$\mathcal{ASSA}$: Computing Stable Extensions with Matrices

Evgenios Hadjisoteriou and Michael Georgiou

Department of Computer Science, University of Cyprus
75 Kallipoleos Str., 1678 Nicosia, Cyprus
csp7he2@cs.ucy.ac.cy
http://www.cs.ucy.ac.cy/
Department of Electrical Engineering, Computer Engineering and Informatics, Cyprus University of Technology, 30 Archbishop Kyprianou Str., 3036 Lemesos, Cyprus
mica.georgiou@cut.ac.cy,
https://www.cut.ac.cy/eecei/

Abstract. Abstract argumentation frameworks with countably many arguments can be presented in a matrix form. We have created reasoning tasks based on matrix operations, that can answer whether a given set of arguments is part of an argumentation extension. Our solver, $\mathcal{ASSA}$, is written in Java and implements the stable semantics of an argumentation framework, giving us the opportunity to participate in ICCMA'15 competition.

Keywords: Argumentation, semantics, stable extension, matrix

1 The Project

Argumentation theory tries to mimic the process of reasoning. It is often used by agents to reason under dynamic environments, e.g. which alternative to choose. Agents perform tasks under incomplete information, thus decisions must be precise and easily computable. Agents need a tool that is able to produce extensions under specific semantics to help them decide what their next move should be.

For what follows, we assume that the reader is familiar to basic matrix tools and operations [5] as well as the fundamentals on argumentation frameworks [2, 1, 3]. We have created a theory on how to use mathematical matrix operations, to **navigate** through argumentation frameworks. We then used this method to compute extensions in an abstract argumentation framework.

The computational complexity for multiplying two matrices with n digit numbers using the "default" algorithm is $\mathcal{O}(n^3)$ [6]. Of course there are methods that can optimize this result [6, 4]. Because of this complexity our algorithm can handle argumentation frameworks with not many arguments. First we present any abstract argumentation framework into its adjacency matrix.

Definition 1. *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework. Define its adjacency matrix $A = (a_{i,j})$ as follows:* $a_{i,j} = \begin{cases} 1 \text{ if } (i,j) \in \mathcal{R} \\ 0 \text{ if } (i,j) \notin \mathcal{R} \end{cases}$

It is important to know who attacks who. The "tail" of an outgoing attack is represented by the row of the adjacency matrix and each column represents the "tip" of the attack. Therefore, element $a_{3,4}$ can represent the attacks from arguments a_3 to argument a_4 while element $a_{4,3}$ can represent the attacks from a_4 to a_3.

We then represent any given set of arguments as a column vector. Having these two matrices we can perform matrix operations to navigate through the argumentation framework.

Definition 2. *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework with A its adjacency matrix and $S \subseteq \mathcal{A}$. Set S is represented by a column vector $\mathcal{S}_{n\times 1} = (s_{i,1})$, where $s_{i,1} = \begin{cases} 1 \text{ if } a_i \in S \\ 0 \text{ if } a_i \notin S \end{cases}$*

Proposition 1. *Let A be the adjacency matrix of an argumentation framework $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ and $S \subseteq \mathcal{A}$ be a set of arguments with $\mathcal{S}$ its column vector (resp. $\mathcal{S}^T$ row vector) representation. The product $A\mathcal{S}$ (resp. $\mathcal{S}^T A$) is a column (resp. row) vector where the entry $(A\mathcal{S})_{i,1}$ (resp. $(\mathcal{S}^T A)_{1,i}$) shows how many times argument $a_i \in \mathcal{A}$ attacks (resp. is attacked by) S.*

With the help of matrix operations we can answer question such as which arguments in $\mathcal{A}$ attack (resp. are attacked by) a specific set of arguments. In this way it is like using a series of matrix tools to navigate through the argumentation framework. Many times, crucial information may get lost throughout the process and therefore a gentle manipulation is needed. A method to keep track of the information that might be used at a later point is useful.

Proposition 2 (conflict free test). *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework and A its adjacency matrix. Let $S \subseteq \mathcal{A}$ be a given set of arguments with $\mathcal{S}$ its column vector representation. Let $\Gamma = \mathcal{S}^T A$. $\mathcal{S}$ passes the conflict free test if and only if whenever $\gamma_i \neq 0 \in \Gamma$ then $s_i = 0 \in \mathcal{S}$.*

By constructing a matrix multiplication we can answer if a given set of arguments S is conflict free. When a row matrix passes (resp. fails) the test we conclude that S is (resp. is not) conflict free.

Proposition 3 (stable extensions test). *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework with adjacency matrix A. Let $S \subseteq \mathcal{A}$ be a given set of arguments and $\mathcal{S}$ the column vector of S and $\Gamma = \mathcal{S}^T A$. The set S passes the stable extensions test if and only if:*

1. *$\mathcal{S}$ passes the conflict free test, and*
2. *$\forall i$ such that $s_i = 0, \gamma_i \neq 0$.*

Note that for the stable extension test we do not use the admissibility test. Intuitively, attacking anything that is "outside" of you means that you attack all your attackers. This is true since passing the conflict free test shows that there do not exist attacks coming "inside" of you thus any existing attacks should be from "outside" and you attack them back anyway.

Informally, we try to introduce a mathematical approach based on matrix operations to research, and answer questions of the form: "Is set S an extension accordingly to a semantic σ"? We accomplish this by converting the argumentation framework as well as the set S into a matrix form $\mathcal{A}$ and $\mathcal{S}$ respectively. We then perform matrix tools and operations to extract knowledge and answer questions for set S, i.e. is S conflict free, admissible, ground or complete? At a later point we want to expand our theory on other semantics as well.

2 ICCMA'15 Competition

In order to participate in ICCMA'15 competition we slightly changed our algorithm to fit the rules of this contest. Specifically, we decided not to handle each set of arguments as an individual and try to answer questions based on this set, but to find all possible cases of such sets and combine them into a massive matrix $S^{'}$. Each $S^{'}$ column is one of the instances of S sets.

After building our solver, $A\mathcal{SS}A$, and we were able to handle stable extensions, we discovered that our approach is time consuming and the computational hardness exponentially grows as the number of arguments become bigger and bigger. For example, in order to consider all possible combinations for an argumentation framework with n-many arguments, we have to compute a matrix $S^{'}$, with 2^n columns. It was not much of a surprise when we discovered that our solver runs out of memory after compiling it with an argumentation framework with more than twenty arguments. We know that the result we provide suffers from the computations complexity that follows matrix operations. In a later version of $A\mathcal{SS}A$, we plan to fix this by computing matric $S^{'}$ in a smarter way. We think that by checking some critical arguments at an early stage may reduce the size, and therefore the computational complexity, as well as the memory our solver needs. Nonetheless, one can overcome this problem by running $A\mathcal{SS}A$ under powerful parallel computers and then distribute matrix $S^{'}$ to several machines that can handle $S^{'}$ as one matrix.

In conclusion, being part of a competition is intriguing. We have learned a lot just by trying to implement our theory. Our solver is still at version one and we hope that at a later point we can make it run faster and may be answer questions of a more complicated nature. By finding stable extensions, we can also find some complete extensions by definition. Unfortunately, this kind of questions is not part of $A\mathcal{SS}A$. We have decided to provide support only for questions related to stable semantics:

- Given an abstract argumentation framework, determine some extension
- Given an abstract argumentation framework, determine all extensions
- Given an abstract argumentation framework and some argument, decide whether the given argument is credulously inferred
- Given an abstract argumentation framework and some argument, decide whether the given argument is skeptically inferred

3 Description

Our solver is written in Java and uses the **trivial graph format**. Beside the desire to get a good score, competitions are always interesting and informative. Due to the former, we decided to participate in this contest despite the fact that there is room for improvement on how fast $A\mathcal{SSA}$ can perform. We believe that our idea is unique and we hope that many researches will get inspired.

On receiving a file containing the argumentation framework in tgf form we read it and create its matrix representation, A. We then create all possible instances of selected arguments and combine them into a matrix, $S^{'}$. Based on matrix operations and specifically left and right matrix multiplication we can navigate inside the argumentation. Finding then which arguments attack other arguments and which arguments are under attack, we extract all conflict free sets. Based on some comparison to the system output matrices and $S^{'}$, we then find all stable extensions.

What we have learned while implementing the solver is that a more targeted selection of arguments is necessary as $A\mathcal{SSA}$ can quickly run out of memory. Our solver contains the folders $bin, classes, data, lib$ and src. Under the $data$ folder a tgf file format is expected. To run $A\mathcal{SSA}$ under bin, click compile and then click $Solver$. Results are printed on the screen.

References

1. Pietro Baroni, Martin Caminada, and Massimiliano Giacomin, *An introduction to argumentation semantics*, Knowledge Eng. Review **26** (2011), no. 4, 365–410.
2. Phan Minh Dung, *On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games*, Artif. Intell. **77** (1995), no. 2, 321–358.
3. Phan Minh Dung, Paolo Mancarella, and Francesca Toni, *Computing ideal sceptical argumentation*, Artif. Intell. **171** (2007), no. 10-15, 642–674.
4. Christos H Papadimitriou, *Computational complexity*, John Wiley and Sons Ltd., 2003.
5. Gilbert W Stewart, *Introduction to matrix computations*, (1973).
6. Andrew James Stothers, *On the complexity of matrix multiplication*, (2010).