Benchmark selection at ICCMA'17

Sarah Alice GAGGL [a], Thomas LINSBICHLER [b], Marco MARATEA [c] and Stefan WOLTRAN [b]

[a] *Computational Logic Group, Technische Universität Dresden, Germany*
[b] *Institute of Information Systems, TU Wien, Austria*
[c] *DIBRIS, University of Genova, Italy*

Abstract. This note introduces all steps performed to implement the benchmarks selection phase at the Second International Competition on Computational Models of Argumentation (ICCMA'17).

1. Competition design

The Second International Competition on Computational Models of Argumentation [Gaggl et al., 2016] evaluates the performance of submitted solvers on reasoning tasks for abstract argumentation frameworks.

The semantics under consideration are complete (**CO**), preferred (**PR**), stable (**ST**), semi-stable (**SST**), stage (**STG**), grounded (**GR**), and ideal (**ID**). The computational problems employed are deciding credulous acceptance of an argument (**DC**), deciding skeptical acceptance of an argument (**DS**), enumerating all extensions (**EE**), and enumerating a single extension (**SE**). A computational problem for a semantics constitutes a task.

For each semantics, all corresponding tasks consolidate to a track. For grounded and ideal semantics, only **DC** and **SE** are employed. A special track, Dung's Triathlon (**D3**) combines the enumeration of grounded, stable, and preferred extension to a single task.

Tasks are grouped according to difficulty of the respective tasks. The classification into groups A to E is based on known complexity results and corroborated by the analysis of the percentage of solved instances in the 2015 edition. The applied grouping is the following:

A: **DS-PR**, **EE-PR**, **EE-CO**
B: **DC-ST**, **DS-ST**, **EE-ST**, **SE-ST**, **DC-PR**, **SE-PR**, **DC-CO**
C: **DS-CO**, **SE-CO**, **DC-GR**, **SE-GR**
D: **DC-ID**, **SE-ID**
E: **SST**-*, **STG**-*

Note that groups D and E include the newly employed semantics. Each group A to C gets its own set of benchmarks, while for groups D and E the same set of benchmarks as for group A is used.

2. Benchmarks selection process

The goal of this phase is to select the instances that are indeed run in the competition. First, the domains are identified based on the benchmark submissions. Then, a set of instances is collected (or generated) for each domain. These instances are subsequently 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 each group of tasks has its own set of benchmarks, 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 tasks use the same benchmark set as group A.

The following subsections present 1) how instances have been collected, 2) how instances have been classified, and 3) how instances have been selected, respectively.

2.1. Benchmark collection

Following the dedicated call for benchmarks, the competition received 6 submissions:

"ABA2AF". Assumption-Based Argumentation Translated to Argumentation Frameworks.

AdmBuster. A benchmark example for (strong) admissibility.

AFBenchGen2. A Generator for Random Argumentation Frameworks:

- Barabasi-Albert
- Erdös-Rényi
- Watts-Strogatz

"Planning2AF". Exploiting Planning Problems for Generating Challenging Abstract Argumentation Frameworks.

SemBuster. A benchmark example for semi-stable semantics.

Traffic. Traffic Networks Become Argumentation Frameworks.

Short descriptions of these benchmarks can be found at `http://www.dbai.tuwien.ac.at/iccma17/submissions.html#benchmarks`.

The generator AFBenchGen2 provides AFs of 3 different graph classes (Barabasi-Albert, Erdös-Rényi, and Watts-Strogatz), each giving rise to a domain of its own. All other submissions make up exactly one domain. Moreover, the generators from ICCMA'15, namely GroundedGenerator, SccGenerator, and StableGenerator, are reused. Detailed descriptions of these generators can be found in [Thimm and Villata, 2017]. To summarize, the following domains are considered:

- ABA2AF
- AdmBuster
- Barabasi-Albert
- Erdös-Rényi
- GroundedGenerator

Table 1. Description of (generated) benchmarks considered for selection.

Domain	No. instances	Parameters
ABA2AF	426	all submitted instances
AdmBuster	13	no. arguments: 1K, 2K, 4K, 6K, 8K, 10K, 20K, 50K, 100K, 200K, 500K, 1000K, 2000K
Barabasi-Albert	500	5 random instances for each (no. arguments, `probCycles`) in $\{20,40,\ldots,200\} \times \{0,0.1,\ldots,0.9\}$
Erdös-Rényi	500	10 random instances for each (no. arguments, `probAttacks`) in $\{100,200,\ldots,500\} \times \{0.1,0.2,\ldots,1.0\}$
GroundedGenerator	50	no. arguments `random`$[100,1500]$; 10 random instances for each `probAttacks` in $\{0.01,0.02,\ldots,0.05\}$
Planning2AF	385	all submitted instances
SccGenerator	600	no. arguments `random`$[100,1500]$; no. SCCs `random`$[1,50]$; 25 random instances for each (`innerAttackProb`, `outerAttackProb`) in $\{0.3,0.4,\ldots,0.7\} \times \{0.05,0.1,0.15,0.2\}$. no. arguments `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	no. arguments: 60, 150, 300, 600, 900, 1200, 1500, 1800, 2400, 3000, 3600, 4200, 4800, 5400, 6000, 7500
StableGenerator	500	no. arguments `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	(no. arguments, k, `baseDegree`, `probCycles`) in $\{100,200,\ldots,500\} \times \log_2(\text{no. arguments}) \cdot \{1,2,3,4\} \times \{0.1,0.3,\ldots,0.9\} \times \{0.1,0.3,0.5,0.7\}$

- Planning2AF
- SccGenerator
- SemBuster
- StableGenerator
- Traffic
- Watts-Strogatz

Given these domains, we collect and generate a total of 3990 instances. Table 1 gives details on the collected benchmarks, stating, for each domain, the number of instances as well as the parameters for generating the instances. If the benchmark submission consisted of a set of instances, we simply consider them all. For domains emerging from submissions of benchmark generators, we try to produce 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 Table 1. In some cases, parameters are chosen randomly from an interval (denoted by `random`$[a,b]$), in other cases all values among a set of values are considered (denoted by $\{v_1,v_2,\ldots,v_n\}$).

2.2. Benchmark classification

In the next step the collected instances are classified with respect to their expected level of difficulty.

To classify the hardness of instances, competitions in other research fields such as SAT, ASP and Planning, 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 (strongly) 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 to use the same set of benchmarks. We identify the following representative tasks which will be used for classification:

- A: **EE-PR**
- B: **EE-ST**
- 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 classification 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. The following solvers from ICCMA'15 are selected (see `http://argumentationcompetition.org/2015/solvers.html` for system descriptions):

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

Both Cegartix and ArgSemSAT implement (iterative) SAT based approaches; CoQuiAAS makes use of Partial Max-SAT; Aspartix-V and Aspartix-D employ a translation to ASP; ConArg is based on Constrain Programming; and LabSATSolver implements a direct approach (for **SE-GR**). All of the solvers have been among the top 5 solvers of the respective tasks in ICCMA'15. Hence, the selection is in line with (i) and (ii).

Hardness categories. The obtained performance results are then taken to classify instances into hardness categories by picking the upmost category such that the following conditions apply:

Table 2. Classification results for task group A.

A: **EE-PR**	total	very easy	easy	medium	hard	too hard	not classified
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

1. **[very easy]** Instances completed by all systems in less than 6 seconds solving time.
2. **[easy]** Instances completed by all systems in less than 60 seconds solving time.
3. **[medium]** Instances completed by all systems in less than 10 minutes solving time.
4. **[hard]** Instances completed by at least one system in 20 minutes (twice the time-out) solving time.
5. **[too hard]** Instances such that none of the systems finished solving in 20 minutes.

The results of the classification are exemplified for task group A in Table 2. It can be seen that almost every combination of domain and difficulty category is inhabited by some 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 running the representative solvers does not deliver meaningful results for an instance, the instance is not classified and therefore not considered for selection.

2.3. Benchmark selection

The final benchmark set 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 "very hard" instances for group C, we increase the number of "hard" instances to 150 there.

Not only we do 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 have 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

Table 3. Number of selected instances for each task group, difficulty class, and domain.

Task group	A						B						C					
Hardness 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

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 domains where I_d is non-empty is higher than the number of instances that remains to be selected, the domains to be chosen from are determined randomly. For example, if we have 1 instance for domain α, 2 for β, 4 for γ, and 11 for δ; and want to select 10 instances, we take 1 from α, 2 from β, 3 from γ, 3 from δ, and 1 randomly either from γ or δ.

The numbers of selected instances for every domain, task group, and difficulty category can be read off in Table 3.

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.

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 drop the "very easy" instances for acceptance tasks and select two arguments to be queried for the "very hard" instances.

For each task group except group D (see Section 2.4), 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.

2.4. Further issues

No stable extensions. Semi-stable and stage extensions coincide with stable extensions if at least one of the latter exists. Therefore, in order 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 contain a considerable amount of benchmarks possessing no stable extensions. To this end, we check the selected instances on existence of stable extensions; the numbers are shown in Table 4. We consider the number of instances without stable extensions to be satisfactory.

Argument selection for ideal semantics. While the selection of arguments for the decision tasks **DC** and **DS** in all task groups except D was done randomly, for ideal se-

Table 4. Share of instances without stable extensions.

hardness category	$\mathbf{ST}(F) \neq \emptyset$	$\mathbf{ST}(F) = \emptyset$	unknown	share
very easy	34	16	0	32%
easy	34	16	0	32%
medium	60	40	0	40%
hard	56	33	11	$> 33\%$
too hard	30	9	11	$> 18\%$
total	214	114	22	$> 34\%$

mantics we are aiming for a more sophisticated selection in order to select the "interesting" arguments for the acceptance task. That selection is based on the insights that every argument contained in the grounded extension is accepted under ideal semantics, and every argument not contained in every preferred extension is rejected under ideal semantics. Hence, we aim for a considerable number of instances for which we select an argument contained in all preferred extensions, but not in the grounded extension. We do so by considering the following strategy: Given an AF $F = (A,R)$, let α and β be random variables taking values between 0 and 1.

- if $\bigcup \mathbf{PR}(F) \setminus \mathbf{GR}(F) \neq \emptyset$ and $\alpha < 0.9$, select an argument randomly from $\bigcup \mathbf{PR}(F) \setminus \mathbf{GR}(F)$;
- otherwise, if $\mathbf{GR}(F) \neq \emptyset$ and $\beta < 0.6$, select an argument randomly from $\mathbf{GR}(F)$;
- otherwise, select an argument randomly from $A \setminus \bigcup \mathbf{PR}(F)$.

We apply this strategy to the selection of query arguments for instances in hardness category easy and medium, and obtained the following distributions for the selected arguments a:

	$a \in \mathbf{GR}(F)$	$a \in \bigcup \mathbf{PR}(F) \setminus \mathbf{GR}(F)$	$a \in A \setminus \bigcup \mathbf{PR}(F)$
easy	14	**15**	21
medium	21	**21**	58

We randomly select the arguments for the hard and too hard instances.

References

Sarah A. Gaggl, Thomas Linsbichler, Marco Maratea, and Stefan Woltran. Introducing the second international competition on computational models of argumentation. In Matthias Thimm, Federico Cerutti, Hannes Strass, and Mauro Vallati, editors, *Proceedings of the 1st International Workshop on Systems and Algorithms for Formal Argumentation (SAFA 2016)*, pages 4–9, 2016. URL `https://www.dbai.tuwien.ac.at/iccma17/Introducing_ICCMA17.pdf`.

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