



Christian Pasero, BSc

# **Computation of Clustered Argumentation Frameworks via Boolean Satisfiability**

## **MASTER'S THESIS**

to achieve the university degree of

Master of Science

Master's degree programme: Computer Science

submitted to

**Graz University of Technology**

## **Supervisor**

Johannes P. Wallner, Ass.Prof. Dipl.-Ing. Dr.techn. BSc.

Institute of Software Technology

Graz, July 12, 2024



# Abstract

English abstract of your thesis



# Kurzfassung

Deutsche Kurzfassung der Abschlussarbeit



# Acknowledgements

Thanks to everyone who made this thesis possible





# Contents

<b>1</b>	<b>Introduction</b>	<b>17</b>
<b>2</b>	<b>Background</b>	<b>19</b>
2.1	Argumentation Frameworks . . . . .	19
2.2	Clustering of Argumentation Frameworks . . . . .	19
2.3	SAT-Solver . . . . .	19
<b>3</b>	<b>Algorithm</b>	<b>21</b>
3.1	Concretizing Singletons . . . . .	21
3.2	Algorithmic Approach to Compute Faithful Clusterings . . . . .	23
3.3	Heuristics and Refinements . . . . .	23
<b>4</b>	<b>Implementation</b>	<b>25</b>
4.1	Creating AFs . . . . .	25
4.2	BFS and DFS Approach . . . . .	25
4.3	Generating Semantic Sets . . . . .	25
4.4	Faithful/Spurious Determination . . . . .	25
<b>5</b>	<b>Related Works</b>	<b>27</b>
<b>6</b>	<b>Conclusion</b>	<b>29</b>



# List of Figures

3.1	Concrete AF $F$ . . . . .	21
3.2	Abstract AF $F'$ . . . . .	21
3.3	Example: Concretization of arguments . . . . .	21
3.4	Concretized AF $F''$ after Step 1 . . . . .	21
3.5	Concretized AF $F''$ after Step 2 . . . . .	22
3.6	Concretized AF $F''$ after Step 3 . . . . .	22
3.7	Concretized AF $F''$ after Step 4 . . . . .	22
3.8	Concretized AF $F''$ after Step 5 . . . . .	22



## List of Tables



# List of Acronyms and Symbols

<b>AF</b>	Argumentation Framework
<b>AI</b>	Artificial Intelligence
<b>ASP</b>	Answer Set Programming
<b>cf</b>	Conflict-Free
<b>adm</b>	Admissible
<b>stb</b>	Stable
<b>BFS</b>	Breadth First Search
<b>DFS</b>	Depth First Search





# 1 Introduction

We all encounter arguments in our lives frequently. When talking to friends, listening to political discussions, or even making decisions in our head. These arguments can get heated and complex since humans have different beliefs and motivations. Finding a common ground or a "correct" conclusion is complicated and sometimes impossible. However, these imperfections are what make us humans. Artificial Intelligence (AI), conversely, needs to act precisely and logically [5]. That is why much research is being done on knowledge representation and reasoning [4] [10].

When observing arguments objectively, we can distinguish between facts and premises. A fact represents a particular aspect of the argument, and the premises link the different facts together by supporting one another or contradicting. When stating two opposing facts (f.e. *the square  $x$  is red* and *the square  $x$  is blue*), they are contradicting each other. While proving the correctness of facts is very important, refuting facts is even more critical in an argument.

Thus, when modeling arguments, the premises are counterarguments of the linked facts. If a fact or, i.e., an argument  $a$  is a counterargument of another argument  $b$ , we can say that  $a$  attacks  $b$ . With this generalization, we can abstract our model with directed graphs. The arguments are represented as nodes, and the attacks as directed edges [7]. With this abstraction, we can define Argumentation Frameworks (AFs) and use them to evaluate conclusions [9]. Drawing a conclusion from the AF can be challenging and tears down to the definition of semantics.

A semantic defines a subset of argument sets that satisfy the semantic-specific rules. The specific rules can be defined via a boolean formula. They can be used to encode the AFs to be solvable with different boolean solvers like Answer Set Programming (ASP) [2] or, as in our case, with a Boolean Satisfiability Solver (SAT-Solver) [1]. Unfortunately, drawing a conclusion from an AF can be challenging, e.g., it can be NP-complete to decide whether an argument is acceptable under a specific argumentation semantics [8].

*TODO: Explain abstract faithful AF generation complex*

One of the first papers describing the concept was written by Dung [6] in 1995. Since then, there has been more and more interest in AFs due to the artificial intelligence community. The argumentation systems and semantics have been modified and improved over the years, and another abstraction layer has been added. This specific abstraction layer is called *clustering* and generalizes multiple arguments into one bundled cluster [11]. Clustering is a technique to reduce the number of arguments without changing the conclusion, which in this instance would be the sets produced by a specific semantic. When producing a clustered (*abstract*) AF, which produces the same semantic sets as the non-clustered (*concrete*) AF, the abstract AF is defined to be *faithful*. While each concrete semantic set has a directly mapped abstract semantic set, not every abstract

semantic set has to have a directly mapped concrete semantic set. If we create an abstract AF that produces a semantic set that cannot be mapped to a concrete semantic set, we call it *spurious*.

Creating abstract, faithful AFs can be challenging and is the main focus of this paper. We created one of the first tools [3] to produce an abstract AFs based on a concrete AFs. We cover different setups and usages, including different semantics and base functionalities:

- Generate semantic sets of a concrete- or abstract AF. The sets calculated iteratively or all at once. The covered semantics are Conflict-Free (cf), Admissible (adm), and Stable (stb), which can be selected throughout the project independently by a parameter in the command line.
- Determine faithfulness or spuriousness by providing two AFs. We provide two approaches, Breadth First Search (BFS) and Depth First Search (DFS), which alter the procedure. While BFS calculates all the semantic sets of the two AFs first and then compares them, DFS calculates iteratively a semantic set of the abstract AF and then verifies it with the concrete AF. The algorithm selection is done via a command line parameter.
- Concretize a set of arguments (i.e., pull out arguments from the cluster) given the concrete AF and an abstract AF (faithful or spurious), and provide faithfulness (by concretizing other arguments not specified in the concretize list as well). The user provides the concretized arguments via a command line parameter.
- ...

*TODO: Further contributions*

*TODO: give pointers to why are non-trivial to obtain*

*TODO: Choice of methods to obtain results*

*TODO: How big AFs are still feasible to solve*

## 2 Background

### 2.1 Argumentation Frameworks

*TODO: What are AFs*

*TODO: Why are they used*

*TODO: Example*

*TODO: Definitions of Semantics*

### 2.2 Clustering of Argumentation Frameworks

*TODO: What are clusters*

*TODO: Why do we need clusters*

*TODO: Definition of Semantics*

### 2.3 SAT-Solver

*TODO: What are SAT-Solvers*

*TODO: Complexity of SAT-Problems*

*TODO: Where and how do we use SAT-Solvers in the research*



## 3 Algorithm

### 3.1 Concretizing Singletons

Concretizing a list of arguments is done iteratively by deep copying the abstract AF  $F'$  to create a new AF  $F''$  and mutating it. The mutation is guided by five steps considering the unchanged abstract AF  $F'$  and the concrete AF  $F$ . To improve the understanding of each step, we accompany the explanation with the example depicted in 3.3, where we concretize the arguments  $a$  and  $b$ .

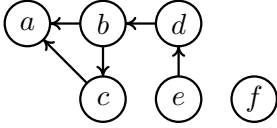


Figure 3.1: Concrete AF  $F$

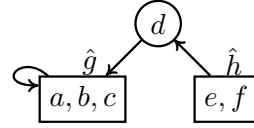


Figure 3.2: Abstract AF  $F'$

Figure 3.3: Example: Concretization of arguments

**Step 1:** Each argument needing concretization is first removed from the parent cluster and added as a singleton in  $F''$ . If an argument is not part of a cluster or is invalid, we remove it and continue with the filtered valid list. We do not consider attacks in this step since they depend on the concrete- and abstract AFs.

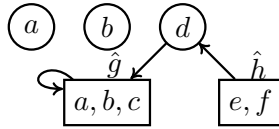


Figure 3.4: Concretized AF  $F''$  after Step 1

**Step 2:** We add the new attacks from all concretized arguments to the remaining clusters. We must do this after removing the arguments from the clusters because if an argument  $a$  attacks argument  $b$  in the concrete AF, and  $b$  is part of the cluster  $F'$  in the abstract AF, by concretizing  $b$ , the attack  $(a, F')$  would not be valid anymore.

**Step 3:** After adding the new attacks, we need to check which attacks from  $F'$  are still valid in  $F''$ . If an attack is not valid anymore through the concretization, we remove it in

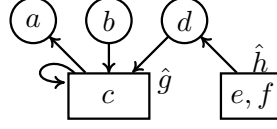


Figure 3.5: Concretized AF  $F''$  after Step 2

$F''$ . An attack is not valid anymore; if we remove one of the arguments being attacked or attacked by argument  $a$  from the cluster  $f$  and no other attack exists, s.t.  $a$  is attacked from/attacking an argument within  $f$ . Selfattacks of clusters could also change by the concretization of arguments. Therefore, we need to check the clusters from which the arguments are concretized.

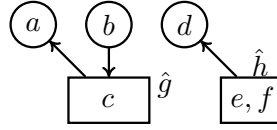


Figure 3.6: Concretized AF  $F''$  after Step 3

**Step 4:** In this step we add the new attacks between the singletons. Due to the fact, that we copied all the attacks from  $F'$ , we only have to take into consideration the attacks from or to the concretized singletons. So instead of iterating over all singletons of the AF, we can limit the attack creation to the concretized singletons.

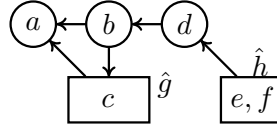


Figure 3.7: Concretized AF  $F''$  after Step 4

**Step 5:** The last step is to clean up the argumentation framework  $F''$  by removing all empty clusters and mutating the clusters with exactly one singleton to the mentioned singleton.

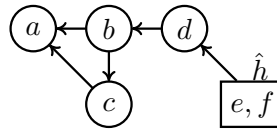


Figure 3.8: Concretized AF  $F''$  after Step 5

## **3.2 Algorithmic Approach to Compute Faithful Clusterings**

*TODO: Concretize singletons of clustered AF algorithm*

## **3.3 Heuristics and Refinements**

*TODO: Define every Heuristic and refinement we used for each semantic*





## 4 Implementation

### 4.1 Creating AFs

*TODO: Explain AF creation algorithms (Random + Grid-Based)*

### 4.2 BFS and DFS Approach

*TODO: BFS and DFS approach in current research + when BFS is better than DFS*

### 4.3 Generating Semantic Sets

*TODO: Semantic sets generation algorithm*

### 4.4 Faithful/Spurious Determination

*TODO: Determine faithful/spurious algorithm*



## 5 Related Works



## 6 Conclusion



# Bibliography

- [1] Leila Amgoud and Caroline Devred. “Argumentation frameworks as constraint satisfaction problems.” In: *Ann. Math. Artif. Intell.* 69.1 (2013), pp. 131–148. DOI: 10.1007/S10472-013-9343-0. URL: <https://doi.org/10.1007/s10472-013-9343-0>.
- [2] Günther Charwat, Johannes Peter Wallner, and Stefan Woltran. “Utilizing ASP for Generating and Visualizing Argumentation Frameworks.” In: *CoRR* abs/1301.1388 (2013). arXiv: 1301.1388. URL: <http://arxiv.org/abs/1301.1388>.
- [3] Pasero Christian. *argumentation-framework-clustering*. <https://github.com/p4s3r0/argumentation-framework-clustering>. 2024.
- [4] James P. Delgrande et al. “Current and Future Challenges in Knowledge Representation and Reasoning (Dagstuhl Perspectives Workshop 22282).” In: *Dagstuhl Manifestos* 10.1 (2024), pp. 1–61. DOI: 10.4230/DAGMAN.10.1.1. URL: <https://doi.org/10.4230/DagMan.10.1.1>.
- [5] Emmanuelle Dietz, Antonis C. Kakas, and Loizos Michael. “Editorial: Computational argumentation: a foundation for human-centric AI.” In: *Frontiers Artif. Intell.* 7 (2024). DOI: 10.3389/FRAI.2024.1382426. URL: <https://doi.org/10.3389/frai.2024.1382426>.
- [6] Phan Minh Dung. “On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and N-Person Games.” In: *Artificial Intelligence* 77.2 (1995), pp. 321–357. DOI: 10.1016/0004-3702(94)00041-X.
- [7] Phan Minh Dung. “On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games.” In: *Artificial Intelligence* 77.2 (1995), pp. 321–357. ISSN: 0004-3702. DOI: [https://doi.org/10.1016/0004-3702\(94\)00041-X](https://doi.org/10.1016/0004-3702(94)00041-X). URL: <https://www.sciencedirect.com/science/article/pii/000437029400041X>.
- [8] Wolfgang Dvorák et al. “The complexity landscape of claim-augmented argumentation frameworks.” In: *Artif. Intell.* 317 (2023), p. 103873. DOI: 10.1016/J.ARTINT.2023.103873. URL: <https://doi.org/10.1016/j.artint.2023.103873>.
- [9] Hector Geffner. “A Formal Framework for Clausal Modeling and Argumentation.” In: *Practical Reasoning, International Conference on Formal and Applied Practical Reasoning, FAPR '96, Bonn, Germany, June 3-7, 1996, Proceedings*. Ed. by Dov M. Gabbay and Hans Jürgen Ohlbach. Vol. 1085. Lecture Notes in Computer Science. Springer, 1996, pp. 208–222. DOI: 10.1007/3-540-61313-7\74. URL: [https://doi.org/10.1007/3-540-61313-7%5C\\_74](https://doi.org/10.1007/3-540-61313-7%5C_74).

- [10] Dragos Constantin Popescu and Ioan Dumitrache. “Knowledge representation and reasoning using interconnected uncertain rules for describing workflows in complex systems.” In: *Inf. Fusion* 93 (2023), pp. 412–428. DOI: 10.1016/J.INFFUS.2023.01.007. URL: <https://doi.org/10.1016/j.inffus.2023.01.007>.
- [11] Zeynep G. Saribatur and Johannes Peter Wallner. “Existential Abstraction on Argumentation Frameworks via Clustering.” In: *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*. Ed. by Meghyn Bienvenu, Gerhard Lake-meyer, and Esra Erdem. 2021, pp. 549–559. DOI: 10.24963/KR.2021/52. URL: <https://doi.org/10.24963/kr.2021/52>.