Containerisation and Dynamic Frameworks in ICCMA'19

Stefano BISTARELLI [a,1], Lars KOTTHOFF [b], Francesco SANTINI [a,1] and Carlo TATICCHI [c]

[a] *Dipartimento di Matematica e Informatica, University of Perugia, Italy*
email: [stefano.bistarelli, francesco.santini]@unipg.it
[b] *Department of Computer Science, University of Wyoming, U.S.*
email: larsko@uwyo.edu
[c] *Gran Sasso Science Insitute, L'Aquila, Italy*
email: carlo.taticchi@gssi.it

Abstract. The *International Competition on Computational Models of Argumentation* (ICCMA) is a successful event dedicated to advancing the state of the art of solvers in Abstract Argumentation. We describe two proposals that will further improve the third and next edition of the competition, i.e. ICCMA 2019. The first novelty concerns the packaging of each solver-application participating in the competition in a virtual "light" *container* (using Docker): this allows for easy deployment and to (re)running all of the submissions on different architectures (Linux, Windows, macOS, and also in the cloud). The second proposal consists of a new track focused on solvers processing dynamic frameworks, i.e., solvers described in terms of changes w.r.t. previous ones: a solver can reuse the solution obtained previously to be faster on the same framework modulo a new argument/attack.

1. Introduction

An *Abstract Argumentation* semantics [12] is a declarative or procedural method for deriving a set of subsets of arguments (each called an *extension*) from an *Abstract Argumentation Framework* (*AF*), which is simply defined by a set of arguments and an attack relationship, i.e., $\langle A, R\rangle$. Such subsets define different levels of "acceptability", in the sense that selected arguments need to survive the conflict defined by R together.

The popularity of the abstract approach defined in [12] has increased more and more in terms of the interest in the scientific community. Abstracting from both the internal structure of arguments in terms of *premises* and *conclusion*, and from the logic defining the inference process, is appealing: it leads to the investigation of general properties of debating. Strong links of Abstract Argumentation to Graph Theory and Logic Programs are well-known as well, and advanced by the seminal original paper itself [12].

Recent years have experienced a wide interest in solving problems related to Abstract Argumentation (e.g., [5]). The evidence, and a further incentive, has been the or-

[1] Authors supported by projects "REMIX" (funded by Banca d'Italia and Fondazione Cassa di Risparmio di Perugia) and "RACRA" (funded by "Ricerca di Base 2018-2020, University of Perugia).

ganisation of the first edition of the *International Competition on Computational Models of Argumentation* (*ICCMA'15*) [17], followed by the 2017 [14] and 2019[2] editions. The latter is scheduled for the next year, and it will be organised by the authors of this paper: the reference solver for checking results will be ConArg[3] [4,6,7], since it is maintained by some of the competition organisers (*Probo* was the reference in ICCMA'15). This kind of competition aims to showcase the current state of the art, as well as nurture research and development of implementations for computational models of argumentation.

The goal of this paper is to present the two main novelties that will be considered in ICCMA'19: in particular, (i) the *containerisation*, using the *Docker* platform, of the solvers submitted to the competition, and (ii) new tracks dedicated to *dynamic* systems. The main motivation behind the first point is to allow each solver to be delivered with its complete run time environment to make setup and deployment easier; moreover, a dockerised application can be launched on different platforms (e.g., Windows, Linux, macOS, and in the cloud), making it possible to recompute the experiments anywhere. The motivation behind introducing dynamic tracks to check the current capabilities and performance of solvers in exploiting previous results with the purpose of solving the same task after a small change in the considered AF (e.g., adding an attack). A debate is a dynamic process by nature, and we hope that the new tracks will highlight the capabilities of current systems in this respect.

2. ICCMA

The ICCMA competition aims at nurturing research and development of implementations for computational models of Argumentation. Previous editions of the competition have seen the participation of 18 and 16 solvers respectively (i.e., 2015[4] and 2017[5]).

For each tested semantics, 4 different problems have been tested: determine some (SE) and all (EE) extensions, credulous (DC) and sceptical (DS) acceptance of a given argument (passed as a parameter of the problem together with an AF). The considered semantics in ICCMA'15 were complete, preferred, stable, and grounded (all defined in [12]), for a total of 16 different tasks; 14 in practice, since SE and EE, and DC and DS, correspond to the same task due to the uniqueness of the grounded extension. In ICCMA'17 the problems were the same, but on a larger set of semantics, considering also the semi-stable [10], stage [18], and ideal [13] ones. Hence, the total number of tasks was 24 (both the grounded and ideal semantics return a single extension). In addition, ICCMA featured a special track called "Dung's Triathlon", which required solvers to enumerate all grounded, preferred and stable extensions at once. In both editions a solver could be submitted to be tested on only a subset of the proposed tracks.

In ICCMA'15, benchmarks were randomly generated based on three different graph models: (i) a very large grounded extension (and having many nodes in general), (ii) many complete/preferred/stable extensions, and (iii) a rich structure of strongly connected components. Each generator was used to create three classes of frameworks; *small*, *medium*, and *large* (according to the number of nodes), each of them storing 24

[2]`http://iccma2019.dmi.unipg.it/`.
[3]`http://www.dmi.unipg.it/conarg/`.
[4]`http://argumentationcompetition.org/2015/index.html`.
[5]`http://argumentationcompetition.org/2017/index.html`.

different instances. For ICCMA'17, a call for benchmarks was put out, and the competition received 6 submissions. Some of them were an example set of frameworks, others the generating tool itself. The generators of ICCMA15 were reused, and one of the submitted generators was able to create three different classes of networks, the total number of different classes was 11. From these classes, 350 AAFs were chosen for each of three representative tasks; EE-preferred, EE-stable, and SE-grounded. This resulted in 50 very easy, 50 easy, 100 medium, 100 hard, and 50 too hard to solve AAFs. For each task, three different solvers from ICCMA'15 were used to test instances hardness. Moreover, the AAFs to test the ideal and semi-stable semantics, and Dung's Triathlon, were chosen in accordance to the characteristics of these problems.[6]

The timeout for returning the result of each task was set to 10 minutes. In ICCMA'15, the solvers were ranked w.r.t. the number of timeouts on these instances, ties were broken by the actual runtime on the instances. An output error (e.g., unparsable output) was considered as an unsolved instance that caused a timeout. The ranking was created using the Borda count across all the tracks. In ICCMA'17, for each instance each solver got a score of 1 for a *correct* answer, -5 for an *incorrect* answer, and 0 otherwise (e.g., in case of output errors and timeouts). Points for each instance were aggregated per semantics, e.g., DS, DC, SE, and EE for the preferred semantics.

3. Docker

Docker[7] is an open-source implementation of operating-system-level virtualisation, also known as *containerisation*. Docker is primarily developed for Linux, where it uses the resource isolation features of the Linux kernel such as cgroups and kernel namespaces, and a union-capable file system, to allow independent "containers" to run within a single Linux instance. The main aim is to avoid the overhead of starting and maintaining virtual machines. Figure 1 shows the general structure compared to virtual machines.Docker allows applications to use the same Linux kernel as the system that they are running on and only requires applications to be shipped with things not already running on the host computer. The container can also be executed on other operating systems: besides different Linux distros such as Debian, Fedora, and Ubuntu, there also exist engines for Mac OS, Windows, Amazon Web Services, and Microsoft Azure. The latter allow to move an application quickly into the cloud without modifications.

Containers in ICCMA. We will require each solver to be submitted to the competition to be packaged in a Docker container. To do so, a submitter needs three files: (i) a Dockerfile, (ii) a *solver_interface.sh* file, and (iii) the solver itself. The Dockerfile defines the environment in the container. Access to resources like networking interfaces and disk drives is virtualised inside this environment. An example is shown in Figure 2: it creates an Alpine Linux container (a minimal distro) and defines the environment where the solver will run. Environment variables describing the input file (i.e., the AF) and the task to be solved by the solver can be added. The command specified by CMD is run when the container is started. The file *solver_interface.sh* is a shell-script file with a common interface used to run each solver; this file will be provided by the organisers of ICCMA'19, and only minor changes are needed to be personalised to a specific solver.

[6] `http://argumentationcompetition.org/2017/benchmark_selection_iccma2017.pdf`.
[7] Docker Inc.: `https://www.docker.com`.

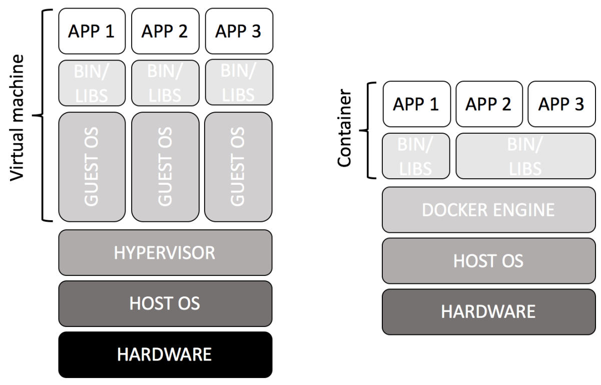

Figure 1. Difference in abstraction layers between virtual machines and Docker containers.

A Docker image is built with the command `docker build -t imageName` and run with `docker run imageName`. Docker images can be *pushed* to and *pulled* from online repositories (such as Docker Hub), and all benchmark AFs can be mounted in a Docker volume. This allows the submissions to be prepared independently of the benchmark instances and then simply mounted into the submission container to run the competition.

4. Dynamic Frameworks

In previous ICCMA editions, all the frameworks in each database were considered static in the sense that all the AFs were sequentially passed as input to solvers, representing different and independent information: all tasks are computed from scratch without taking any potentially useful knowledge from previous runs into account.

However, in many practical applications, an AF represents only a temporary situation: arguments and attacks can be added/retracted to take into account new knowledge that becomes available. For instance, in disputes among users of online social networks [16], arguments/attacks are repeatedly added/retracted by users to express their point of view in response to the last move made by the adversaries in the current digital polylogue (often disclosing as few arguments/attacks as possible).

The dynamics of frameworks has attracted recent and wide interest in the Argumentation community. We describe some related work, which also points to the research groups interested in the organisation of such a track. In [9], the authors investigate the principles according to which a grounded extension of a Dung's AF does not change

```
FROM alpine
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY . /usr/src/app
ENV INPUTFILE ""
...
ENV PROBLEM ""
CMD [ "/usr/src/app/ConArg_interface.sh" ]
```

Figure 2. An example of Dockerfile calling a shell-script file personalised to ConArg; more environment variables can be set, while running the created container using the ENV parameter.

when the set of arguments/attacks are changed. The work of [11] studies how the extensions can evolve when a new argument is considered. The authors focus on adding one argument interacting with one starting argument (i.e., an argument which is not attacked by any other argument). In further work [19], the authors study the evolution of the set of extensions after performing a change operation (addition/removal of arguments/interaction). In [2], the authors propose a division-based method to divide the updated framework into two parts: "affected" and "unaffected". Only the status of affected arguments is recomputed after updates. A matrix-reduction approach that resembles the previous division method is presented in [19]. A recent work that tests complete, preferred, stable, and grounded semantics on an AF and a set of updates is [1]. This approach finds a reduced (updated) AF sufficient to compute an extension of the whole AF, and uses state-of-the-art algorithms to recompute an extension of the reduced AF only.

Modifications of AFs are also studied in the literature as a base to compute *robustness* measures of frameworks [8]. In particular, by adding/removing an argument/attack, the set of extensions satisfying a given semantics may or may not change. For instance, one could be interested in computing the number of modifications needed to bring a change in this set, or measure the number of modifications needed to have a different set of extensions satisfying a desired semantics. In the latter case, the user is interested in having an estimate on how distant two different points of views are; this kind od approach has also been proposed in [3].

Dynamic Track in ICCMA. We will organise the following dynamic track in ICCMA'19. Its goal is (i) to determine what solvers work incrementally, and (ii) to test how much better they tackle the same problems compared to static solvers. To achieve this, we will use some of the AFs adopted in the benchmark to test static solvers, which will be collected by reusing some of ICCMA'15 and ICCMA'17 AFs, and by putting out a call for benchmark or generators as in ICCMA'17. Some of these AFs will be chosen from (some of) these sets as initial frameworks, which will be passed to solvers together with a list of changes in a separate file. Changes may consist of a sequence of random additions/deletions of attacks, which will be provided through a simple text format, e.g., *+att(a,b). -att(d,e).* ..., representing the introduction and deletion of an attack, respectively (two AFs to process in the end). We will then expect as many outputs as the number of changes plus one (the original framework). In order to compare performance, static solvers will be tested on these networks as well, by providing full AFs instead of lists of changes. The track will consider the four semantics originally advanced in [12], i.e., complete, preferred, stable, and grounded, and as tasks SE, EE, DC, and DS.

5. Conclusion

We have described two proposals that we will introduce in ICCMA'19. The first allows for easy moving and testing of submissions on different platforms and in different contexts – everything a solver needs is packaged into a Docker container. We strongly believe in *recomputability* [15] – recomputing computational experiments (i.e. rerunning them with exactly the same conditions) should be very easy. Using Docker will make ICCMA easy to recompute, allowing submitters and independent parties to easily reproduce our results and build on them to advance the state of the art. As the second improvement, we will organise a track dedicated to dynamic solvers, where previous results can be used to rapidly reach a solution on a slightly modified AF, instead of solving the whole problem from scratch. This will provide a different lens to assess the current state of the art and make the competition more relevant to current research in the community.

References

[1] G. Alfano, S. Greco, and F. Parisi. Efficient computation of extensions for dynamic abstract argumentation frameworks: An incremental approach. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, pages 49–55. ijcai.org, 2017.

[2] P. Baroni, M. Giacomin, and B. Liao. On topology-related properties of abstract argumentation semantics. A correction and extension to dynamics of argumentation systems: A division-based method. *Artif. Intell.*, 212:104–115, 2014.

[3] R. Baumann and G. Brewka. Expanding argumentation frameworks: Enforcing and monotonicity results. In *Computational Models of Argument (COMMA)*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 75–86. IOS Press, 2010.

[4] S. Bistarelli, F. Rossi, and F. Santini. A comparative test on the enumeration of extensions in abstract argumentation. *Fundam. Inform.*, 140(3-4):263–278, 2015.

[5] S. Bistarelli, F. Rossi, and F. Santini. Not only size, but also shape counts: abstract argumentation solvers are benchmark-sensitive. *J. Log. Comput.*, 28(1):85–117, 2018.

[6] S. Bistarelli, F. Rossi, and F. Santini. A novel weighted defence and its relaxation in abstract argumentation. *Int. J. Approx. Reasoning*, 92:66–86, 2018.

[7] S. Bistarelli and F. Santini. Modeling and solving afs with a constraint-based tool: Conarg. In *Theorie and Applications of Formal Argumentation - First International Workshop, TAFA*, volume 7132 of *LNCS*, pages 99–116. Springer, 2012.

[8] S. Bistarelli, F. Santini, and C. Taticchi. On looking for invariant operators in argumentation semantics. In *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS*, pages 537–540. AAAI Press, 2018.

[9] G. Boella, S. Kaci, and L. W. N. van der Torre. Dynamics in argumentation with single extensions: Abstraction principles and the grounded extension. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty ECSQARU*, volume 5590 of *LNCS*, pages 107–118. Springer, 2009.

[10] M. W. A. Caminada, W. Alexandre Carnielli, and P. E. Dunne. Semi-stable semantics. *J. Log. Comput.*, 22(5):1207–1254, 2012.

[11] C. Cayrol, F. de Saint-Cyr, and M.-C. Lagasquie-Schiex. Change in abstract argumentation frameworks: Adding an argument. *J. Artif. Intell. Res.*, 38:49–84, 2010.

[12] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.

[13] P. Minh Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. *Artif. Intell.*, 171(10-15):642–674, 2007.

[14] S. A. Gaggl, T. Linsbichler, M. Maratea, and S. Woltran. Introducing the second international competition on computational models of argumentation. In *Proceedings of the First International Workshop on Systems and Algorithms for Formal Argumentation (SAFA)*, volume 1672 of *CEUR Workshop Proceedings*, pages 4–9. CEUR-WS.org, 2016.

[15] I. P. Gent and L. Kotthoff. Recomputation.org: Experiences of its first year and lessons learned. In *Proceedings of the 7th IEEE/ACM International Conference on Utility and Cloud Computing, UCC*, pages 968–973. IEEE Computer Society, 2014.

[16] N. Kökciyan, N. Yaglikci, and P. Yolum. Argumentation for resolving privacy disputes in online social networks: (extended abstract). In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 1361–1362. ACM, 2016.

[17] M. Thimm and S. Villata. The first international competition on computational models of argumentation: Results and analysis. *Artif. Intell.*, 252:267–294, 2017.

[18] B. Verheij. Two approaches to dialectical argumentation: admissible sets and argumentation stages. *Proceedings of the Eighth Dutch Conference on Artificial Intelligence (NAIC)*, 96:357–368, 1996.

[19] Y. Xu and C. Cayrol. The matrix approach for abstract argumentation frameworks. In *Theory and Applications of Formal Argumentation TAFA*, volume 9524 of *LNCS*, pages 243–259. Springer, 2015.