (ii) ensuring that the selected solvers are based on different solving approaches, in order not to have results biased through a single solving approach. The following solvers from ICCMA'15 are selected (see (Thimm and Villata, 2015) for system descriptions):

- Group A: Cegartix, CoQuiAAS, Aspartix-V
- Group B: Aspartix-D, ArgSemSAT, ConArg
- Group C: CoQuiAAS, LabSATSolver, ArgSemSAT

Both Cegartix (Dvořák et al., 2014) and ArgSemSAT (Cerutti et al., 2014a) implement (iterative) SAT based approaches; CoQuiAAS (Lagniez et al., 2015) makes use of Partial Max-SAT; Aspartix-V and Aspartix-D (Egly et al., 2010; Gaggl et al., 2015) employ a translation to ASP; ConArg (Bistarelli and Santini, 2011) is based on Constraint Programming; and LabSATSolver (Beierle et al., 2015) implements a direct approach (for **SE**-**GR**). All of the solvers have been among the top 5 solvers in the respective tasks in ICCMA'15. Hence, the selection is in line with (i) and (ii).

Hardness categories. The obtained performance results of the 3 selected solvers in each group are then taken to classify instances into hardness categories by picking the upmost category such that the following conditions apply:

[very easy] Instances completed by all systems in less than 6 seconds solving time.

[easy] Instances completed by all systems in less than 60 seconds solving time.

[medium] Instances completed by all systems in less than 10 minutes solving time.

[hard] Instances completed by at least one system in 20 minutes (twice the timeout) solving time.

[too hard] Instances such that none of the systems finished solving in 20 minutes.

The results of the classification are summarized in Tables 3, 4, and 5 for task groups A, B, and C[7], respectively. It can be seen that almost every combination of domain and difficulty category contains instances. Only for the "too hard" category we are not able to obtain instances for every domain (even for no domain for task group C). If at least two of the representative solvers crashes for an instance, the instance is not classified (abbreviated by "n. c." in the tables), and therefore not considered for selection.

[7] AdmBuster domain in Table 5 contains two additional instances with n of 1500000 and 2500000.

Table 3: Classification results for task group A.

A: **EE-PR**	total	very easy	easy	medium	hard	too hard	n. c.
ABA2AF	426	381	19	16	10	0	0
AdmBuster	13	4	3	2	4	0	0
Barabasi-Albert	500	267	25	20	42	145	1
Erdös-Rényi	500	180	109	43	46	122	0
Watts-Strogatz	400	264	28	10	12	86	0
GroundedGenerator	50	9	8	6	27	0	0
Planning2AF	385	95	35	34	187	33	1
SccGenerator	600	398	78	44	79	0	1
SemBuster	16	2	1	3	9	1	0
StableGenerator	500	260	34	24	182	0	0
Traffic	600	164	11	11	284	127	3
Total	3990	2024	351	213	882	514	6

Table 4: Classification results for task group B.

B: **EE-ST**	total	very easy	easy	medium	hard	too hard	n. c.
ABA2AF	426	407	18	1	0	0	0
AdmBuster	13	9	1	1	2	0	0
Barabasi-Albert	500	262	19	5	122	92	0
Erdös-Rényi	500	247	102	31	49	71	0
Watts-Strogatz	400	201	39	26	76	58	0
GroundedGenerator	50	19	25	5	1	0	0
Planning2AF	385	117	5	5	159	99	0
SccGenerator	600	248	66	65	218	3	0
SemBuster	16	6	6	4	0	0	0
StableGenerator	500	225	26	37	73	139	0
Traffic	600	275	7	2	70	245	1
Total	3990	2016	314	182	770	707	1

Table 5: Classification results for task group C.

C: **SE-GR**	total	very easy	easy	medium	hard	too hard	n. c.
ABA2AF	426	404	21	1	0	0	0
AdmBuster	15	7	1	1	6	0	0
Barabasi-Albert	500	500	0	0	0	0	0
Erdös-Rényi	500	424	44	11	21	0	0
Watts-Strogatz	400	296	36	21	47	0	0
GroundedGenerator	50	20	25	1	4	0	0
Planning2AF	385	359	23	3	0	0	0
SccGenerator	600	485	84	31	0	0	0
SemBuster	16	3	1	0	12	0	0
StableGenerator	500	308	62	42	88	0	0
Traffic	600	459	42	51	50	0	0
Total	3992	3265	339	162	228	0	0

5.2. Benchmark selection

The final benchmark set for each task group is made up of 350 instances, distributed over the difficulty categories as follows:

- 50 very easy,
- 50 easy,
- 100 medium,
- 100 hard,
- 50 too hard.

Due to the lack of "too hard" instances for group C (cf. Table 5), the number of "hard" instances is increased to 150 there.

We aim for an even distribution of benchmarks over levels of difficulty, but also among domains. Now, in order to select n instances for a certain task group and a certain class of difficulty, we apply the following procedure: for each domain d, we are given the set I_d of instances and want to select a subset S_d of these instances. Now for each domain such that I_d is non-empty, we select one element of I_d at random, i.e. remove it from I_d and add it to S_d. We repeat this process until we have selected n instances, i.e. the sum over all $|S_d|$ is n. In the last iteration, when the number of domains where I_d is non-empty is

Table 6: Number of selected instances for each task group, difficulty class, and domain, where difficulty classes 1 to 5 stand for very easy, easy, medium, hard, and too hard, respectively. "T" indicates the total number of selected instances.

Task group	A						B						C					
Difficulty class	1	2	3	4	5	T	1	2	3	4	5	T	1	2	3	4	5	T
ABA2AF	5	5	12	10	0	32	5	5	1	0	0	11	5	6	1	0	0	12
AdmBuster	4	3	2	4	0	13	4	1	1	2	0	8	4	1	1	6	0	12
Barabasi-Albert	5	5	11	10	10	41	5	5	5	14	8	37	5	0	0	0	0	5
Erdös-Rényi	5	5	11	10	9	40	5	5	19	13	7	49	5	6	11	21	0	43
Watts-Strogatz	5	5	10	10	10	40	5	5	20	14	8	52	5	6	21	36	0	68
GroundedGenerator	4	5	6	9	0	24	4	4	5	1	0	14	5	6	1	4	0	16
Planning2AF	5	6	12	10	10	43	5	5	5	14	8	37	5	6	3	0	0	14
SccGenerator	5	5	11	9	0	30	4	5	19	14	3	45	4	6	21	0	0	31
SemBuster	2	1	3	9	1	16	4	5	4	0	0	13	3	1	0	12	0	16
StableGenerator	5	5	11	9	0	30	4	5	19	14	8	50	4	6	20	35	0	65
Traffic	5	5	11	10	10	41	5	5	2	14	8	34	5	6	21	36	0	68
Total	50	50	100	100	50	350	50	50	100	100	50	350	50	50	100	150	0	350

higher than the number of instances that remains to be selected, the domains to be chosen from are determined randomly. A more rigorous description of this procedure can be found at `http://argumentationcompetition.org/2017/benchmark-selection-algorithm.pdf`.

Example 2. Assume domains $D = \{\alpha, \beta, \gamma, \delta\}$ such that we have 1 instance for domain α, 2 for β, 4 for γ, and 11 for δ, i.e. $|S_\alpha| = 1$, $|S_\beta| = 2$, $|S_\gamma| = 4$, and $|S_\delta| = 11$. Further assume that we want to select $n = 10$ instances. The selection algorithm will return all instances from α and β, 3 instances from γ and δ, and 1 additional instance randomly selected from either γ or δ.

The numbers of selected instances for every domain, task group, and difficulty category can be read off from Table 6.

The instances for Dung's triathlon are selected based on the classification for task group A, but by a separate process. That means that the numbers of instances per domain coincide with group A, but instances are not necessarily the same.

No stable extensions. Semi-stable and stage extensions coincide with stable extensions if at least one of the latter exists. In this case, the complexity of the reasoning tasks drops to the level of the corresponding tasks for stable semantics (cf. Table 1). Therefore, in order to force solvers to deal with the "full hardness" of semi-stable and stage semantics, we want to make sure that the selection for

these semantics contains a sufficient amount of benchmarks possessing no stable extensions. To this end, we checked the selected instances on existence of stable extensions by running ASPARTIX-D from ICCMA'15 (winning solver for all tasks involving stable semantics). The numbers are shown in Table 7: for 22 instances no answer is provided by ASPARTIX-D. We consider the number of instances without stable extensions (114) to be satisfactory.

Table 7: Analysis of the existence of stable extensions.

hardness category	$\mathbf{ST}(F) \neq \emptyset$	$\mathbf{ST}(F) = \emptyset$	unknown
very easy	34	16	0
easy	34	16	0
medium	60	40	0
hard	56	33	11
too hard	30	9	11
total	214	114	22

5.3. Argument Selection

Due to the joint evaluation of all tasks for a semantics, making up a track, the number of benchmarks has to be constant among the tasks. Therefore, for the acceptance tasks we cannot select multiple arguments for every instance. Instead, we select only one argument for each instance, with the exception that we dropped the "very easy" instances for acceptance tasks and selected two arguments to be queried for the "too hard" instances, which again amounts to 350 instances in total.

For each task group except group D the query arguments are selected at random, maintaining a minimum number of yes- and no-instances, respectively. For group A and E, the same arguments are used.

Ideal Semantics. While the selection of arguments for the decision tasks **DC** and **DS** in all task groups except D was done randomly, for the task **DC-ID** we were aiming for a more sophisticated selection in order to select the "interesting" arguments for the acceptance task.

That selection was based on the following insights:

- if the query argument is contained in the grounded extension, then the answer to **DC-ID** is always yes;

Table 8: Distribution of selected arguments for **DC-ID**, with F being the AF and G its grounded extension.

	G	$\bigcap\mathbf{PR}(F) \setminus G$	$A \setminus \bigcap\mathbf{PR}(F)$
easy	14	**15**	21
medium	21	**21**	58

- if the query argument is not contained in every preferred extension, then the answer to **DC-ID** is always no.

Hence, we aimed for a considerable number of instances for which we select an argument contained in all preferred extensions, but not in the grounded extension.

We did so by considering the following strategy: Given an AF $F = (A,R)$, let $G \in \mathbf{GR}(F)$ be its grounded extension. Moreover, let α and β be random variables with a uniform distribution in the interval $[0,1]$.

1. if $\bigcap\mathbf{PR}(F) \setminus G \neq \emptyset$ and $\alpha < 0.9$, select an argument randomly from $\bigcap\mathbf{PR}(F) \setminus G$;
2. otherwise, if $G \neq \emptyset$ and $\beta < 0.6$, select an argument randomly from G;
3. otherwise, select an argument randomly from $A \setminus \bigcap\mathbf{PR}(F)$.

That is, if arguments that we consider "interesting" as described before exist, we select one of them with a high probability (0.9). Otherwise we give a slight preference (probability of 0.6) to the arguments contained in the grounded extension, given that the grounded extension is not empty.

This strategy is applied to the selection of query arguments for instances in the easy and medium hardness category. The obtained distributions of the selected arguments is given in Table 8. We randomly select the arguments for the hard and too hard instances.

6. Participants

Sixteen solvers participate in the competition, and are listed in Table 9, together with the list of contributors and their institutions, and a main reference in the last column. New entries compared to the previous edition are marked by *.

System descriptions for all solvers can be found on the competition webpage at `http://argumentationcompetition.org/2017/submissions.html`. The set of participants is characterized by a great variety of solving approaches. We provide a grouping based on these approaches and provide some highlights for each group. Detailed results will be presented in Section 7.

Solver	Contributors	Reference
argmat-clpb⋆	Fuan Pu (Tsinghua University, China) Guiming Luo (Tsinghua University, China) Yucheng Chen (Tsinghua University, China)	Pu et al. (2017) https://sites.google.com/site/argumatrix/
argmat-dvisat⋆	Fuan Pu (Tsinghua University, China) Guiming Luo (Tsinghua University, China) Ya Hang (Tsinghua University, China)	Pu et al. (2017) https://sites.google.com/site/argumatrix/
argmat-mpg⋆	Fuan Pu (Tsinghua University, China) Guiming Luo (Tsinghua University, China) Ya Hang (Tsinghua University, China)	Pu et al. (2017) https://sites.google.com/site/argumatrix/
argmat-sat⋆	Fuan Pu (Tsinghua University, China) Guiming Luo (Tsinghua University, China) Ya Hang (Tsinghua University, China)	Pu et al. (2017) https://sites.google.com/site/argumatrix/
ArgSemSAT	Federico Cerutti (Cardiff University, UK) Mauro Vallati (University of Huddersfield, UK) Massimiliano Giacomin (University of Brescia, Italy) Tobia Zanetti (University of Brescia, Italy)	Cerutti et al. (2014a) https://sourceforge.net/projects/argsemsat/
ArgTools	Samer Nofal (German Jordanian University, Jordan) Katie Atkinson (University of Liverpool, UK) Paul E. Dunne (University of Liverpool, UK)	Nofal et al. (2016) https://sourceforge.net/projects/argtools
ASPrMin⋆	Wolfgang Faber (University of Huddersfield, UK) Mauro Vallati (University of Huddersfield, UK) Federico Cerutti (Cardiff University, UK) Massimiliano Giacomin (University of Brescia, Italy)	Faber et al. (2016) https://helios.hud.ac.uk/scommv/storage/ASPrMin-v1.0.tar.gz
cegartix	Wolfgang Dvořák (TU Wien, Austria) Matti Järvisalo (University of Helsinki, Finland) Johannes P. Wallner (TU Wien, Austria)	Dvořák et al. (2014) http://www.dbai.tuwien.ac.at/proj/argumentation/cegartix/
Chimærarg⋆	Federico Cerutti (Cardiff University, UK) Mauro Vallati (University of Huddersfield, UK) Massimiliano Giacomin (University of Brescia, Italy)	Cerutti et al. (2018) https://github.com/federicocerutti/Chimaerarg
ConArg	Stefano Bistarelli (University of Perugia, Italy) Fabio Rossi (University of Perugia, Italy) Francesco Santini (University of Perugia, Italy)	Bistarelli and Santini (2011) http://www.dmi.unipg.it/conarg/
CoQuiAAS	Jean-Marie Lagniez (University of Artois, France) Emmanuel Lonca (University of Artois, France) Jean-Guy Mailly (University of Artois, France)	Lagniez et al. (2015) http://www.cril.univ-artois.fr/coquiaas
EqArgSolver⋆	Odinaldo Rodrigues (King's College London, UK)	Gabbay and Rodrigues (2016) http://nms.kcl.ac.uk/odinaldo.rodrigues/eqargsolver
gg-sts⋆	Tomi Jahunen (Aalto University, Finland) Shahab Tasharrofi (Aalto University, Finland)	Bogaerts et al. (2016) https://research.ics.aalto.fi/software/sat/gg-sts/
goDIAMOND	Stefan Ellmauthaler (Leipzig University, Germany) Hannes Strass (Leipzig University, Germany)	Ellmauthaler and Strass (2014) https://sourceforge.net/p/diamond-adf/code/ci/go/tree/go/
heureka⋆	Nils Geilen (University of Koblenz-Landau, Germany) Matthias Thimm (University of Koblenz-Landau, Germany)	Geilen and Thimm (2017) https://github.com/nilsgeilen/heureka
pyglaf⋆	Mario Alviano (University of Calabria, Italy)	Alviano (2017) http://alviano.com/software/pyglaf/

Table 9: List of participants, with contributors, main reference paper, and link to the solver home page. ⋆ means newly submitted in the ICCMA series.

- Reductions to SAT: argmat-dvisat, argmat-sat, ArgSemSAT, cegartix, CoQuiAAS, gg-sts. All of these systems are implemented in C++. argmat-dvisat, argmat-sat, ArgSemSAT, and cegartix rely on reductions to SAT or (iterative) calls to SAT solvers. Two of them are among the top five solvers for each track except **GR**. While the backbone of both ArgSemSAT and cegartix is MiniSAT (Eén and Sörensson, 2003), argmat-dvisat and argmat-sat use CryptoMiniSat (`https://github.com/msoos/cryptominisat`) for SAT solving. gg-sts does not use SAT directly, but a reduction to an extension of the second-order logic system presented in (Bogaerts et al., 2016). Finally, CoQuiAAS uses various constraint programming techniques such as MaxSAT and Maximal Satisfiable Sets extraction.

- Reductions to CSP: argmat-clpb, argmat-mpg, ConArg. All of these systems are implemented in C++. argmat-clpb employs Constraint Logic Programming over Boolean variables in Prolog, while argmat-mpg uses a reduction to CSP using Gecode (`http://www.gecode.org/`). Both are based on formulations of argumentation problems in Boolean matrix algebra. Also ConArg implements a CSP approach using Gecode.

- Reductions to circumscription: pyglaf. pyglaf is implemented in Python and uses a circumscription solver extending the SAT solver glucose (Audemard and Simon, 2009). pyglaf participated in all tracks and is one of the most successful participants (see below).

- Reductions to ASP: ASPrMin, goDIAMOND. Both systems rely on the state-of-the-art ASP system clingo (Gebser et al., 2014). While goDIAMOND consists of a suite of different encodings for all the considered semantics (plus some native implementation for **GR** and **ID**), ASPrMIN makes use of a particular feature of clingo to control the heuristics such that only a certain form of subset-maximal answer-sets are delivered. This can be used to enumerate prefererred extensions. Consequently, ASPrMIN only participated in the **EE-PR** task (and, in fact, was the best solver for this single task) , whereas goDIAMOND entered all tracks (and reached the 2nd place in **ST**).

- Direct approaches: ArgTools, EqArgSolver, heureka. All of these solvers implement genuine algorithms in C++. EqArgSolver is an enhancement of GRIS (submitted to ICCMA'15, (Thimm and Villata, 2015)) and uses the discrete version of the Gabbay-Rodrigues iteration schema (Gabbay and

Rodrigues, 2016). ArgTools and heureka use various forms of backtracking algorithms on the basis of labellings of arguments.

- Portfolio-based approaches: Chimærarg. This system uses all the solvers that took part in the **EE-PR**, and respectively, **EE-ST** tasks of ICCMA'15, for generating a static schedule of solvers, whose performance are measured in terms of PAR10 score. Chimærarg participated in these two tasks in ICCMA'17, running Cegartix, GRIS, LabSATSolver and ArgTools. Unfortunately, Chimærarg delivered some wrong results and thus did not rank very well. Checking the number of solved instances however shows the potential of this system. We shall provide a separate analysis of comparing best solvers from ICCMA'15 and ICCMA'17 in Section 7.2.

In Table 10 we also provide information about the participation to tasks of each solver. The table contains the solvers in its rows, and the tasks in its columns: a "$\surd$" indicates that a solver competes in a task. The table is completed by a last row reporting the number of solvers participating to each task, and a last column with the number of tasks supported by each solver. Without taking into account ASPrMin and Chimærarg, which are specifically designed for enumeration and focus on very few semantics, all other solvers participate in at least 10 tasks. Half of the submitted solvers participate in all 25 tasks. The number of participants in single tasks ranges from 9 to 15 solvers. As far as participation in tracks is concerned, each track includes between 9 (**STG** semantics) and 14 (**CO**, **ST**, and **GR** semantics) solvers.

7. Results and Awards

In this section we present the results of our experiments, run on a cluster of Intel Xeon (Haswell) with 2.60GHz, where time and memory limits have been set to 10 minutes and 4 GB for all tasks but **D3**, and to 30 minutes and 6.5 GB for **D3**. The first sub-section is devoted to announce the winners. In the second sub-section we compare the award winners of this year and the best solvers from the ICCMA'15 competition on this year's benchmarks, on common tracks.

7.1. Award winners

In this sub-section we outline the winners of the competition. We remind that the winner of each track has been awarded.

	D3	CO				PR				ST				SST				STG				GR		ID		
Solver		**DC**	**DS**	**SE**	**EE**	**DC**	**DS**	**SE**	**EE**	**DC**	**DS**	**SE**	**EE**	**DC**	**DS**	**SE**	**EE**	**DC**	**DS**	**SE**	**EE**	**DC**	**DS**	**DC**	**SE**	**#Task**
argmat-clpb		√	√	√	√					√	√	√	√									√	√			10
argmat-dvisat	√	√	√	√	√	√	√	√	√	√	√	√	√									√	√	√	√	17
argmat-mpg	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	25
argmat-sat	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	25
ArgSemSAT		√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√					√	√			18
ArgTools		√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	24
ASPrMin									√																	1
cegartix	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	25
Chimærarg									√				√													2
ConArg	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	25
CoQuiAAS	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	25
EqArgSolver	√	√	√	√	√	√	√	√	√	√	√	√	√									√	√			15
gg-sts	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	25
goDIAMOND	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	25
heureka		√	√	√	√	√	√	√	√	√	√	√	√									√	√			14
pyglaf	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	25
#Solver	10	14	14	14	14	13	13	13	15	14	14	14	15	10	10	10	10	9	9	9	9	14	14	10	10	

Table 10: Tasks supported by solvers.

Results are presented in Figures 2–9, where at the top there is the ranking of solvers, and at the bottom the companion cactus plots. More specifically, the ranking of solvers is presented through tables organized as follows: the first column contains the name of the solver, the second column is the score of the respective solver (computed as defined in Section 3), while the third column reports the cumulative time of correctly solved instances. The fourth and fifth columns count the number of correct and wrong solutions given by each solver. In the sixth column the number of instances reaching timeout (TO) is given. The entries in seventh column (*Other*) stand for all other instances which also got 0 points. These are incomplete, memory-out and not-parseable solutions including those where the solvers could only return some error messages. The last column with *USC (u)* shows the unique solver contributions (USC), being the number of instances where only one solver could give a solution. The additional entries *(u)* stand for *unchecked*, that is the number of USC which could not be verified (this is not specified when USC is 0). Solvers are ordered by score, and ties are broken by cumulative time, as defined already in Section 3. Cactus plots, instead, present another view of the results by showing the cumulative number of correctly solved instances (x-axis) within a given CPU time (y-axis).

To sum up:

- pyglaf has been the winner of the **CO**, **ST**, and **ID** semantics;
- argmat-sat has been the winner of the **SST** and **STG** semantics;
- ArgSemSAT, CoQuiAAS and argmat-dvisat won the **PR**, **GR**, and **D3** semantics, respectively.

Interestingly, argmat-dvisat was not awarded as winner in any of the other track, but is the best solver in the **D3** track, where different semantics are considered. It is also worth to be noted that the set of winner solvers involves AF solvers based on different forms of reductions to SAT, CSP and circumscription.

In the following we discuss the correctness of the solvers and the USC. The solvers argmat-clpb, argmat-dvisat, argmat-mpg, argmat-sat, ArgSemSAT, EqArgSolver and heureka always returned correct solutions in all tracks. The solver pyglaf had only one incorrect solution in **DS-PR**, ConArg returned 4 incorrect answers in **EE-CO**, goDIAMOND had in total 15 wrong answers in tracks **EE-CO** and **EE-PR**. ArgTools had wrong solutions in tracks **DS-ST**, **DC-SST**, **DS-SST** and **DC-STG**, **DS-STG** and **EE-STG**. Although the solver CoQuiAAS is the

Solver	**Points**	Time	Correct	Wrong	TO	Other	USC (u)
pyglaf	1229	28774.77	1229	0	168	3	0
cegartix	1188	19846.86	1188	0	205	7	1 (0)
argmat-sat	1167	10472.57	1167	0	204	29	0
goDIAMOND	1156	18166.98	1176	4	181	39	2 (0)
argmat-dvisat	1151	15259.38	1151	0	226	23	0
CoQuiAAS	1132	10785.98	1132	0	149	119	0
argmat-mpg	1126	15133.06	1126	0	227	47	2 (2)
heureka	1018	9869.94	1018	0	309	73	0
ConArg	1017	51015.41	1037	4	130	229	19 (11)
ArgTools	935	36134.08	935	0	444	21	0
ArgSemSAT	900	20077.48	900	0	299	201	0
EqArgSolver	401	5430.45	401	0	92	907	0
argmat-clpb	40	4779.14	40	0	1109	251	0
gg-sts	-1170	18203.86	834	402	107	57	12 (12)

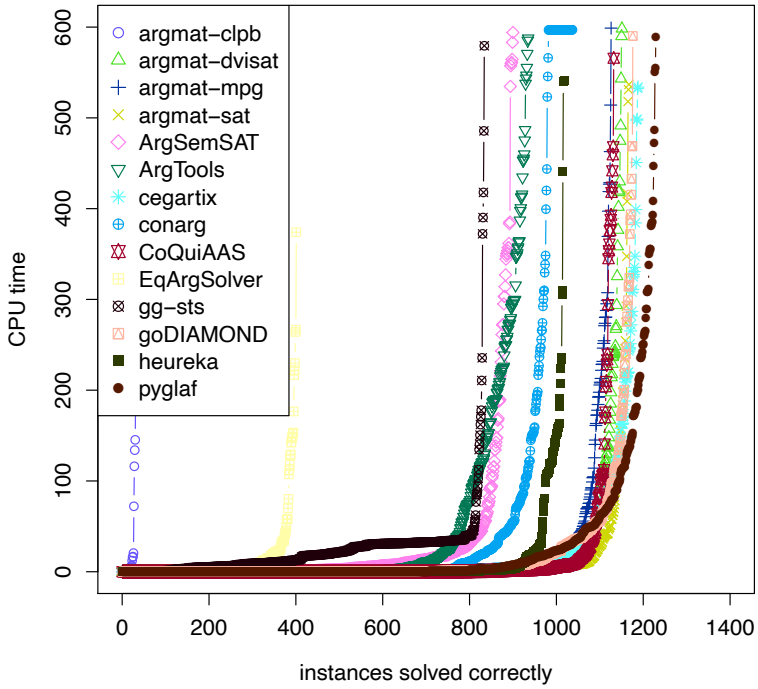

Figure 2: **CO** track: Ranking of solvers (top). Cactus plot of runtimes (bottom).

winner of the track **GR** and had no sanity problems in **CO**, in all other tracks many wrong answers were given. Finally gg-sts had wrong answers in all tracks. From the ranking of the solvers in all tracks it is easy to see that the penalty of -5 for each wrong answer had the desired effect to rank solvers with many wrong answers at the very end of the ranking.

The solvers ChimaerArg and ASPrMin are not listed in the tables, as they did not contribute in all tasks of a track, thus we summarize the results for them in the following. ASPrMin was the winner of the task **EE-PR** with 285 correct solutions, 0 wrong answers and thus obtained the score 285. The 2 USCs have been verified and 63 instances resulted in timeouts while 2 fall into the category Other. The solver ChimaerArg returned 255 correct solutions for the task **EE-ST** and 95 wrong answers, this results in the score -220. From the 21 USCs, 12 could not be verified. For **EE-PR**, ChimaerArg had 207 correct solutions and 23 wrong answers resulting in the score 92. All 120 answers with 0 points fall into the category Other.

Finally, Table 11 gives more details for the track winners. In particular it is given, for each track, the number of points acquired by the winning system in each domain. More in details, the table is organized as follows: the rows contain the domains and the columns the track winners. Each column is then divided in two sub-columns containing the number of points acquired by the solver and the maximum acquirable number of points in a domain, respectively. The table is complemented by a last row and a last column containing the total number of points acquired (or, acquirable) by each solver and in a domain, respectively.

7.2. Comparison to the results of ICCMA'15

By comparing the award winners of the 2017 event with those of the first edition, which cumulatively awarded CoQuiAAS, ArgSemSAT, and LabSATSolver in first, second and third place, respectively, we notice that CoQuiAAS and ArgSemSAT are in this year the winners of two tracks and ArgSemSAT is second-best in another track, while for the remaining semantics other AF solvers, mainly newcomers, have best performance.

Goal of this sub-section is to (qualitatively) compare the award winners of this year's event to the best solvers in the past competition on common tracks. The comparison is done using the benchmarks from the current competition.

Given that the first competition awarded only global results, we applied the Borda count to the tracks of 2015 to get track winners. Thus, the 2015 (version of the) solvers ASPARTIX-D, ArgSemSAT, again ASPARTIX-D, and CoQuiAAs have been run for **CO**, **PR**, **ST**, and **GR** semantics, respectively. Such additional

Solver	**Points**	Time	Correct	Wrong	TO	Other	USC (u)
ArgSemSAT	1146	36607.37	1146	0	234	20	8 (0)
argmat-sat	1139	25110.57	1139	0	245	16	0
pyglaf	1122	43394.57	1127	1	272	1	5 (5)
argmat-dvisat	1075	28597.16	1075	0	307	18	2 (2)
cegartix	1075	58263.31	1075	0	302	23	0
goDIAMOND	1014	51717.30	1069	11	289	31	0
ArgTools	898	53147.54	898	0	501	1	0
ConArg	773	48197.84	773	0	433	194	1 (0)
heureka	745	19691.87	745	0	655	0	0
argmat-mpg	745	30744.76	745	0	470	185	0
EqArgSolver	652	6930.97	652	0	139	609	0
CoQuiAAS	-863	7756.35	477	268	228	427	0
gg-sts	-1107	32999.15	678	357	285	80	2 (1)

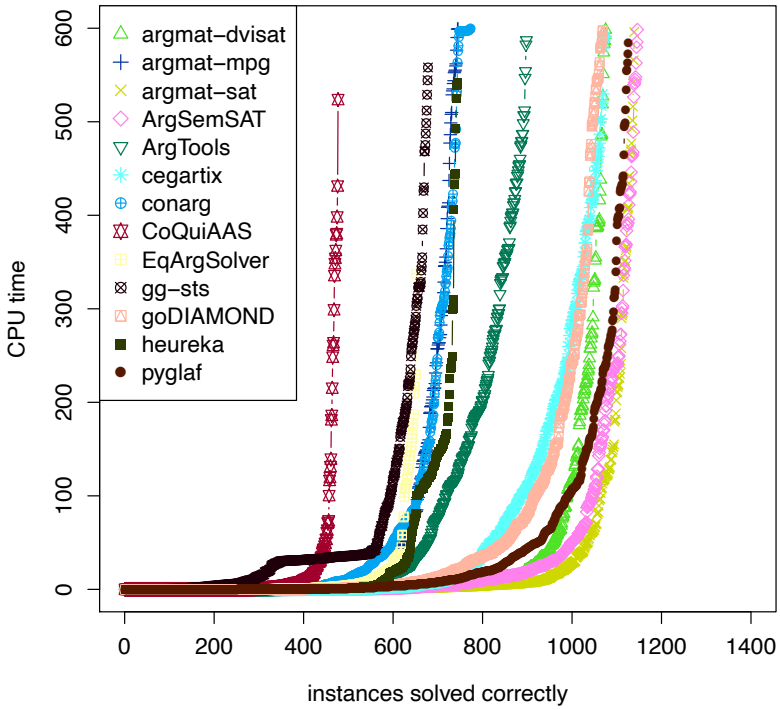

Figure 3: **PR** track: Ranking of solvers (top). Cactus plot of runtimes (bottom).

Solver	**Points**	Time	Correct	Wrong	TO	Other	USC (u)
pyglaf	1183	47155.98	1183	0	217	0	0
goDIAMOND	1143	30116.76	1143	0	224	33	5 (0)
argmat-sat	1129	22087.70	1129	0	247	24	0
cegartix	1102	33963.81	1102	0	283	15	1 (0)
argmat-mpg	1073	52284.56	1073	0	311	16	1 (1)
argmat-dvisat	1039	22591.20	1039	0	334	27	1 (0)
ConArg	1002	58792.29	1002	0	348	50	0
heureka	938	29417.69	938	0	439	23	0
ArgSemSAT	888	23200.99	888	0	291	221	1 (0)
ArgTools	687	45465.87	917	46	316	121	0
EqArgSolver	558	7820.17	558	0	118	724	0
argmat-clpb	135	8840.31	135	0	1133	132	0
CoQuiAAS	-299	13647.26	821	224	297	58	0
gg-sts	-1193	19037.19	782	395	187	36	1 (0)

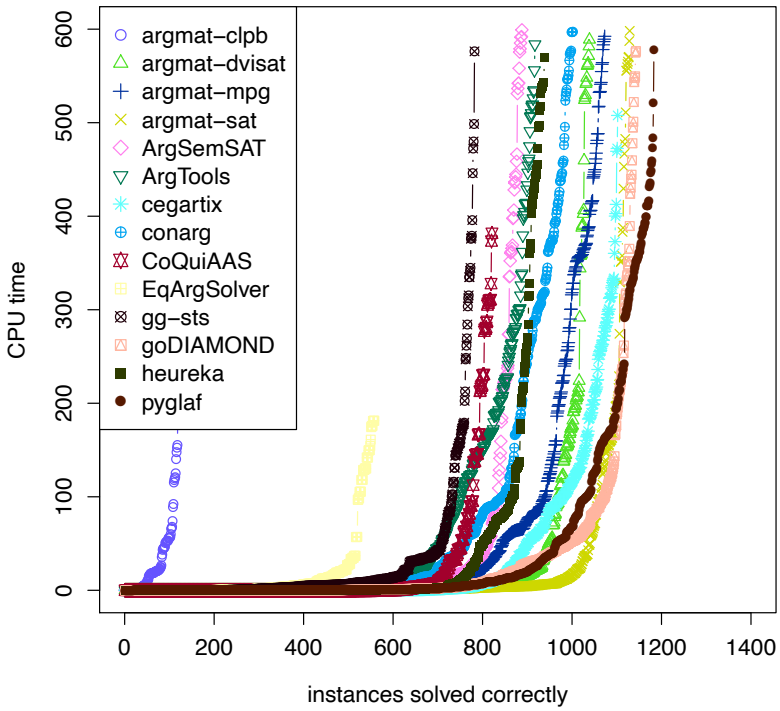

Figure 4: **ST** track: Ranking of solvers (top). Cactus plot of runtimes (bottom).

Solver	**Points**	Time	Correct	Wrong	TO	Other	USC (u)
argmat-sat	1164	26043.50	1164	0	236	0	4 (1)
ArgSemSAT	1113	38816.07	1113	0	264	23	3 (0)
cegartix	1091	62543.78	1091	0	282	27	8 (0)
pyglaf	1047	41378.28	1047	0	349	4	1 (0)
goDIAMOND	1032	57957.15	1032	0	323	45	0
argmat-mpg	755	11464.36	755	0	419	226	3 (3)
ConArg	668	38572.13	668	0	437	295	24 (24)
ArgTools	268	52108.16	568	60	614	158	0
gg-sts	-1321	22846.63	564	377	237	222	8 (2)
CoQuiAAS	-1642	4855.65	218	372	215	595	0

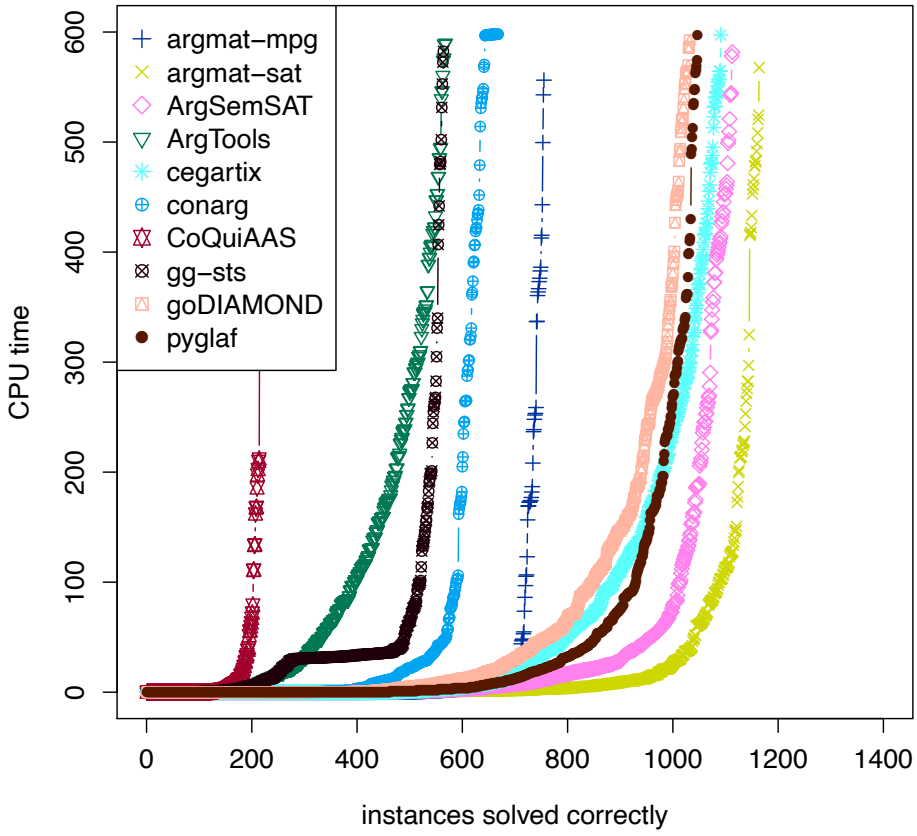

Figure 5: **SST** track: Ranking of solvers (top). Cactus plot of runtimes (bottom).

Solver	**Points**	Time	Correct	Wrong	TO	Other	USC (u)
argmat-sat	1065	19948.06	1065	0	332	3	50 (1)
pyglaf	909	32019.47	909	0	488	3	2 (0)
cegartix	898	62852.40	898	0	502	0	3 (0)
goDIAMOND	724	31394.75	724	0	629	47	0
ConArg	649	43482.21	649	0	490	261	29 (29)
argmat-mpg	618	8381.57	618	0	396	386	4 (0)
ArgTools	67	9558.97	172	21	1207	0	3 (3)
CoQuiAAS	-305	4162.59	320	125	272	683	0
gg-sts	-1325	8242.35	185	302	654	259	4 (0)

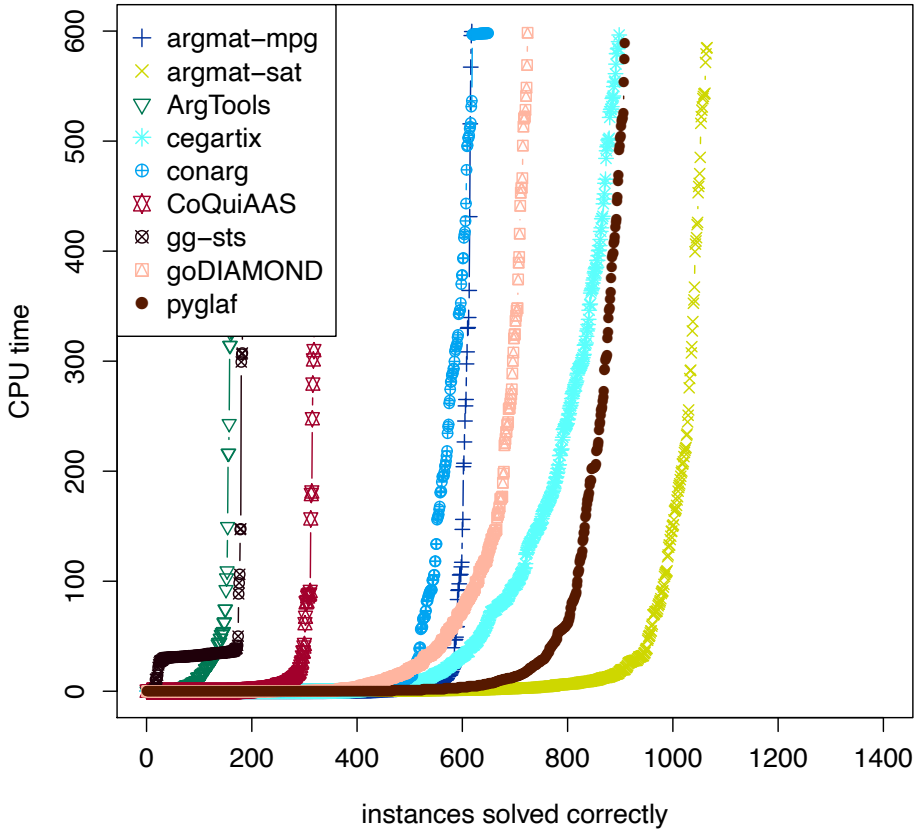

Figure 6: **STG** track: Ranking of solvers (top). Cactus plot of runtimes (bottom).

Solver	**Points**	Time	Correct	Wrong	TO	Other	USC
CoQuiAAS	695	335.85	695	0	3	2	0
cegartix	695	1152.51	695	0	0	5	0
heureka	690	671.37	690	0	8	2	0
goDIAMOND	688	627.43	688	0	12	0	0
pyglaf	683	11595.16	683	0	14	3	0
argmat-dvisat	682	163.80	682	0	4	14	0
argmat-clpb	682	263.21	682	0	4	14	0
EqArgSolver	682	502.80	682	0	18	0	0
argmat-sat	682	504.75	682	0	4	14	0
ArgTools	674	15664.26	674	0	26	0	0
argmat-mpg	662	580.80	662	0	4	34	0
ConArg	588	703.33	588	0	0	112	0
ArgSemSAT	561	11444.85	561	0	119	20	0
gg-sts	-1871	4246.95	264	427	0	9	0

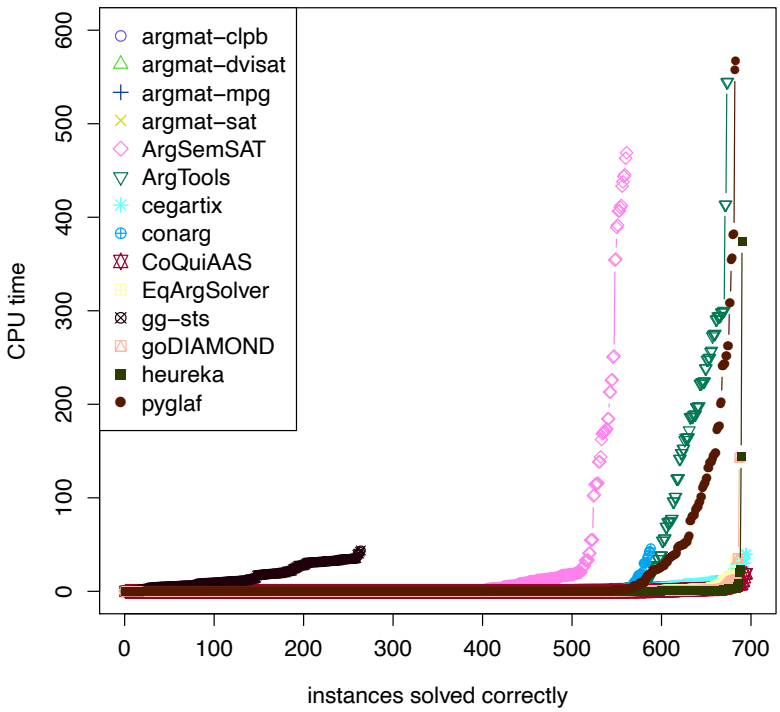

Figure 7: **GR** track: Ranking of solvers (top). Cactus plot of runtimes (bottom).

Solver	**Points**	Time	Correct	Wrong	TO	Other	USC (u)
pyglaf	585	17341.50	585	0	88	27	4 (0)
argmat-dvisat	493	17650.83	493	0	199	8	0
argmat-sat	477	16605.80	477	0	215	8	2 (0)
goDIAMOND	414	22496.34	414	0	270	16	0
cegartix	368	25388.79	548	36	109	7	0
ArgTools	268	20089.40	268	0	385	47	0
argmat-mpg	217	16031.89	217	0	396	87	0
ConArg	181	13254.90	181	0	434	85	1 (0)
CoQuiAAS	-794	2597.28	156	190	94	260	1 (0)
gg-sts	-1050	13379.17	205	251	197	47	2 (0)

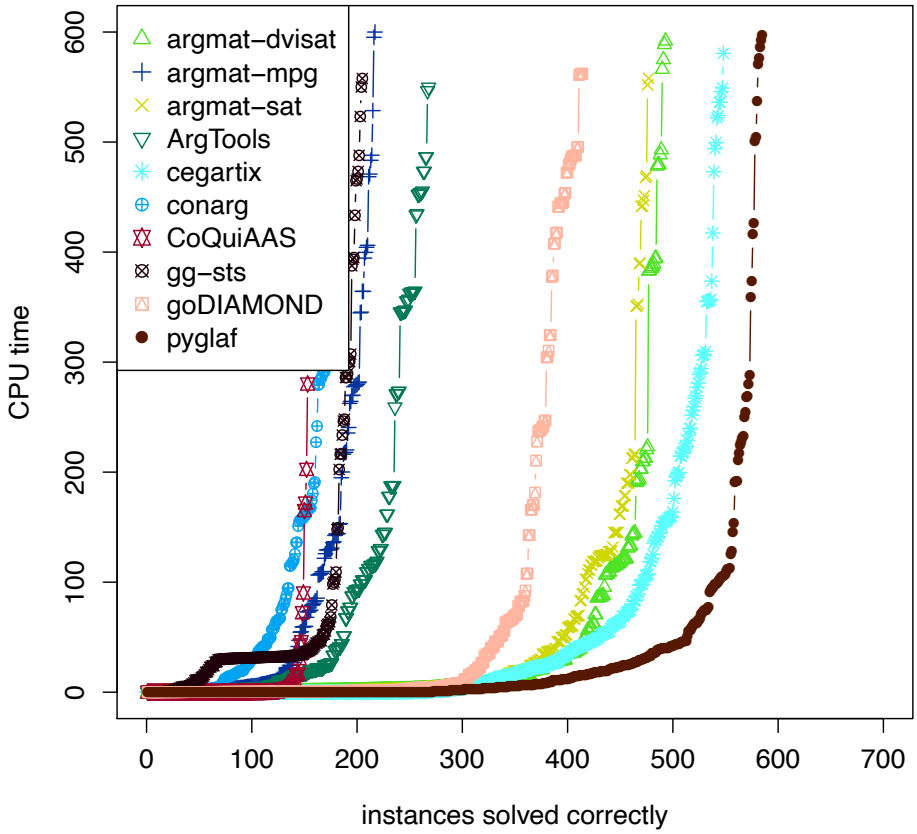

Figure 8: **ID** track: Ranking of solvers (top). Cactus plot of runtimes (bottom).

Solver	**Points**	Time	Correct	Wrong	TO	Other	USC (u)
argmat-dvisat	276	20222.07	276	0	68	6	5 (5)
pyglaf	275	25212.29	275	0	55	20	1 (1)
argmat-sat	271	22441.56	271	0	64	15	3 (3)
cegartix	259	35715.67	259	0	80	11	1 (0)
EqArgSolver	192	6577.89	192	0	32	126	0
ConArg	192	52007.99	192	0	20	138	2 (2)
goDIAMOND	179	28857.58	179	0	52	119	0
argmat-mpg	164	35916.74	164	0	158	28	0
gg-sts	-326	25767.12	144	94	77	35	0
CoQuiAAS	-498	441.22	32	106	43	169	0

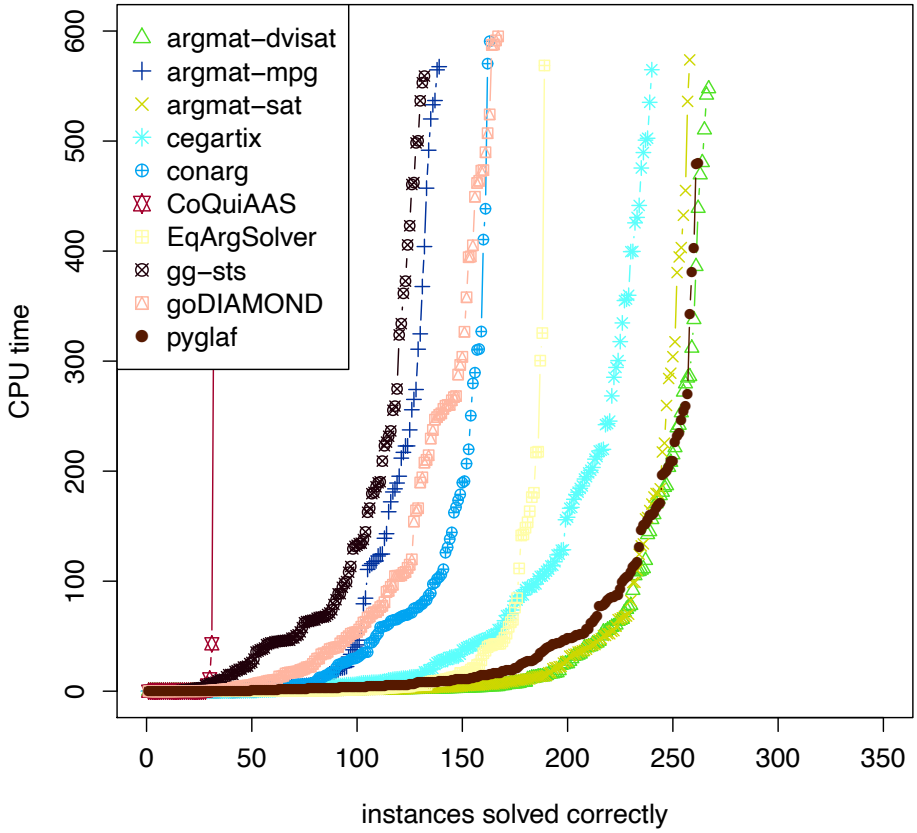

Figure 9: **D3** track: Ranking of solvers (top). Cactus plot of runtimes (bottom).

	CO – pyglaf		*PR – ArgSemSAT*		*ST – pyglaf*		*SST – argmat-sat*		*STG – argmat-sat*		*GR – CoQuiAAS*		*ID – pyglaf*		*D3 – argmat-dvisat*		Total of points	Max. number of points
ABA2AF	57	57	76	76	34	34	118	118	118	118	19	19	59	59	32	32	513	513
AdmBuster	36	39	28	34	24	24	44	44	41	44	17	22	14	22	7	13	211	242
BA	71	86	130	164	146	154	156	174	156	174	5	5	87	87	31	41	782	885
ER	157	181	147	184	124	200	97	168	74	168	90	90	58	84	27	40	774	1115
GroundedGenerator	61	61	58	68	48	48	88	88	88	88	27	27	44	44	24	24	438	448
Planning2AF	81	106	136	168	147	154	169	182	169	182	23	23	91	91	38	43	854	949
SccGenerator	131	132	141	144	167	178	106	110	63	110	58	58	53	55	30	30	749	817
SemBuster	33	57	37	53	35	44	62	62	62	62	32	32	6	31	13	16	280	357
StableGenerator	198	224	126	159	165	208	82	110	73	110	140	140	33	55	21	30	838	1036
Traffic	203	224	121	158	134	142	149	174	148	174	146	146	87	87	22	41	1010	1146
WS	201	233	146	192	159	214	93	170	73	170	138	138	53	85	31	40	894	1242
	1229	1400	1146	1400	1183	1400	1164	1400	1065	1400	695	700	585	700	276	350	7343	8750

Table 11: Points acquired by track winners for each domain.

experiments have been conducted on a separate machine, which is an Intel Xeon CPU E5345, 2.33GHz; 2 processors with each 4 physical cores; no hyperthreading enabled.

Results are reported in Figures 10–16, where each figure contains 4 plots comparing two solvers on two tasks with the following structure: the top and bottom plots are devoted to each task, while the left and right plots present results in terms of box (i.e. a per-instance analysis where a point represents the results of the two compared solvers on the same instance) and cactus (i.e. a cumulative analysis that shows the number of solved instances within a certain CPU time), respectively. Moreover, in the left plots the 2015 solver is on the x-axis and the 2017 solver is on the y-axis, while in the right plots the behavior of the 2015 solver is indicated with a solid blue line with circle, while for the 2017 solver is used a dashed red line with triangles. Figures 10, 12, and 14 contain the analysis for the **DC** and **DS** decision tasks, in top and bottom plots, respectively, of the **CO**, **PR**, and **ST** tracks, while Figures 11, 13, and 15 contain analysis for the **SE** and **EE** enumeration tasks, in top and bottom plots, respectively, of the same semantics. Figure 16 contains the results of the single-status semantics **GR**.

Let us have a closer look on these comparisons. For the **CO** track (Figures 10-11) we can see that pyglaf outperforms ASPARTIX-D on **DS** and **SE** tasks, while it is the opposite for the **EE** task. They perform similarly on the **DC** task. In the **PR** track (Figures 12-13) the general advantages of the 2017 solver winner corresponds to the improvements of the 2017 version of ArgSemSAT in comparison to the 2015 version. About **ST** track (Figures 14-15), we can note that ASPARTIX-D performances are still state of the art, given that it performs (slightly) better on all tasks than pyglaf. Finally, results of the comparison on the **GR** track (Figure 16) show that the performances of the 2017 and 2015 versions of CoQuiAAs are quite similar, still being the state of the art.

To sum up, we can see that in comparison to the best 2015 solvers on a track basis, results are mixed: sometimes the best new solvers perform (much) better than the best of 2015, sometimes is the opposite. When the solver is the same, it is either the case that it improved from the 2015 edition, or basically has similar performance. We think that this, on the one hand, shows that some significant improvements in AF solving have been in place, on the other hand it further confirms that there is space for improvements, by either designing new solutions, or re-importing and improving (ASP-based) solutions already employed.

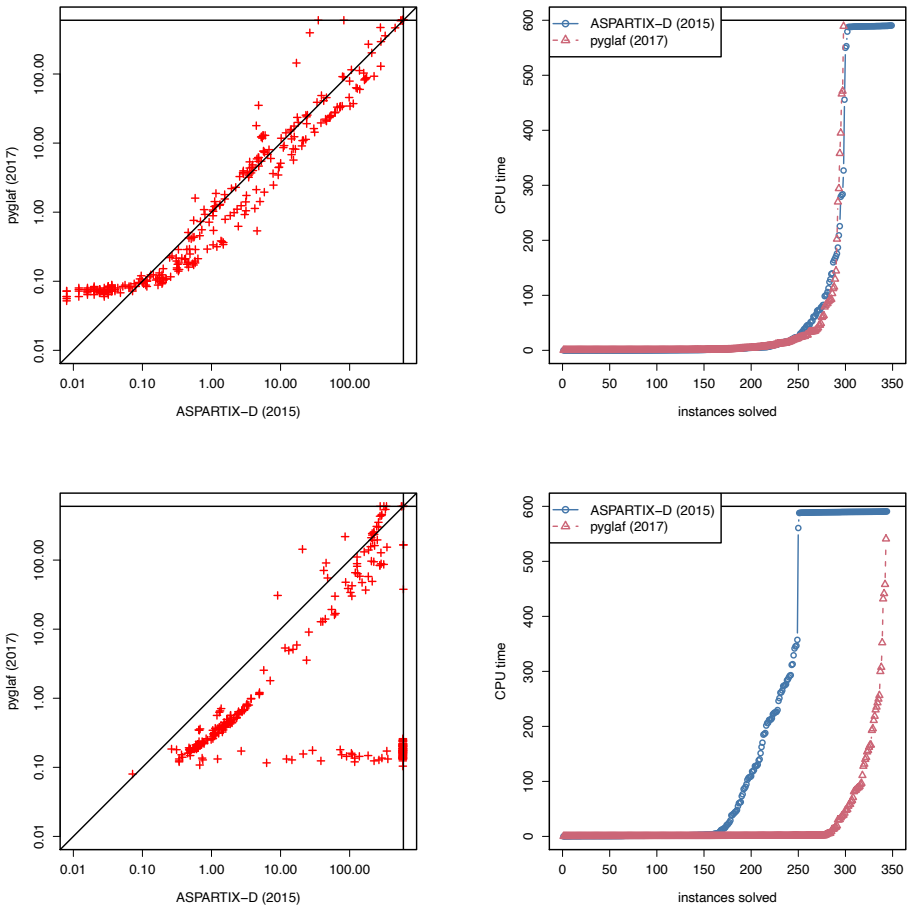

Figure 10: **CO** track, **DC** and **DS** tasks: Comparison between ASPARTIX-D (2015) and pyglaf (2017).

8. Related Competitions

This section discusses how the introduced novelties in this year competition are treated in related competitions. A paragraph is devoted to each of such novelties.

Benchmark suite. For the first time, the competition has featured a *call for benchmarks*, whose goal was to enlarge the set of domains to be included in the evalua-

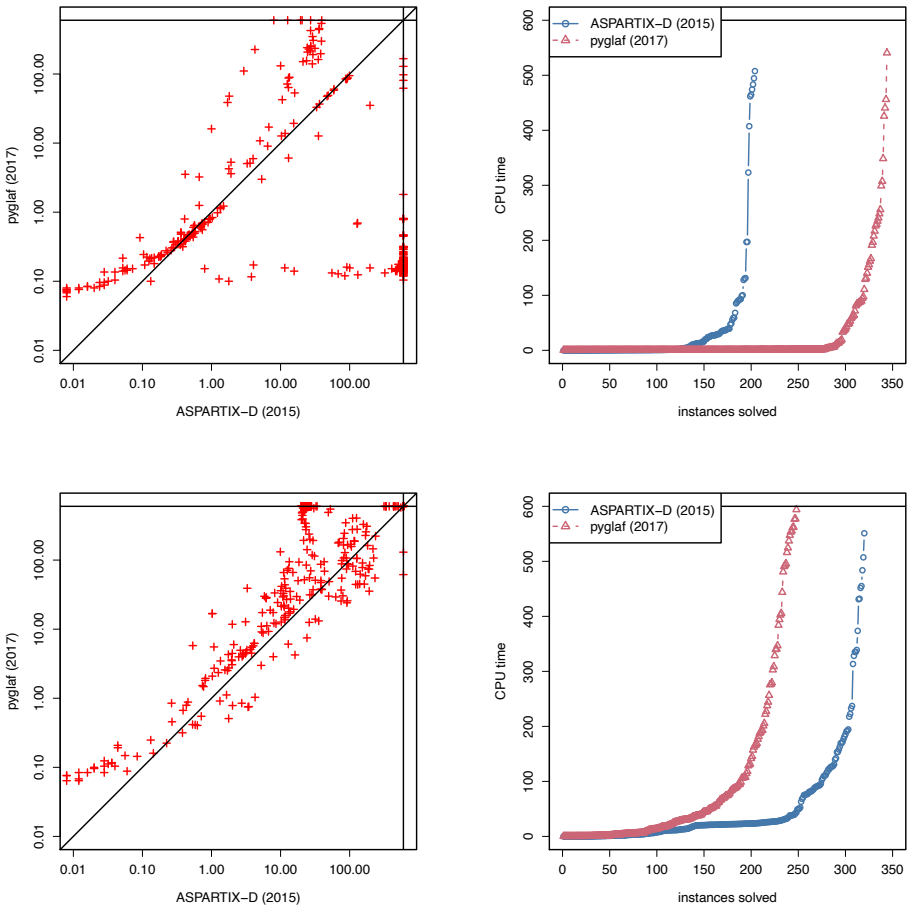

Figure 11: **CO** track, **SE** and **EE** tasks: Comparison between ASPARTIX-D (2015) and pyglaf (2017.)

tion, possibly having a more heterogeneous set. As we can note from Section 4.2, the response from the community was positive. Call for benchmarks are customary in other close competitions, especially in the first events where the benchmark suite has to be developed.

Benchmark selection. Starting from the benchmark suite, the procedure for the selection of instances follows similar procedures employed in SAT and ASP com-

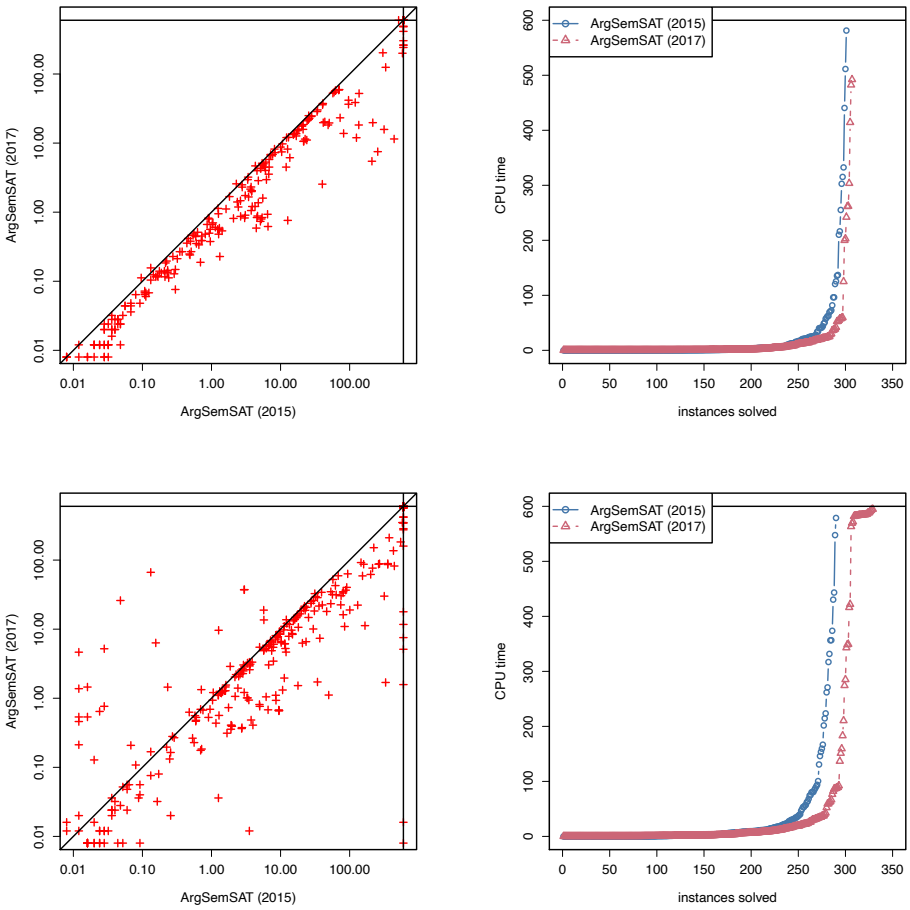

Figure 12: **PR** track, **DC** and **DS** tasks: Comparison between ArgSemSAT (2015) and ArgSemSAT (2017).

petitions (SAT-Comp, 2009; Järvisalo et al., 2012; Balint et al., 2015; Gebser et al., 2017). The main differences in our benchmark selection, some of them due to the intrinsic characteristics of AF, are detailed in the following. Differently from ASP, and similarly to SAT, there is no "non-groundable" hardness category (Section 5.1), given that the benchmarks are inherently ground. Moreover, the variety of semantics and reasoning tasks considered posed additional challenges and decisions to be made for the selection, which are explained in details in Section 5.2

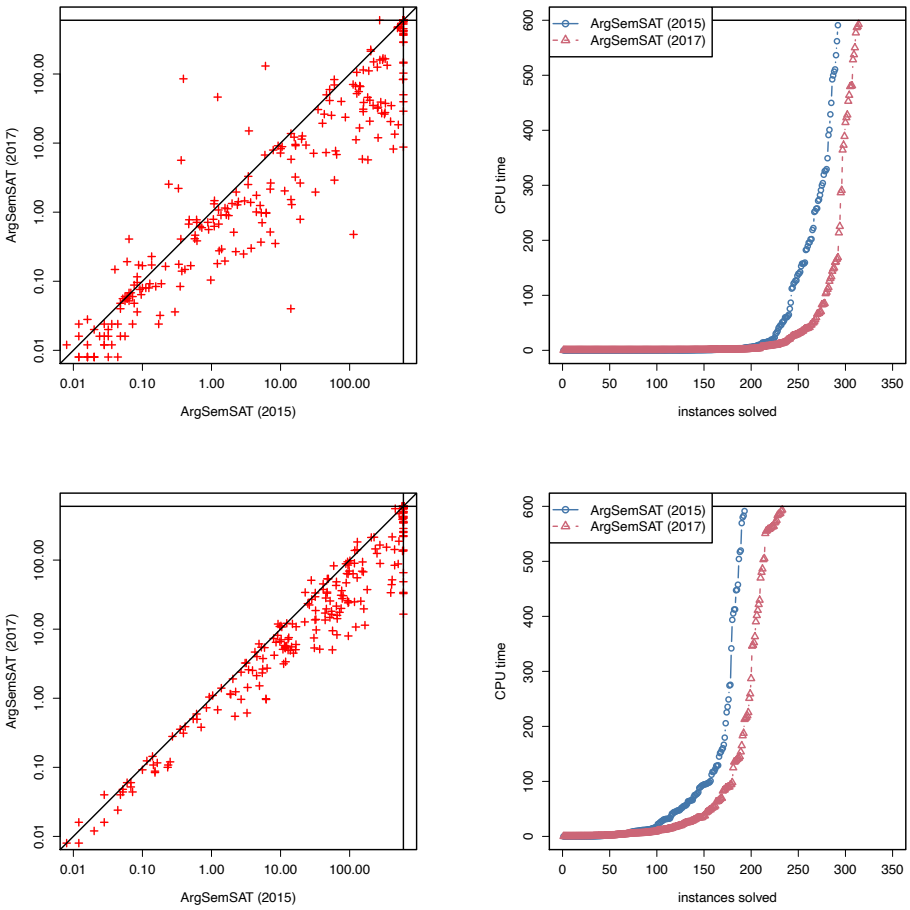

Figure 13: **PR** track, **SE** and **EE** tasks: Comparison between ArgSemSAT (2015) and ArgSemSAT (2017).

and 5.3. As far as solvers employed for the classification of the instances are concerned, in the 2014 IPC competition (Vallati et al., 2015) actual participant systems have been employed for evaluating the empirical hardness of instances. With this choice, the risk is to have a selection biased toward the performance of such systems.

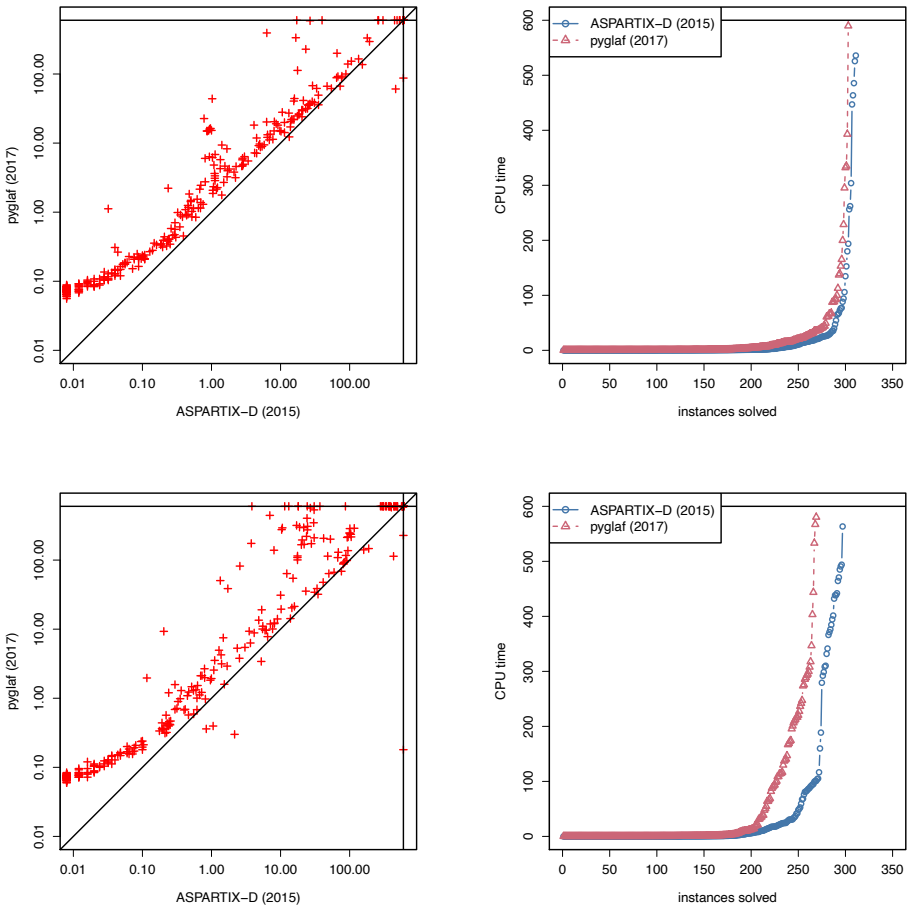

Figure 14: **ST** track, **DC** and **DS** tasks: Comparison between ASPARTIX-D (2015) and pyglaf (2017).

Scoring schema. This edition's scoring schema put focus on correctness by giving a high penalty to incorrect solutions. In the following, we briefly overview the general scoring rules employed in most recent related competitions, even if the details usually change from different events. In the SAT competitions, the total number of solved instances is the main metric to award winners in the tracks. A solver is disqualified in a track if it returns a wrong answer, or a wrong certificate for SAT instances. Considering the last ASP competitions, instead, on Decision

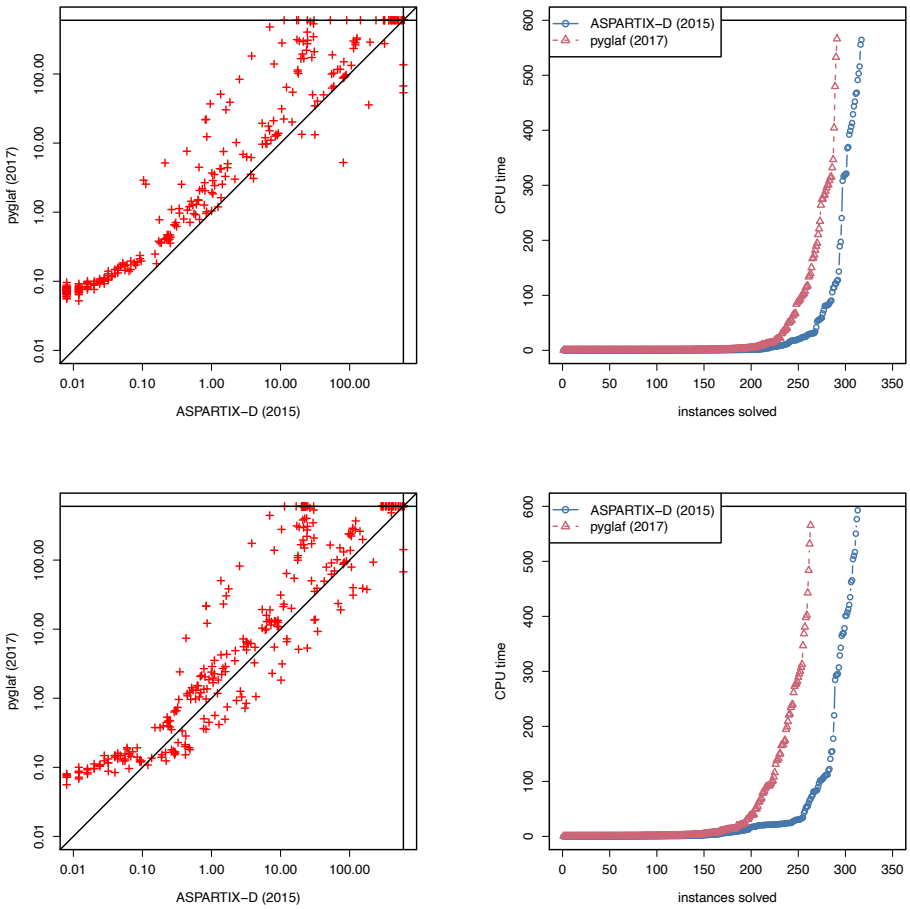

Figure 15: **ST** track, **SE** and **EE** tasks: Comparison between ASPARTIX-D (2015) and pyglaf (2017).

and Query problems a solver can be disqualified for the same reasons, but the disqualification is applied to the domain the instance belongs. The score of each domain on such problems is computed by means of number of solved instances, and ties are broken with the cumulative times of solved instances, while for optimization problems a score based on the "quality" of returned solution and related ranking of solvers is considered. Optimization issues are not considered in ICCMA. The global score then sums the score of each domain. In the IPCs, the

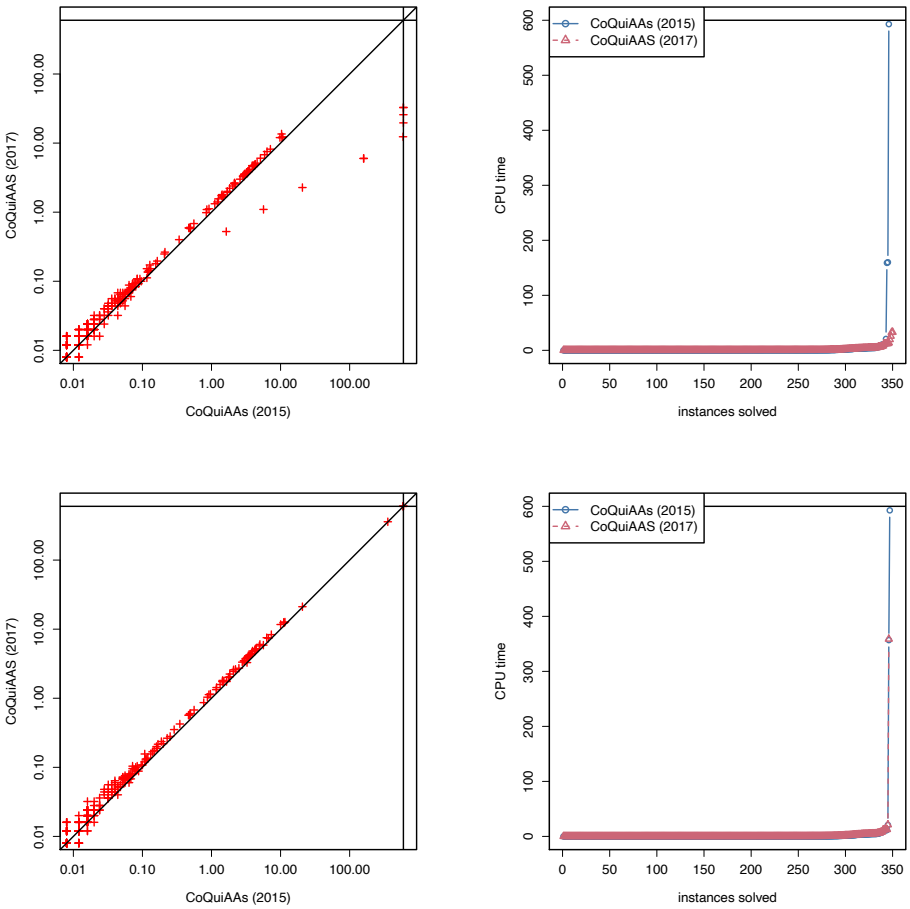

Figure 16: **GR** track, **DC** and **SE** tasks: Comparison between CoQuiAAs (2015) and CoQuiAAs (2017).

two main metrics for scoring planners are the solving times and the "quality" of returned plan. In the deterministic track of the 2014 IPC competition focus was put toward plan's quality. In "optimal" tracks, only optimal solutions were taken into account: a non-optimal solution disqualified a solver from a domain, and if this happens in two domains the planner is disqualified from the track. In IPC "satisfying" tracks, instead, the quality of the returned plans is taken into account. Score of a solver in a track is the sum of the scores in each domain constituting a

track. Similar to our competition, the SMT competitions employ a "per-division" constant penalty for erroneous results (see, e.g. (Cok et al., 2014)). For each division, if it contains a wrong answer, a penalty based on the number of instances in the division is computed; instead, a positive score defined as a function of the number of correctly solved instances and total number of evaluated instances in computed. The global ranking for each track is given by the sum of the results in all divisions.

Special tracks. Among the "most common" special tracks, we mention the "Marathon" and "Parallel" tracks. The Marathon track has been introduced in the 2006 QBF Competition (QBF-Comp, 2006), and then used since 2015; it has been also run in the 2015 ASP Competition (Gebser et al., 2017). In this track the best solvers of the "Regular" track are given more time (usually about one order of magnitude more) to solve (a selection of) the benchmarks that were not solved in the Regular track, in order to test their behavior when more time is given, and ultimately the impact of time limits on performance results. The Parallel track, instead, allows solvers to rely on multiple processors/cores for their computation. This track is in place in several related competitions, e.g. SAT and ASP competitions. The Dung's Triathlon track we have introduced in ICCMA'17, differently from these kinds of tracks, is made of a combination of tasks employed in the competition, instead of strengthening a particular aspect.

9. Conclusions, Lessons Learned, and Future Developments

In this report we have presented the design and results of the Second International Competition on Computational Models of Argumentation (ICCMA'17). We have focused in particular on the novelties that have been introduced in comparison to the first edition in 2015. As far as the results are concerned, the fact that about 2/3 of the tracks have been won by solvers newly introduced at ICCMA'17 shows that the field of computational models of argumentation is not only vibrant but also highly amenable for further improvements and innovation. In particular, pyglaf (winner of 3 tracks) uses a novel approach based on reduction to circumscription.

In the following, we outline some of the lessons that we have learned while organizing the competition, and possible suggestions for the chairs of the third event that will take place in 2019:

More variety in solving approaches. The results of the competition indicate that even more variety of solving techniques can be fruitful for the development

of the field. This is related in particular to pyglaf, but not only, e.g. ASPrMin has the best performance on the task it can deal with (**EE-PR**). Also portfolio-based approaches, here followed by the Chimærarg solver, could be developed more, possibly building on current work, e.g. (Vallati et al., 2017, 2018); in related competitions, such portfolio-based approaches won some of the categories, e.g. the multi-engine ME-ASP ASP solver (Maratea et al., 2014) ver. 2 won the single processor category of the 5th ASP Competition (Calimeri et al., 2016). Other alternatives can include the employment of QBFs, as e.g. the authors of gg-sts are planning (see, (Jahunen and Tasharrofi, 2017)), and for which implementations are already in place (Diller et al., 2015).

Maintain benchmark classification and selection. Our benchmark classification and selection allowed to run the competition on a "meaningful" set of benchmarks with a high variety of expected hardness, differently from ICCMA'15, where a significant number of the instances were easy. This helped in particular on the new domains which were unseen to solvers. Thus, also considering that, in future editions, we expect more new domains, we think that ICCMA should stick to a guided instance selection process as described in this report.

More variety in benchmarks. The community should aim for benchmarks from more real-world domains to be included in future benchmark suites. In particular, the existing formalisms that use instantiations of AFs such as structured argumentation formalisms or defeasible knowledge bases could be explored towards obtaining new AF benchmarks. An example was recently provided by Yun et al. (2017), where AFs are instantiated with existential rules in a Semantic Web context.

Verification of answers. As we have seen before, the verification of answers has been a challenging issue. For decision tasks, which involve the computation of (at most) a single extension, we have used an ASP encoding for the verification of correctness. The resulting procedure was not particularly fast, but practical, given that we managed to check all outputs. When the verification of answers in enumeration tasks comes into play, the situation is more difficult. Some possible directions that could be pursued in the future are: (a) an extension of the approach for single extension, i.e. having an ASP encoding where answer sets corresponds to extensions, (b) a more practical and a-priori solution, by aiming at selecting benchmarks with a limited number of solutions, and/or (c) another practical approach where only part of the extensions (e.g., randomly picked) is selected for verifying correctness.

Output format. On the more technical side, the output format adopted from the first edition of the competition turned out to be unfavourable for checking solutions of the **EE** task. In particular, the fact that the solution is to be provided in a single line makes the processing of large solutions with customary text oriented tools quite cumbersome. Introducing line breaks as well as requiring the extensions to be in a format more amenable for verification could be beneficial for the verification process in the next edition.

Acknowledgments. We thank the Center for Information Services and High Performance Computing (ZIH) at TU Dresden for generous allocation of computer time. We also thank Peter Steinke and Norbert Manthey for providing the scripts to run the competition on the cluster, as well as Christian Al-Rabbaa for implementing the evaluation scripts. We finally thank the TAFA'17 officials for the co-location of the event, and all ICCMA'17 contributors, who worked hard on their systems and benchmarks, and made the competition possible.

This work has been supported by the German Research Foundation (DFG) (project BR 1817/7-2) and the Austrian Science Fund (FWF) (projects I2854 and Y698).

References

Alviano, M., 2017. Model enumeration in propositional circumscription via unsatisfiable core analysis. Theory and Practice of Logic Programming 17 (5-6), 708–725.

Amendola, G., Dodaro, C., Ricca, F., 2016. ASPQ: An ASP-based 2QBF solver. In: Lonsing, F., Seidl, M. (Eds.), Proceedings of the 4th International Workshop on Quantified Boolean Formulas (QBF 2016) co-located with 19th International Conference on Theory and Applications of Satisfiability Testing (SAT 2016),. Vol. 1719 of CEUR Workshop Proceedings. CEUR-WS.org, pp. 49–54.

Arora, S., Barak, B., 2009. Computational Complexity – A Modern Approach. Cambridge University Press.
URL `http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264`

Atkinson, K., Baroni, P., Giacomin, M., Hunter, A., Prakken, H., Reed, C., Simari, G., Thimm, M., Villata, S., 2017. Towards artificial argumentation. AI Magazine 38 (3), 25–36.

Audemard, G., Simon, L., 2009. Predicting learnt clauses quality in modern SAT solvers. In: Boutilier, C. (Ed.), Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009). pp. 399–404.

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

Barabasi, A. L., Albert, R., 1999. Emergence of scaling in random networks. Science 286, 509–512.

Baroni, P., Caminada, M., Giacomin, M., 2011. An introduction to argumentation semantics. The Knowledge Engineering Review 26 (4), 365–410.

Baroni, P., Caminada, M., Giacomin, M., 2018. Abstract argumentation frameworks and their semantics. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (Eds.), Handbook of Formal Argumentation. College Publications, Ch. 4, pp. 159–236.

Beierle, C., Brons, F., Potyka, N., 2015. A software system using a SAT solver for reasoning under complete, stable, preferred, and grounded argumentation semantics. In: Hölldobler, S., Krötzsch, M., Peñaloza, R., Rudolph, S. (Eds.), Proceedings of the 38th Annual German Conference on AI (KI 2015). Vol. 9324 of Lecture Notes in Computer Science. Springer, pp. 241–248.

Bench-Capon, T. J. M., Dunne, P. E., 2007. Argumentation in artificial intelligence. Artificial Intelligence 171 (10-15), 619–641.

Bistarelli, S., Rossi, F., Santini, F., 2014. Benchmarking hard problems in random abstract afs: The stable semantics. In: Parsons, S., Oren, N., Reed, C., Cerutti, F. (Eds.), Proceedings of the 5th International Conference on Computational Models of Argument (COMMA 2014). Vol. 266 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 153–160.

Bistarelli, S., Rossi, F., Santini, F., 2015. A comparative test on the enumeration of extensions in abstract argumentation. Fundamenta Informaticae 140 (3-4), 263–278.

Bistarelli, S., Rossi, F., Santini, F., 2018. Not only size, but also shape counts: abstract argumentation solvers are benchmark-sensitive. Journal of Logic and Computation 28 (1), 85–117.

Bistarelli, S., Santini, F., 2011. Conarg: A constraint-based computational framework for argumentation systems. In: Proceedings of the IEEE 23rd International Conference on Tools with Artificial Intelligence (ICTAI 2011). IEEE Computer Society, pp. 605–612.

Bogaerts, B., Janhunen, T., Tasharrofi, S., 2016. Declarative solver development: Case studies. In: Baral, C., Delgrande, J. P., Wolter, F. (Eds.), Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR 2016). AAAI Press, pp. 74–83.

Calimeri, F., Gebser, M., Maratea, M., Ricca, F., 2016. Design and results of the fifth answer set programming competition. Artificial Intelligence 231, 151–181.

Caminada, M., 2014. Strong admissibility revisited. In: Parsons, S., Oren, N., Reed, C., Cerutti, F. (Eds.), Proceedings of the 5th International Conference on Computational Models of Argument (COMMA 2014). Vol. 266 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 197–208.

Caminada, M., Carnielli, W. A., Dunne, P. E., 2012. Semi-stable semantics. Journal of Logic and Computation 22 (5), 1207–1254.

Caminada, M., Sá, S., Alcântara, J., Dvořák, W., 2015. On the equivalence between logic programming semantics and argumentation semantics. International Journal of Approximate Reasoning 58, 87–111.

Caminada, M. W., Verheij, B., 2010. On the existence of semi-stable extensions. In: Danoy, G., Seredynski, M., Booth, R., Gateau, B., Jars, I., Khadraoui, D. (Eds.), Proceedings of the 22nd Benelux Conference on Artificial Intelligence (BNAIC 2010). Available at `http://bnaic2010.uni.lu/proceedings.html`.

Cerutti, F., Giacomin, M., Vallati, M., 2014a. ArgSemSAT: Solving argumentation problems using SAT. In: Parsons, S., Oren, N., Reed, C., Cerutti, F. (Eds.), Proceedings of the 5th International Conference on Computational Models of Argument (COMMA 2014). Vol. 266 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 455–456.

Cerutti, F., Giacomin, M., Vallati, M., 2016a. Generating structured argumentation frameworks: AFBenchGen2. In: Baroni, P., Gordon, T. F., Scheffler, T.,

Stede, M. (Eds.), Proceedings of the 6th International Conference on Computational Models of Argument (COMMA 2016). Vol. 287 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 467–468.

Cerutti, F., Oren, N., Strass, H., Thimm, M., Vallati, M., 2014b. A benchmark framework for a computational argumentation competition. In: Parsons, S., Oren, N., Reed, C., Cerutti, F. (Eds.), Proceedings of the 5th International Conference on Computational Models of Argument (COMMA 2014). Vol. 266 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 459–460.

Cerutti, F., Vallati, M., Giacomin, M., 2016b. Where are we now? state of the art and future trends of solvers for hard argumentation problems. In: Baroni, P., Gordon, T. F., Scheffler, T., Stede, M. (Eds.), Proceedings of the 6th International Conference on Computational Models of Argument (COMMA 2016). Vol. 287 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 207–218.

Cerutti, F., Vallati, M., Giacomin, M., 2018. On the impact of configuration on abstract argumentation automated reasoning. International Journal of Approximate Reasoning 92, 120–138.

Charwat, G., Dvořák, W., Gaggl, S. A., Wallner, J. P., Woltran, S., 2015. Methods for solving reasoning problems in abstract argumentation - A survey. Artificial Intelligence 220, 28–63.

Cok, D. R., Déharbe, D., Weber, T., 2014. The 2014 SMT competition. Journal on Satisfiability, Boolean Modeling and Computation 9, 207–242.

Cyras, K., Fan, X., Schulz, C., Toni, F., 2018. Assumption-based argumentation: Disputes, explanations, preferences. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (Eds.), Handbook of Formal Argumentation. College Publications, Ch. 7, pp. 365—408.

Diller, M., Wallner, J. P., Woltran, S., 2015. Reasoning in abstract dialectical frameworks using Quantified Boolean Formulas. Argument & Computation 6 (2), 149–177.

Dimopoulos, Y., Torres, A., 1996. Graph theoretical structures in logic programs and default theories. Theoretical Computer Science 170 (1-2), 209–244.

Dung, P. M., 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77 (2), 321–358.

Dung, P. M., Mancarella, P., Toni, F., 2007. Computing ideal sceptical argumentation. Artificial Intelligence 171 (10–15), 642–674.

Dunne, P. E., 2009. The computational complexity of ideal semantics. Artificial Intelligence 173 (18), 1559–1591.

Dunne, P. E., Bench-Capon, T. J. M., 2002. Coherence in finite argument systems. Artificial Intelligence 141 (1/2), 187–203.

Dunne, P. E., Dvořák, W., Woltran, S., 2013. Parametric properties of ideal semantics. Artificial Intelligence 202, 1–28.

Dvořák, W., Järvisalo, M., Wallner, J. P., Woltran, S., 2014. Complexity-sensitive decision procedures for abstract argumentation. Artificial Intelligence 206, 53–78.

Dvořák, W., Dunne, P. E., 2018. Computational problems in formal argumentation and their complexity. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (Eds.), Handbook of Formal Argumentation. College Publications, Ch. 14, pp. 631–687, also appears in IfCoLog Journal of Logics and their Applications 4(8):2557–2622.

Dvořák, W., Woltran, S., 2010. Complexity of semi-stable and stage semantics in argumentation frameworks. Information Processing Letters 110 (11), 425–430.

Eén, N., Sörensson, N., 2003. An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (Eds.), Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT 2003). Selected Revised Papers. Vol. 2919 of Lecture Notes in Computer Science. Springer, pp. 502–518.

Egly, U., Gaggl, S. A., Woltran, S., 2010. Answer-set programming encodings for argumentation frameworks. Argument & Computation 1 (2), 147–177.

Ellmauthaler, S., Strass, H., 2014. The DIAMOND system for computing with abstract dialectical frameworks. In: Parsons, S., Oren, N., Reed, C., Cerutti, F. (Eds.), Proceedings of the 5th International Conference on Computational Models of Argument (COMMA 2014). Vol. 266 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 233–240.

Erdös, P., Rényi, A., 1959. On random graphs I. Publicationes Mathematicae Debrecen 6, 290–297.

Faber, W., Vallati, M., Cerutti, F., Giacomin, M., 2016. Solving set optimization problems by cardinality optimization with an application to argumentation. In: Kaminka, G. A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., van Harmelen, F. (Eds.), Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016). Vol. 285 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 966–973.

Gabbay, D. M., Rodrigues, O., 2016. Further applications of the Gabbay-Rodrigues iteration schema in argumentation and revision theories. In: Beierle, C., Brewka, G., Thimm, M. (Eds.), Computational Models of Rationality, Essays dedicated to Gabriele Kern-Isberner on the occasion of her 60th birthday. College Publications, pp. 392–408.

Gaggl, S. A., Linsbichler, T., Maratea, M., Woltran, S., 2016. Introducing the second international competition on computational models of argumentation. In: Thimm, M., Cerutti, F., Strass, H., Vallati, M. (Eds.), Proceedings of the 1st International Workshop on Systems and Algorithms for Formal Argumentation (SAFA 2016) co-located with the 6th International Conference on Computational Models of Argument (COMMA 2016). Vol. 1672 of CEUR Workshop Proceedings. CEUR-WS.org, pp. 4–9.

Gaggl, S. A., Linsbichler, T., Maratea, M., Woltran, S., 2018. Summary report of the second international competition on computational models of argumentation. AI Magazine. To appear.

Gaggl, S. A., Manthey, N., Ronca, A., Wallner, J. P., Woltran, S., 2015. Improved answer-set programming encodings for abstract argumentation. Theory and Practice of Logic Programming 15 (4-5), 434–448.

Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T., 2014. Clingo = ASP + control: Preliminary report. CoRR abs/1405.3694.

Gebser, M., Maratea, M., Ricca, F., 2017. The sixth answer set programming competition. Journal of Artificial Intelligence Research 60, 41–95.

Geilen, N., Thimm, M., 2017. Heureka: A general heuristic backtracking solver for abstract argumentation. In: Black, E., Modgil, S., Oren, N. (Eds.), Proceedings of the 4th International Workshop on Theory and Applications of Formal

Argumentation (TAFA 2017). Revised Selected Papers. Vol. 10757 of Lecture Notes in Computer Science. Springer, pp. 143–149.

Giunchiglia, E., Lierler, Y., Maratea, M., 2006. Answer set programming based on propositional satisfiability. Journal of Automated Reasoning 36 (4), 345–377.

ICCMA'17-Soldes, 2017. http://www.argumentationcompetition.org/2017/submissions.html.

ICCMA'17-Solreq, 2017. http://www.argumentationcompetition.org/2017/SolverRequirements.pdf.

Jahunen, T., Tasharrofi, S., 2017. http://www.argumentationcompetition.org/2017/gg-sts.pdf.

Järvisalo, M., Berre, D. L., Roussel, O., Simon, L., 2012. The international SAT solver competitions. AI Magazine 33 (1).

Johnson, D. S., Papadimitriou, C. H., Yannakakis, M., 1988. On generating all maximal independent sets. Information Processing Letters 27 (3), 119–123.

Kröll, M., Pichler, R., Woltran, S., 2017. On the complexity of enumerating the extensions of abstract argumentation frameworks. In: Sierra, C. (Ed.), Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017). ijcai.org, pp. 1145–1152.

Lagniez, J., Lonca, E., Mailly, J., 2015. CoQuiAAS: A constraint-based quick abstract argumentation solver. In: Proceedings of the 27th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2015). IEEE Computer Society, pp. 928–935.

Lehtonen, T., Wallner, J. P., Järvisalo, M., 2017. From structured to abstract argumentation: Assumption-based acceptance via AF reasoning. In: Antonucci, A., Cholvy, L., Papini, O. (Eds.), Proceedings of the 14th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2017). Vol. 10369 of Lecture Notes in Computer Science. Springer, pp. 57–68.

Maratea, M., Pulina, L., Ricca, F., 2014. A multi-engine approach to answer-set programming. Theory and Practice of Logic Programming 14 (6), 841–868.

Modgil, S., Prakken, H., 2014. The *ASPIC*$^+$ framework for structured argumentation: A tutorial. Argument & Computation 5 (1), 31–62.

Nofal, S., Atkinson, K., Dunne, P. E., 2016. Looking-ahead in backtracking algorithms for abstract argumentation. International Journal on Approximate Reasoning 78, 265–282.

Pu, F., Luo, G., Jiang, Z., 2017. Encoding argumentation semantics by Boolean algebra. IEICE Transactions 100-D (4), 838–848.

Pulina, L., 2016. The ninth QBF solvers evaluation - preliminary report. In: Lonsing, F., Seidl, M. (Eds.), Proceedings of the 4th International Workshop on Quantified Boolean Formulas (QBF 2016) co-located with 19th International Conference on Theory and Applications of Satisfiability Testing (SAT 2016). Vol. 1719 of CEUR Workshop Proceedings. CEUR-WS.org, pp. 1–13.

QBF-Comp, 2006. QBF Evaluation 2006. `http://www.qbflib.org`.

SAT-Comp, 2009. SAT Competition 2009. `http://www.satcompetition.org/2009/`.

Sideris, A., Dimopoulos, Y., 2010. Constraint propagation in propositional planning. In: Brafman, R. I., Geffner, H., Hoffmann, J., Kautz, H. A. (Eds.), Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS 2010). AAAI, pp. 153–160.

Strozecki, Y., 2010. Enumeration complexity and matroid decomposition. Ph.D. thesis, Universit'e Paris Diderot – Paris 7.

Thimm, M., Villata, S., 2015. System descriptions of the first international competition on computational models of argumentation (ICCMA'15). CoRR abs/1510.05373.
URL `http://arxiv.org/abs/1510.05373`

Thimm, M., Villata, S., 2017. The first international competition on computational models of argumentation: Results and analysis. Artificial Intelligence 252, 267–294.

Thimm, M., Villata, S., Cerutti, F., Oren, N., Strass, H., Vallati, M., April 2016. Summary report of the first international competition on computational models of argumentation. AI Magazine 37 (1), 102–104.

Toni, F., 2014. A tutorial on assumption-based argumentation. Argument & Computation 5 (1), 89–117.

Vallati, M., Cerutti, F., Giacomin, M., 2017. On the combination of argumentation solvers into parallel portfolios. In: Peng, W., Alahakoon, D., Li, X. (Eds.), Advances in Artificial Intelligence - Proceedings of the 30th Australasian Joint Conference (AI 2017). Vol. 10400 of Lecture Notes in Computer Science. Springer, pp. 315–327.

Vallati, M., Cerutti, F., Giacomin, M., 2018. Predictive models and abstract argumentation: The case of high-complexity semantics. Knowledge Engineering Review. To appear.

Vallati, M., Chrpa, L., Grzes, M., McCluskey, T. L., Roberts, M., Sanner, S., 2015. The 2014 international planning competition: Progress and trends. AI Magazine 36 (3), 90–98.

Verheij, B., 1996. Two approaches to dialectical argumentation: Admissible sets and argumentation stages. In: Proceedings of the 8th Dutch Conference on Artificial Intelligence (NAIC'96). pp. 357–368.

Watts, D. J., Strogatz, S. H., 1998. Collective dynamics of "small-world" networks. Nature 393, 440–442.

Wu, Y., Caminada, M., Gabbay, D. M., 2009. Complete extensions in argumentation coincide with 3-valued stable models in logic programming. Studia Logica 93 (2-3), 383–403.
URL https://doi.org/10.1007/s11225-009-9210-5

Wyner, A. Z., Bench-Capon, T. J. M., Dunne, P. E., Cerutti, F., 2015. Senses of 'argument' in instantiated argumentation frameworks. Argument & Computation 6 (1), 50–72.

Yun, B., Vesic, S., Croitoru, M., Bisquert, P., Thomopoulos, R., 2017. A structural benchmark for logical argumentation frameworks. In: Adams, N. M., Tucker, A., Weston, D. J. (Eds.), Proceedings of the 16th International Symposium on Advances in Intelligent Data Analysis (IDA 2017). Vol. 10584 of Lecture Notes in Computer Science. Springer, pp. 334–346.