# Benchmarking Hard Problems in Random Abstract AFs: The Stable Semantics

STEFANO BISTARELLI [a,1], FABIO ROSSI [a] and FRANCESCO SANTINI [b,2]
[a] *Dipartimento di Matematica e Informatica, Università di Perugia, Italy*
*[bista,rossi]@dmi.unipg.it*
[b] *Istituto di Informatica e Telematica (IIT-CNR), Pisa, Italy, francesco.santini@iit.cnr.it*

**Abstract.** In this paper we test four different implementations of reasoning tools dedicated to Abstract Argumentation Frameworks. These systems are ASPARTIX, dynPARTIX, Dung-O-Matic, and ConArg2. The tests are executed over three different models of randomly-generated graphs, i.e., the Erdős-Rényi model, the Kleinberg small-world model, and the scale-free Barabasi-Albert model. We compare these four tools with the purpose to test the search of all the possible stable extensions. Then we benchmark dynPARTIX and ConArg2 on the credulous and skeptical acceptance of arguments. Finally, we also evaluate ConArg2 to check the existence of a stable extension.

## 1. Introduction

An *Abstract Argumentation Framework* (*AAF*), or System, as introduced in a seminal paper by Dung [8], is simply a pair $\langle \mathcal{A}, R \rangle$ consisting of a set $A$ whose elements are called arguments and of a binary relation $R$ on $A$, called "attack" relation. An abstract argument is not assumed to have any specific structure but, roughly speaking, an argument is anything that may attack or be attacked by another argument. The sets of arguments (or *extensions*) to be considered are then defined under different semantics, which are related to varying degrees of skepticism or credulousness.

In this paper we focus on the stable semantics [8], which basically consists in conflict-free extensions that attack all the outside arguments. The simple intuition behind this semantics has significant counterparts in several contexts, since it represents a sort of equilibrium: such extensions are in correspondence with solutions of cooperative n-person games, solutions of the stable marriage problem, extensions of Reiter's default logic, and stable models of logic programs [16, Ch. 2]. A significant drawback is that this semantics does not always exists in a given AAF, and this is why the problem of finding its existence is crucial though, unfortunately, NP-complete at the same time (see Sec. 2). In addition, an argument is credulously accepted if it belongs at least to one extension satisfying a given semantics, while it is skeptically accepted if it belongs to all

[1]Partially supported by MIUR-PRIN "Metodi logici per il trattamento dell'informazione".
[2]Partially supported by MIUR-PRIN "Security Horizons".

such extensions over an AAF. Considering the stable semantics, checking the credulous or skeptical justification is an NP-complete and coNP-complete problem respectively.

The main goal of this paper is to better understand how efficiently such hard problems related to the stable semantics can be solved by state of the art reasoners. For instance, we randomly select a $10\%$ of the arguments with the purpose to test their credulous/skeptical justification state. We measure their performance over random AAFs with different topology and size. Specifically, we test four tools whose main objective is the pure computation of extensions, i.e., ASPARTIX, dynPARTIX, Dung-O-Matic, and ConArg2 (which is developed by the authors of this paper, see Sec 3.1). We consider three different randomly-generated graph models, thus assembling a variegate benchmark for this kind of testing. These networks are respectively generated according to Erdős-Rényi, Kleinberg, and Barabasi-Albert principles (see Sec. 3.2). We have not considered interaction graphs from the "real-world" due to the current lack of benchmarks extracted from real debates. Mining and testing real-world AAFs, as well as improving the performance of ConArg2, will be part of future work (see Sec. 5). Note that we use three different random-models in our testbed because a clear definition of the topological properties behind real AAFs has not been yet identified in the literature (see Sec. 5).

The remainder is organized as follows: in Sec. 2 we briefly introduce AAFs, while in Sec. 3 we define our benchmark environment, by describing adopted networks and tools. Afterwards, in Sec. 4 we show the results obtained during our tests, and we discuss the charts trying to give some general considerations and guidelines. At the end, Sec. 5 presents further ideas about future work.

## 2. Preliminaries

In this section we briefly summarise the background information related to classical AAFs [8]. We focus on the basic definitions of an AAF, and on the extension-based semantics that will be tested in our comparison (see Sec. 4).

**Definition 1 (Abstract AFs)** *An Abstract Argumentation Framework (AAF) is a pair $F = \langle A, R \rangle$ of a set $A$ of arguments and a binary relation $R \subseteq A \times A$, called the attack relation. $\forall a, b \in A$, $aR\,b$ (or, $a \rightarrowtail b$) means that $a$ attacks $b$. An AAF may be represented by a directed graph (an interaction graph) whose nodes are arguments and edges represent the attack relation. A set of arguments $S \subseteq A$ attacks an argument $a$, i.e., $S \rightarrowtail a$, if $a$ is attacked by an argument of $S$, i.e., $\exists b \in S . b \rightarrowtail a$.*

The following notion of defence [8] is fundamental to AAFs.

**Definition 2 (Defence)** *Given an AAF, $F = \langle A, R \rangle$, an argument $a \in A$ is defended (in $F$) by a set $S \subseteq A$ if for each $b \in A$, such that $b \rightarrowtail a$, also $S \rightarrowtail b$ holds. Moreover, for $S \subseteq A$, we denote by $S_R^+$ the set $S \cup \{b \mid S \rightarrowtail b\}$.*

In Def. 3 we give the formal definition of stable extension ($stb$), while Def. 4 defines the credulous and skeptical state of justification for an argument.

**Definition 3 (Stable semantics)** *Let $F = \langle A, R \rangle$ be an AAF. A set $S \subseteq A$ is conflict-free (in F), denoted $S \in cf(F)$, iff there are no $a, b \in S$, such that $(a, b), (b, a) \in R$. For $S \in cf(F)$, it holds that $S \in stb(F)$, if foreach $a \in A \backslash S$, $S \rightarrowtail a$, i.e., $S_R^+ = A$.*

**Definition 4 (Justification state)** *Given a semantics $\mathcal{S}$ and a framework $\mathcal{A}$, an argument $a$ is* i) *skeptically justified iff* $\forall E \in \mathcal{E}_{\mathcal{S}}(\mathcal{A})\ a \in E$ *, and* ii) *credulously justified if* $\exists E \in \mathcal{E}_{\mathcal{S}}(\mathcal{A})\ a \in E$.

On the stable semantics, the complexity of deciding the credulous and the skeptical acceptance of an argument is NP-complete and co-NP-complete respectively [16, Ch. 5]. A third hard problem we solve in the paper is the existence of a stable extension in a given AAF: "is $stb(F) \neq \emptyset$?" is an NP-complete question [16, Ch. 5].

## 3. Tools and Graphs

We organise the content of this section in two subsections: Section 3.1 presents a description of the four reasoning tools we assess in our comparison, while Sec. 3.2 describes the random graphs we generated as benchmark.

### *3.1. Tools*

The **ASPARTIX**[3] system [11] is a tool for computing acceptable extensions for a broad range of formalisations of Dung's AAFs and generalisations, such as value-based [3] or preference-based AAFs [1]. *ASPARTIX* relies on a fixed disjunctive *Datalog* program which takes an instance of an argumentation framework as input, and uses an Answer-Set solver for computing the type of extension specified by the user. ASPARTIX is able to solve admissible, stable, complete, grounded, preferred, semi-stable, ideal, stage, cf2, resolution-based grounded and stage2 extensions. It has been improved with the *metasp*[4] optimization front-end for the ASP-package *gringo/claspD*[5], which provides direct commands to filter answer sets satisfying certain subset-minimality (or -maximality) constraints. We have included ASPARTIX in the comparison because it represents the state of the art of ASP-based solvers [18], and the first comprehensive reasoning-tool in general.

**dynPARTIX**[6] is a system based on decomposition and dynamic programming; it is motivated by the theoretical results in [10]. The underneath algorithms make use of the graph-parameter tree-width, which measures the "tree-likeness" of a graph. More specifically, tree-width is defined via the so-called tree-decompositions. A tree decomposition is a mapping from an AAF to a tree where the nodes in the tree contain *bags* of arguments from the AAF. Each argument appears in at least one bag, adjacent arguments are together in at least one bag, and bags containing the same argument are connected. The benchmarks in [9] show that the run-time performance of dynPARTIX heavily depends on the tree width of the considered graph: for example, with instances of small tree width, dynPARTIX outperforms ASPARTIX, while high tree width the ASPARTIX system still performs better. In out tests we use the new 64bit version (2.0) of dynPARTIX, which has recently become available.

[3] `http://www.dbai.tuwien.ac.at/proj/argumentation/systempage/`
[4] `http://www.cs.uni-potsdam.de/wv/metasp/`
[5] `http://potassco.sourceforge.net`
[6] `http://www.dbai.tuwien.ac.at/proj/argumentation/dynpartix/`

**Dung-O-Matic**[7] is an Abstract Argument computation engine implemented by the *javaDungAF* class. Dung-O-Matic supports the solution of several semantics over AAFs, namely the admissible, complete, eager, grounded, ideal, preferred, semi-stable, and stable semantics. javaDungAF is currently contained in a couple of java classes. Source code and documentation are available to download. For each of the proposed extensions, the tool implements a different algorithm presented in the literature; for instance, the grounded semantics is computed with the original algorithm designed by Dung in [8]. We have included Dung-O-Matic in the comparison just because all the implemented semantics come from ad-hoc algorithms recently designed and proposed in the literature.

**ConArg2**. ConArg[8] [5,6] is a reasoner based on the *Java Constraint Programming* solver[9] (JaCoP), a Java library that provides a *Finite Domain Constraint Programming* paradigm [17]. The tool comes with a graphical interface, which visually shows all the obtained extensions for each problem. ConArg is able to solve also the weighted and coalition-based problems presented in [4,7]. Moreover, it can import/export AAFs with the same text format of ASPARTIX. Recently, we have extended the tool to its second version, i.e., ConArg2 (freely downloadable from the same webpage of ConArg), in order to improve its performance: we implemented all the models in Gecode[10], which is an open, free, and efficient C++ environment where to develop constraint-based applications. Hence, we model each semantics as a *Constraint Satisfaction Problem* (*CSP*) [17]. We have also dropped the graphical interface, having a textual output only, with the purpose to have a tool exclusively oriented to performance. So far, on classical AAFs, ConArg2 is able to find all conflict-free, admissible, complete, stable, grounded, preferred, semi-stable, and ideal extensions; moreover, it solves the credulous and skeptical acceptance of arguments given the stable semantics, and its existence problem.

### *3.2. Graphs*

To generate random graphs we adopted two different libraries. The first one is the *Java Universal Network/Graph Framework* (*JUNG*[11]), which is a Java software library for the modeling, generation, analysis and visualization of graphs. With JUNG we generate *Barabasi-Albert* [2] and *Kleinberg* [15] graphs. The second library we use is *NetworkX*[12], and it consists of a Python software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. With NetworkX we generate *Erdős-Rényi* [12] graphs.

In order to test tools over sensibly wide AAFs, our attention has mainly turned to random networks with small-world, as Kleinberg or Barabasi-Albert (which is also a scale-free model [2]). Big hub-nodes in scale-free networks are responsible for the so-called small-world phenomenon, where most nodes can be reached from every other by a small number of hops. An example is given by Barabasi-Albert graphs. Kleinberg graphs show the typical grid-like structure of Kleinberg graphs, with few long-distance connections. The Erdős-Rényi) model does not generate local clustering and does not

[7] `http://www.arg.dundee.ac.uk/?page_id=279`
[8] `http://www.dmi.unipg.it/bista/tt/conarg/`
[9] `http://www.jacop.eu`
[10] `http://www.gecode.org`
[11] `http://jung.sourceforge.net`
[12] `http://networkx.github.io`

account for the formation of hubs: this model is the least indicated to represent social patterns among the three models. Nevertheless, we have included it in this study for the sake of completeness.

We are not aware of any study matching AAFs to a precise graph model (or several models, depending on the kind of the discussion, e.g., persuasion or negotiation-oriented). The justification behind using this kind of graphs following the small-world phenomenon is that several works in the Argumentation literature investigate AAFs extracted from social networks [14], as Facebook [19], or Twitter [13].

In the following we detail the three adopted random-models. In the **Erdős-Rényi** model [12] a graph is constructed by randomly connecting $n$ nodes. Each edge is included in the graph with probability $p$ independent from every other edge. Clearly, as $p$ increases from 0 to 1, the model becomes more and more likely to include graphs with more edges. To create our benchmark we adopt $p = c \cdot log(n)/n$, with $c$ set to 2.5, which ensures the connectedness of such graphs. We require all the graphs in the benchmark to be connected, since we suppose a discussion to dynamically evolve by adding each time an argument in order to contest a different former argument.

**Kleinberg** [15] adds a number of directed long-range random links to an $n \times n$ lattice network (vertices as nodes of a grid, undirected edges between any two adjacent nodes). Links have a non-uniform distribution that favours edges to close nodes over more distant ones. In the implementation provided by JUNG, each node $u$ has four local connections, one to each of its neighbors, and in addition 1 or more long range connection to some node $v$, where $v$ is chosen randomly according to probability proportional to $d^{\theta}$ where $d$ is the lattice distance between $u$ and $v$ and $\theta$ is the clustering exponent. During the generation we set $\theta = 0.9$, in order to have a high clustering coefficient.

In the generation of **Barabasi-Albert** graphs [2] through the JUNG libraries, at each time step a new vertex is created and connected to existing vertices according to the principle of "preferential attachment" [2], whereby vertices with higher degree have a higher probability of being selected for attachment. At a given step, the probability $p$ of creating an edge between an existing vertex $v$ and the newly added vertex is $p = (degree(v) + 1)/(|E| + |V|)$. $|E|$ and $|V|$ are, respectively, the number of edges and vertices currently in the network. Thus, $p$ is not an input parameter.

## 4. Tests and Discussion

The performance results have been collected on an Intel(R) Core(TM) i7 CPU 970 @3.20GHz (6 core, 2 threads per core), and 16GB of RAM. For all the tools, the output has been redirected to */dev/null*, and the standard error to file. To test ASPARTIX we take advantage of *gringo 3.0.5* and *claspD 1.1.4*: no metasp optimisation is available at the moment for the stable semantics. To run ConArg2 we adopt Gecode 4.0, and for Dung-O-Matic we use Java "1.6.0_18", while dynPARTIX is tested in its 2.0 version (64bit). We use a timeout of 300 seconds to stop each of the four tools.

In Fig. 1 we show the behaviour of all the four reasoners (see Sec. 1) on finding all the stable extensions for a given AAF. For each of the reported number of nodes (on the $x$ axis), the left $y$ axis reports the average CPU time (on 100 different instances of AAFs) for all the stable extensions found within the timeout of 300 seconds. On the right $y$ axis we also report the number instances that are not solved within such timeout ("#unsuccessful"), for each solver.

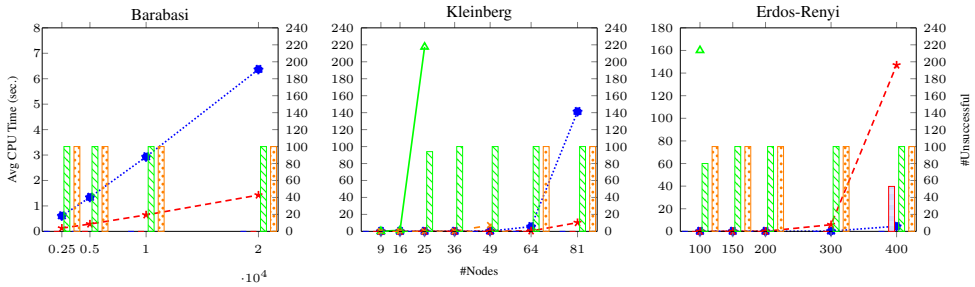



**Figure 1.** ASPARTIX ( ), ConArg2 ( ), Dung-O-Matic ( ), dynPARTIX ( ), #Unsuccessful with ASPARTIX ( ), ConArg2 ( ), Dung-O-Matic ( ), dynPARTIX ( ).

In Fig. 2a and Fig 2b we respectively describe the tests to measure the credulous and skeptical acceptance of arguments. For each of the 100 AAFs, we test a random 10% of the arguments. In this case we only run dynPARTIX and ConArg2, since they are the only two tools that can solve these problems. Chart labels are the same as in Fig. 1, and a timeout of 300 seconds has been applied even in this case. From these two figures we can appreciate that ConArg2 perfectly scales with the number of nodes. However, the poor performance of dynPARTIX could be due to the fact that semi-normalized tree decomposition is not currently implemented in dynPARTIX (only plain normalization), and it is known that it provides better performance with other semantics (e.g., admissible) [10,9].

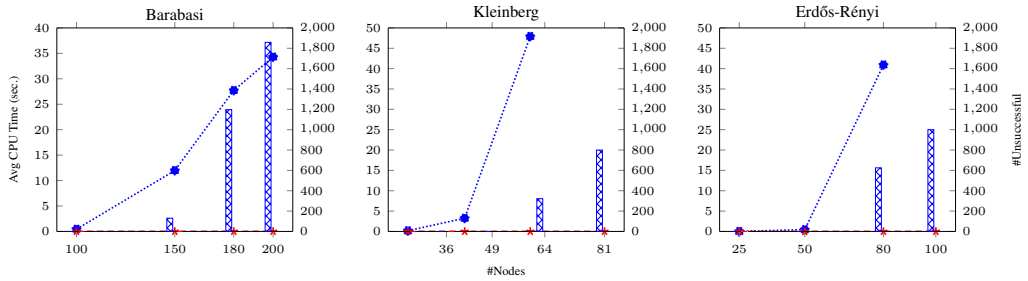
The charts of the third hard problem, related to the existence of a stable extension, are reported in Fig. 3. The curves are almost identical to the ones in Fig. 1: hence, finding one or all the stable extensions require similar efforts.

## 5. Conclusion

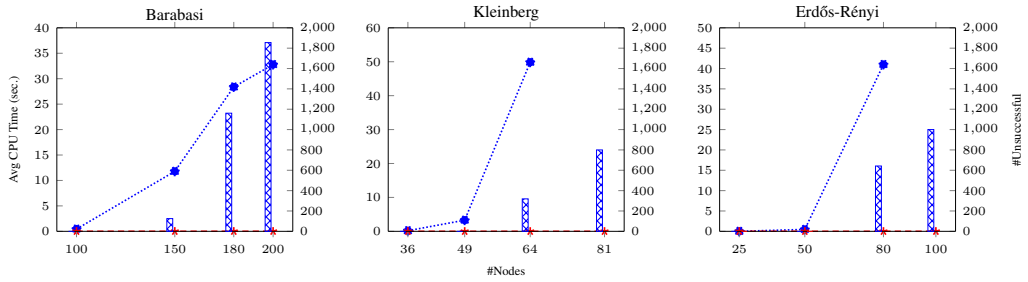
In the paper we have compared four reasoners (ASPARTIX, dynPARTIX, Dung-O-Matic, and ConArg2). The main goal has been to study how well state-of-the-art reasoners behave on hard problems related to the stable semantics (see Sec. 2 for a description of such problems). The testbed has been created by using three different random graph-models.

We guess the results in Sec. 4 can guide the future research along several interesting lines. First of all, we plan to extend ConArg2 to solve weighted problems [4], and all the other hard problems implemented in CEGARTIX and dynPARTIX (e.g., the credulous and skeptical acceptance of an argument in preferred semantics) in order to have a full comparison considering these tools. To do so, we also plan to design and implement specific search-heuristics (using Gecode and its branch-and-bound search, for instance) in order to improve the performance with higher-order extensions (e.g., preferred), so to better manage the maximality of set inclusion. Note that these heuristics can be inspired by, or tuned on, a specific graph model.

Finally, we would like to test these tools over AAFs extracted from real AAFs, for instance using [13]. Related, but on a different perspective, we would like to study the topology of real AAFs, in order to match them to specific (social) graph-models and improve the performance with real data. A regular tree-like structure could not be the right model when dealing with multiparty debates, where a few arguments, at the core of

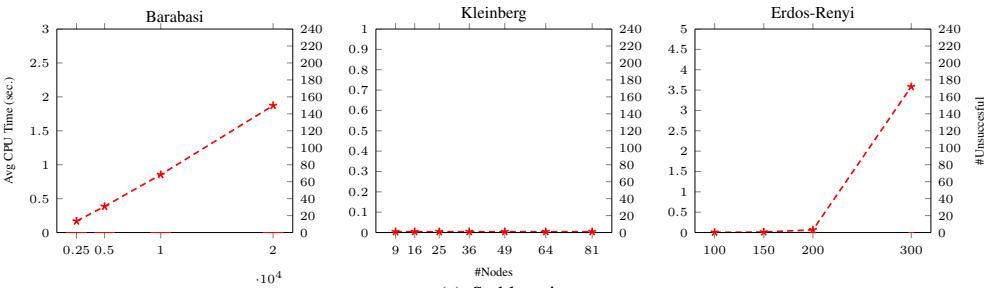



(a) Stable - Credulous




(b) Stable - Skeptical

**Figure 2.** dynPARTIX ( ), ConArg2 ( ), #Unsuccesful with dynPARTIX ( ), ConArg2 ( ).




(a) Stable existence

**Figure 3.** ConArg2 ( ), #Unsuccesful with ConArg2 ( ).

the discussion, show a higher number of incoming attacks. Moreover, we remind that an iterative Barabasi-Albert model where we connect a new node to only one old node, is a tree: hence, this could result in the random graph-model closest to our needs.

## Acknowledgments

We would really like to thank Sarah Alice Gaggl (TU Dresden), Günther Charwat and Wolfgang Dvorak (TU Wien) for guiding us through the use of ASPARTIX and dynPARTIX.

## References


[1] L. Amgoud and C. Cayrol. Inferring from inconsistency in preference-based argumentation frameworks. *J. Autom. Reasoning*, 29(2):125–169, 2002.

[2] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[3] T. J. M. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *J. Log. Comput.*, 13(3):429–448, 2003.

[4] S. Bistarelli and F. Santini. A common computational framework for semiring-based argumentation systems. In *ECAI 2010 - 19th European Conference on Artificial Intelligence*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 131–136. IOS Press, 2010.

[5] S. Bistarelli and F. Santini. Conarg: A constraint-based computational framework for argumentation systems. In *Proceedings of the 23rd International Conference on Tools with Artificial Intelligence*, ICTAI '11, pages 605–612. IEEE Computer Society, 2011.

[6] S. Bistarelli and F. Santini. Modeling and solving AFs with a constraint-based tool: Conarg. In *Theory and Applications of Formal Argumentation*, volume 7132 of *LNCS*, pages 99–116. Springer Berlin Heidelberg, 2012.

[7] S. Bistarelli and F. Santini. Coalitions of arguments: An approach with constraint programming. *Fundam. Inform.*, 124(4):383–401, 2013.

[8] 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):321–357, 1995.

[9] W. Dvořák, M. Morak, C. Nopp, and S. Woltran. dynpartix- A dynamic programming reasoner for abstract argumentation. In *Applications of Declarative Programming and Knowledge Management*, pages 259–268. Springer, 2013.

[10] W. Dvorák, R. Pichler, and S. Woltran. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artif. Intell.*, 186:1–37, 2012.

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

[12] P. Erdős and A. Rényi. On the evolution of random graphs. *Bull. Inst. Internat. Statist*, 38(4):343–347, 1961.

[13] S. Gabbriellini and P. Torroni. Large scale agreements via microdebates. In *AT*, volume 918 of *CEUR Workshop Proceedings*, pages 366–377. CEUR-WS.org, 2012.

[14] S. Gabbriellini and P. Torroni. Arguments in social networks. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 1119–1120, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[15] C. Martel and V. Nguyen. Analyzing Kleinberg's (and other) small-world models. In *Proceedings of the ACM symposium on Principles of distributed computing*, PODC '04, pages 179–188, New York, NY, USA, 2004. ACM.

[16] I. Rahwan and G. R. Simari. *Argumentation in Artificial Intelligence*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[17] F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.

[18] F. Toni and M. Sergot. Argumentation and answer set programming. In *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, volume 6565 of *LNCS*, pages 164–180. Springer Berlin Heidelberg, 2011.

[19] F. Toni and P. Torroni. Bottom-up argumentation. In *Theory and Applications of Formal Argumentation*, volume 7132 of *LNCS*, pages 249–262. Springer Berlin Heidelberg, 2012.