Edinburgh Napier University

Week 9 Report

Computing abstract argumentation semantics

Marcin Szczot
40180425

supervised by
Dr. S. Wells

March 14, 2018

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The aim of this project is to implement high performance, a scalable library for computing abstract argumentation semantics (complete, preferred, stable, grounded, semi-stable and stage semantics) for given argumentation framework. Furthermore, it should have the ability to compute if the given argument from the given argumentation framework is credulously or skeptically inferred for given semantics.
This project can be divided into four separate stages: research, design, development and evaluation. Although the stages proposed can be performed in the sequence, work on this project will be performed in an iterative way. In the work plan for the project, three iterations of design, development and testing have been identified following a period of research, background reading and writing the literature review. However, further research might be required in the later stages.

Each stage of the project has overall goals that will need to be achieved. Research and review stage is the vital part of the project. This will allow to build a better understanding of the problem domain and investigate existing solutions. Following goals can be identified at this stage:

1. Research concept of argumentation framework and abstract argumentation semantics

2. Research and review existing algorithms for solving the problem

3. Review solvers submitted to ICCMA 2017
4. Perform technology review

Once enough research will be performed, the project can proceed to next stages. The knowledge gathered will be used to define the requirements and propose an appropriate solution. The design, development and testing stages will be performed in an iterative way. Goals for those stages include:

1. Perform Requirements Analysis
2. Define Test Plan
3. Based on the requirements analysis design appropriate solution
4. Implement the proposed solution
5. Test the solution to verify implemented requirements
6. Perform benchmark testing to evaluate the performance and correctness of the solution

As mentioned above, 3 iterations of those stages have been initially scheduled. Additionally, during any of those stages, further research might be required, hence it might be needed to go back to research stage.
The outcome of the project will be the developed library for computing abstract argumentation framework. Furthermore, depending on the evaluation process, the library will be used to develop a solver for abstract argumentation framework to be submitted to International Competition on Computational Models of Argumentation [ICCMA, 2018]. Hence, there are number of requirements that will have to be included in the implementation. Below is the list of the main requirements, but this list is not extensive and will be expanded during the course of a project:

1. Should be able to read argumentation from provided file in format apx or tgf

2. Should be able to compute the following semantics from the given argumentation framework and return all or some of the extensions computed:
   (a) Complete Semantics
   (b) Preferred Semantics
   (c) Stable Semantics
   (d) Semi-Stable Semantics
   (e) Stage Semantics
   (f) Grounded Semantics
   (g) Ideal Semantics
3. Should take the type of task to be computed as an argument

The above list shows only the basic requirements for the software. The extended list can be found in Appendix B. Although this list is more comprehensive, during the course of the project, new requirements will be identified. Hence, this list will be a subject to regular updates, based on the progress of the project. Before each development stage, a set of requirements will be selected from the list.
The above requirements are functional requirements of the software. Additionally, there are a number of non-functional requirements that will have to be included in the design and implementation software. Those requirements are needed if the software will be submitted to the ICCMA competition.

Chapter 2

Background

2.1 Abstract Argumentation

Argumentation framework has been introduced by Dung [1995] and is central to the theory of abstract argumentation [Baroni et al., 2011]. It is defined as a pair of a set of arguments, and a binary relation representing the attack relationship between arguments [Dung, 1995].

Definition 2.1. Argumentation Framework
An argumentation framework is a pair $AF = <AR,\ attacks>$ in which AR is a set of finite arguments, and *attacks* is a binary relation on AR, hence $attacks \subseteq AR \times AR$, where $AR \times AR = \{(a,\ b) \mid a \in AR \text{ and } b \in AR\}$

In definition 2.1, AR represents a set of arguments and *attacks* represents set of pairs of arguments $(a,\ b)$, where $(a,\ b) \in attacks$. Each pair of arguments in *attacks* represents two arguments being in conflict. Hence, the arguments a and b are in conflict and the meaning of *attacks(a, b)* is that a attacks b. Based on this definition, we can conclude that the set of arguments AR is conflict-free if and only if there are no arguments a and b in AR such that a attacks b, or b attacks a [Dung, 1995].

The argumentation framework can be represented as directed graph where the nodes represent abstract arguments and edges the attack relation. This can be seen in figure 2.1, where argument a attacks argument b, which in turn attacks argument c. C is also attacked by argument d.

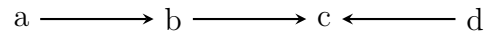

Figure 2.1: Argumentation Framework 2.2

This can also be presented using the following example from Konolige [1988]:

> Suppose Ralph normally goes fishing on Sundays, but on the Sunday which is Mother's day, he typically visits his parents. Furthermore, in the spring of each leap year,1‘ his parents take a vacation, so that they cannot be visited.

If we assume it is Sunday, Mother's day and a leap year, then three arguments can be formulated:

A Ralph goes fishing because it is Sunday.

B Ralph does not go fishing because it is Mother's day, hence he visits his parents.

C Ralph does not go visit his parents, because it is a leap year. Hence, they are on vacation.

In this example argument B attacks argument A and argument C attacks argument B. Since argument C can be justified, as it is not attacked, then B is defeated and does no longer form a reason against A anymore. We can say that argument C reinstates argument A [Caminada, 2004].

Dung in his paper [Dung, 1995] has also defined notions of *acceptable* and *admissible* arguments, which are as follow:

Definition 2.2. Acceptable argument
An argument $a \in AR$ is said to be *acceptable* with respect to set $S \subseteq AR$ if and only if for each argument $b \in AR$ such that $(b, a) \in attacks$, there is some argument $c \in S$ such that $(c, b) \in attacks$

Hence it can be said that the argument from the argumentation framework is acceptable with respect to the set only if it is either conflict free, or there exists an argument from the same set that defends given argument. Based on the definition 2.2, definition of *admissible* set can be concluded:

Definition 2.3. Admissible argument
A conflict-free set of arguments S is *admissible* if and only if each argument in S is acceptable with respect to S.

That means that the set of arguments can be described as admissible only if it is conflict-free and all of its arguments can be defended by other arguments from that set. Both definitions have important role in defining the semantics of abstract argumentation. An argumentation semantics can be described as teh formal definition of a method (declarative or procedural) ruling the argument evaluation process [Baroni and Giacomin, 2009]. Hence, they are used to evaluate if the arguments can be justified, by being defended by other arguments in the set, or rejected. There are two approaches for defining argumentation semantics: extension and labelling based. In the extension-based approach a semantics definition specifies how to derive from an argumentation framework a set of extensions, where an extension E of an argumentation framework $<AR,\ attacks>$ is simply a subset of AR, intuitively representing a set of arguments which can "survive together" or are "collectively acceptable" [Baroni and Giacomin, 2009]. On the other hand labelling-based approach defines how the arguments can be labelled based on the predefined set of labels. Labels define the possible states of argument and those are as follow: *in*, when argument can be justified, *out* when argument is rejected, and *undecided* to any other argument. As shown by Modgil and Caminada, label-based approach is suitable for characterizing argumentation semantics [Modgil and Caminada, 2009].

2.2 Argumentation Semantics

The original concept of abstract argumentation semantics included complete, stable, preferred and grounded extensions [Dung, 1995], which was extended by stage [Verheij, 1996], ideal [Dung et al., 2007], and semi-stable [Caminada, 2006] semantics. All the sematics are based around the definitions of acceptable and admissible arguments. Definitions of the original semantics as described by Dung [1995] are as follow:

Definition 2.4. Preferred Extension
A *preferred extension* of an argumentation framework AF is a maximal (with

respect to set inclusion) admissible set of AF.

From the definition 2.4 it can be seen that the preferred extension is the largest set that is able to defend itself from attacks. If we consider example 2.2 we can see that the extension $E_{PR} = \{\{a,c\}, \{b\}\}$. Argument c is attacked by argument b, which in turn is defeated by a, hence a and c are preferred extension. Furthermore, b is attacked by a, but it defends itself. Hence argument b is also preferred extension.

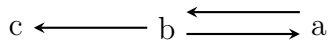

Figure 2.2: Argumentation Framework 2.2

Definition 2.5. Stable Extension
A *conflict-free* set of arguments S is called a *stable extension* if and only if S attacks each argument which does not belong to S.

Stable extension has more aggressive approach than preferred extension. As in definition 2.5, the extension needs to attack all arguments that are not included in it. Based on argumentation framework from example 2.2, the stable extension $E_{ST}(AF_{2.2}) = \{\{a,c\}, \{b\}\}$, which is the same as for preferred extension. This is because argument a attacks b, hence a and c are included, and argument b attack both arguments a and c. Looking at more complex example in figure 2.3, three stable extensions can be identified as follow $E_{ST}(AF_{2.3}) = \{\{a,c\}, \{a,d\}, \{b,d\}\}$.

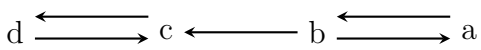

Figure 2.3: Argumentation Framework 2.3

Dung [1995] in his paper also defined a skeptical semantics known as grounded extension. This extension is defined in terms of *characteristic function*, which in turn is defined as:

Definition 2.6. Characteristic Function
The *characteristic function*, denoted by F_{AF}, of an argumentation framework

$AF = <AR, attacks>$ is defined as follows:

$$F_{AF} : 2^{AR} \rightarrow 2^{AR}$$

$$F_{AF}(S) = \{A|\text{A is acceptable with respect to S}\}$$

Definition 2.7. Grounded Extension
The *grounded extension* of an argumentation framework AF, denoted by GE_{AF}, is the least fixed point of F_{AF}.

Based on the definition of characteristic function and grounded extension it can be noted that the grounded extension has very restrictive approach for its computation, as it only includes the arguments whose defence is "rooted" in initial unattacked argument [Baroni and Giacomin, 2009]. Hence, the grounded extension from figure 2.4 is $E_{GR}(AF_{2.4}) = \{\{\text{a,c}\}\}$. Starting from argument *a*, as it is the only unattacked argument, we reject argument *b* and include *c*. Furthermore, argument *d* is being rejected, as it is attacked by *c*. Although, argument *e* attacks argument *f*, it is attacked by *f* as well. Hence, it cannot be part of grounded extension.

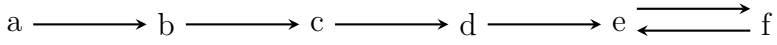

Figure 2.4: Argumentation Framework 2.4

Definition 2.8. Complete Extension
An admissible set S of arguments is called *complete exension* if and only if each argument which is acceptable with respect to S, belongs to S.

Complete extension is defined as a set which is able to defend itself and includes all arguments it defends [Baroni and Giacomin, 2009]. Based on this, we can see that there are three complete extensions in figure 2.2, and those are $E_{CO}(AF_{2.2}) = \{\emptyset, \{\text{a,c}\}, \{\text{b}\}\}$. With a more complex argumentation framework in figure 2.3, it can be seen that the complete extension will definitely include empty set, as there are no initial arguments, argument *a*, as it does not defend argument *c* from *d*, and argument *d*, as it only defends itself. Furthermore, *b* attacks *c*, the only attacked of argument *d*, hence set

{b,d} is a complete extension as well. Additionally, sets {a,c} and {a,d} are also compltete extensions as both of them are admissible. Hence, the complete extension is $E_{CO}(AF_{2.3}) = \{\emptyset, \{a\}, \{d\}, \{a,c\}\ \{a,d\}, \{b,d\}\}$.

As mentioned earlier, Dung [1995] semantics were further extended with additional semantics. Their definitions are as follow:

Definition 2.9. Semi-stable Extension
Let (*AR*, *attacks*) be an argumentation framework and $Args \subseteq AR$. *Args* is called *semi-stable extension* if and only if *Args* is a complete extension where $Args \cup Args^+$ is maximal.

The idea for semi-stable extension consist in expressing a definite opinion on the largest possible set of arguments, while restricting as much as possible those which are left undecided [Baroni et al., 2011], where labelling has been used. Hence, the stable-extension in argumentation framework in figure 2.5 is $E_{SST}(AF_{2.5}) = \{\{b,d\}\}$, which is also a preferred extension.

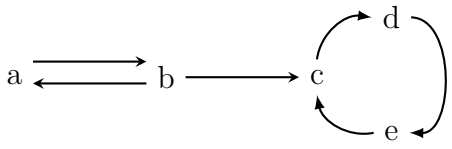

Figure 2.5: Argumentation Framework 2.5

Definition 2.10. Ideal Extension
The *ideal extension* is the greatest with respect to set inclusion admissible set that is a subset of each preferred extension.

By its definition the ideal extension satisfies I-maximality and admissibility principles [Baroni and Giacomin, 2009]. The I-maximiality is a property of the set of extensions, without reference to any generic criterion [Dunne and Bench-Capon, 2006]. It is defined as:

Definition 2.11. I-maximality
A set of extensions E is I-maximal if and only if $\forall E_1, E_2 \in E$, if $E_1 \subseteq E_2$ then $E_1 = E_2$.

Perhaps, the ideal semantics can be best explained using the notion of a judgement aggregation context intoduced by Caminada and Pigozzi [2011]. Given an argumentation framework, if we assume a group of people who all try to accept as much as possible the sets of arguments they all agree on needs to be examined and checked whether this position is still defensible. However, if they are not defensible some of the arguments needs to be abstained from instead of accepting or rejecting them until it becomes defensible. The result is the ideal extension [Baroni et al., 2011].

Definition 2.12. Stage Extension
Let (AR, *attacks*) be an argumentation framework. A *stage extension* conflict-free set $Args \subseteq AR$, where $Args \cup Args^{+}$ is maximal with respect to set inclusion among all conflict-free sets.

Stage extension introduced by Verheij [1996] can also be explained in terms of labelling. It is a conflict-free extension where number of arguments labeled *undec* is minimal.

Above semantics can be used to decide whether a given argument, or set of arguments can be accepted in terms of set inclusion. Based on the definitions a number of inclusions between the sets can be conluded. Those are:

- Grounded extension is the smallest complete extension
- Every preferred extesion is also complete
- Every stable extension is also semi-stable and stage extension

Furthermore, it can be seen that the complete extension provides the link between preferred extension (credulous semantic), and grounded extension (skeptical smenatics) [Dung, 1995]. It can also be concluded seen that the maximal complete extension is also maximal admissible set. On the other hand the grounded extension is determined by the minimal complete extension [Gaggl, 2009]. The relations between the semantics can be seen in figure 2.6.

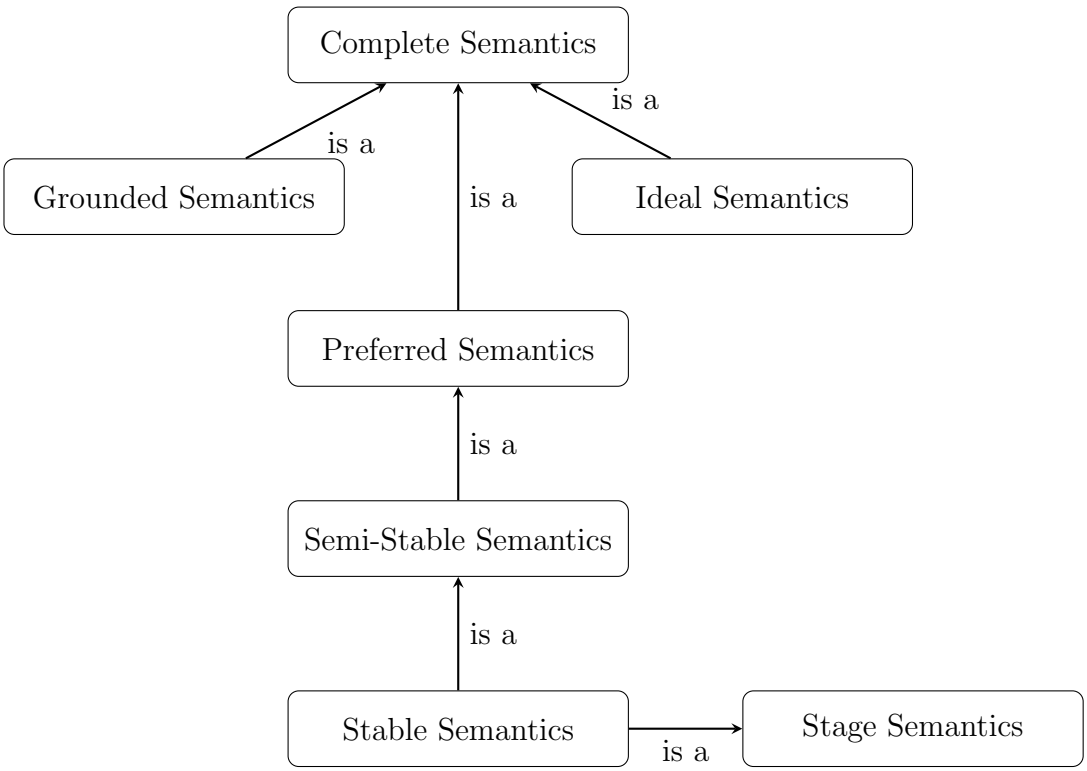

Figure 2.6: Relations of semantics

Additional notion of credulous and sceptical semantics has been introduced. Usually, a credulous semantics is intended to capture as much information as possible [Dix and Pereira, 1995], while skeptical makes less committed choices about the justification of the arguments [Baroni and Giacomin, 2007]. As mentioned previosly grounded semantic is a good example of skeptical extension as it always yields exactly one extension, and this extension is admissible [Caminada, 2007].

Furthermore, argument can be defined as credulous or skeptical in terms of its acceptability. As describe by Arieli [2015] an argument $a \in Args$ is credulously accepted by one type of semantics only if it belongs to some of the semantic extensions of a given argumentation framework. On the other hand, the argument b is skeptically accepted by the semantic if it belongs to all of its extensions of a given argumentation framework. This can be represented using argumentation framework 2.7. It has 5 complete extensions: $\emptyset$, b, e,

b, e, a, c, e, however none of the arguments is skeptically accepted, while *a*, *b*, *c* and *e* are credulously accepted. On the other hand, {*b, e*} and {*a, c, e*} are preferred, stable, semi-stable and stage extensions of argumentation framework 2.7. It can be seen that only argument *e* is skeptically accepted for all those semantics, while *a*, *b* and *c* are credulously accepted.

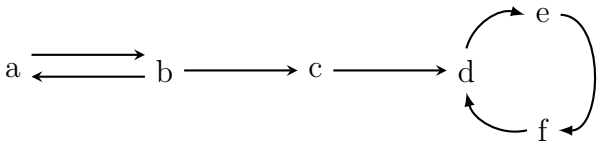

Figure 2.7: Argumentation Framework 2.7

Chapter 3

Plan of Work

The overall schedule for the project can be seen in appendix C. At the beginning of the project the majority of the work will be based on research and planning of the project. This will allow to build the knowledge neccessary for advancing the project to achieve its goal. During the later stages of the project three iterations of design, development and testing processes have been scheduled. Initially one week is allocated to design and testing stages and two weeks for development. Furthermore, during the weekly meetings with project supervisor, Simon Wells, progress of work is reviewed and a set of goals is set to be accomplished for the following meeting. Notes from the meetings can be seen in appendix D.

Chapter 4

Project Evaluation

Evaluation is the critical part of the project. Hence, the developed software will need to be evaluated based on three properties: performance, scalability and correctness.

4.1 Performance Evaluation

In order to evaluate the performance of the proposed solution benchmark tests will be performed. The performance of the software will be compared to solvers submitted for the second edition of International Competition on Computational Models of Argumentation (http://argumentationcompetition.org/). This will allow to appropriately evaluate the performance in comparison to existing solutions. In order to perform benchmark testing the base line of execution times will be created based on the solvers submitted to ICCMA 2017. This tests will include computing semantics from provided argumentation frameworks and evaluating from the given argument and argumentation framework, if the argument is credulously or sceptically inferred in any of the semantics. Seven main tracks are identified, where each track will represent different semantic. Following semantics will be used for benchmark testing:

1. Complete Semantics - CO
2. Preferred Semantics - PR
3. Stable Semantics - ST

4. Semi-Stable Semantics - SST
5. Stage Semantics - STG
6. Grounded Semantics - GR
7. Ideal Semantics - ID

For each track the system will need to solve following reasoning problems, with the exception for Grounded and Ideal semantics where only A and C are applicable:

1. Given an abstract argumentation framework, determine some extension
   SE-σ: Given F=(A,R) return some set E $\subseteq$ A that is a σ-extension
2. Given an abstract argumentation framweork, determine all extensions
   EE-σ: Given F=(A,R) enumerate all sets E $\subseteq$ A that are σ-extensions
3. Given an abstract argumentation framework and some argument, decide whether the given argument is credulously inferred
   DC-σ : Given F=(A,R), a $\in$ A decide whether a is credulously accepted under σ
4. Given an abstract argumentation framework and some argument, decide whether the given argument is skeptically inferred
   DS-σ: Given F=(A,R), a $\in$ A decide whether a is skeptically accepted under σ

There are 4 different sets of argumentation frameworks that will be used, where all of them were used for testing solvers submitted to ICCMA 2017. Each set consist a number of abstract argumentation frameworks in formats *apx* and *tgf*. Furthermore, each group of argumentation frameworks are used for a specific tasks. Graph representation mapped in yEd of one of the argumentation frameworks can be seen in appendix F. The list of relations between the sets and the tasks is as follow:

- Set A - DS-PR, EE-PR, EE-CO, DS-SST, DC-SST, SE-SST, EE-SST, DS-STG, DC-STG, SE-STG, EE-STG

- Set B - DS-ST, DC-ST, SE-ST, EE-ST, DC-PR, SE-PR, DC-CO
- Set C - DS-CO, SE-CO, DC-GR, SE-GR
- Set D - DC-ID, SE-ID

As mentioned above, prior to running the benchmark tests, solvers submitted to ICCMA 2017 will need to be tested with the available argumentation frameworks to evaluate their performance in the given environment. Full list of those solvers can be found in table E.1 with the tasks supported by them in table E.2 in appendix E.

4.2 Scalability Evaluation

Scalability is another important property that the software will be evaluated on. In order to properly test it, the software will be required to perform the same tasks as described above, but for the sets of larger argumentation frameworks. The argumentation frameworks will be prepared in advanced using one of the generators for random argumenation frameworks, such as AFBenchGen2 [Cerutti et al., 2016]. This will allow to specify the number of arguments required per framework.
For the purpose of this test, set of argumentation frameworks will be produced with increasing number of arguments. This will allow to test the solution in terms of scalability of computing the semantics in larg data sets. Comparing the time taken to perform the tasks with argumentation frameworks of different sizes will allow to evaluate how the software scales.

4.3 Correctness Evaluation

Correctness of the software relates to the correctness of the output of computation. In order to evaluate it the set tasks described above will be executed, and the output of the software verified. However, since the generated argumentation frameworks only consists of arguments and attacks, the correct output will need to be verified manually. Hence, the size of the argumentation frameworks need to be of a reasonable size to evaluate them manually.
Testing correctness of the software proves to be problematic with larger data

sets. Hence, further research will need to be performed to optimize the this tasks. Type of evaluation of correctness of existing solvers will be explored. Furthermore, formal verification methods for software might be used. Since they comprise a set of techniques for proving the correctness of a software for a possible combination of input values [Murthy and Sekharam, 2009],it will help in the evaluation. Different approaches will be explored prior to testing.

Bibliography

Arieli, O. (2015). Conflict-free and conflict-tolerant semantics for constrained argumentation frameworks. *Journal of Applied Logic*, 13(4):582–604.

Baroni, P., Caminada, M., and Giacomin, M. (2011). An introduction to argumentation semantics. *The Knowledge Engineering Review*, 26(4):365–410.

Baroni, P. and Giacomin, M. (2007). Comparing argumentation semantics with respect to skepticism. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 210–221. Springer.

Baroni, P. and Giacomin, M. (2009). Semantics of abstract argument systems. In *Argumentation in artificial intelligence*, pages 25–44. Springer.

Caminada, M. (2006). Semi-stable semantics. *COMMA*, 144:121–130.

Caminada, M. (2007). Comparing two unique extension semantics for formal argumentation: ideal and eager. In *Proceedings of the 19th Belgian-Dutch conference on artificial intelligence (BNAIC 2007)*, pages 81–87. Utrecht University Press.

Caminada, M. and Pigozzi, G. (2011). On judgment aggregation in abstract argumentation. *Autonomous Agents and Multi-Agent Systems*, 22(1):64–102.

Caminada, M. W. A. (2004). For the sake of the argument: explorations into argument-based reasoning.

Cerutti, F., Vallati, M., and Giacomin, M. (2016). A generator for random argumentation frameworks. *ICCMA 2017*.

Dix, J. and Pereira, L. M. (1995). *NON-MONOTONIC EXTENSIONS OF LOGIC PROGRAMMING: ICLP'94 Workshop Santa Margherita Ligure, Italy, June 1994 Selected Papers*. Springer-Verlag.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357.

Dung, P. M., Mancarella, P., and Toni, F. (2007). Computing ideal sceptical argumentation. *Artificial Intelligence*, 171(10-15):642–674.

Dunne, P. E. and Bench-Capon, T. J. (2006). *Computational Models of Argument: Proceedings of COMMA 2006*, volume 144. IOS Press.

Gaggl, S. A. (2009). *Solving argumentation frameworks using answer set programming*. na.

ICCMA (2018). ICCMA - Home. [Online; accessed 12. Feb. 2018].

Konolige, K. (1988). Defeasible argumentation in reasoning about events. In *Methodologies for Intelligent Systems*, pages 380–390.

Modgil, S. and Caminada, M. (2009). Proof theories and algorithms for abstract argumentation frameworks. In *Argumentation in artificial intelligence*, pages 105–129. Springer.

Murthy, S. and Sekharam, K. R. (2009). Software reliability through theorem proving. *Defence Science Journal*, 59(3):314.

Verheij, B. (1996). Two approaches to dialectical argumentation: admissible sets and argumentation stages. *Proc. NAIC*, 96:357–368.

Appendices

Appendix A

Initial Project Overview

Initial Project Overview
SOC10101 Honours Project (40 Credits)
Title of Project: Computing abstract argumentation semantics

A.1 Overview of Project Content and Milestones

The aim of this project is to implement scalable software for computing semantics (preferred, stable, grounded, etc.) of abstract argumentation from the given argumentation framework. This project will have a number of milestones that will need to be achieved. They include research of the argumentation framework semantics based on Dung's original notions of complete, grounded, preferred and stable semantics, as well as further proposed notions of semi-stable, stage and ideal semantics. Furthermore, different approaches, such as labelling-based and extension-based approaches along with different algorithms will be reviewed, investigated and evaluated. Additionally, software submitted to the International Competition on Computational Models of Argumentation will be investigated and evaluated based on the approaches used for computing the semantics. Finally, new software will be designed, developed, tested and benchmarked against the solvers from International Competition on Computational Models of Argumentation. The aim will be to produced scalable software for calculating different types of semantics from provided argumentation framework.

A.2 The Main Deliverable(s):

The main deliverables of this project will be the final report, which will include the review of argumentation framework, common algorithms used for calculating abstract argumentation semantics and software to be submitted to ICCMA. The report will also include a review of design, development and testing of the proposed solution. Furthermore, the developed software for computing abstract argumentation semantics will be part of deliverables.

A.3 The Target Audience for the Deliverable(s):

The target audience for projects' deliverables are the researches working on argumentation in computing.

A.4 The Work to be Undertaken:

This project requires research to be conducted on the argumentation framework and the notion of semantics introduced by Dung and extended by Caminada and others. Existing approaches and algorithms for labelling arguments and calculating the semantics will have to be researched and reviewed. Furthermore, new software for computing the semantics from given argumentation framework will be designed, implemented and tested. Existing solutions will be used for benchmarking the performance and scalability of the developed solution.

A.5 Additional Information / Knowledge Required:

In order to complete the project further knowledge is required in the abstract argumentation frameworks and their semantics. Furthermore, new system for computing the semantics will be developed in Python, hence extensive knowledge will be required for designing and implementing scalable approach for traversing large graphs of argumentation frameworks.

A.6 Information Sources that Provide a Context for the Project:

The project will be based on the abstract argumentation semantics introduced by Phan Minh Dung in the paper On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. The semantics were further extended with semi-stable semantics by Martin Caminada in his paper Semi-Stable Seman-

tics, stage semantics by Bart Verheij in Two approaches to dialectical argumentation: admissible sets and argumentation stages and ideal semantics by Phan Minh Dung, Paolo Mancarella and Francesca Toni in Computing ideal sceptical argumentation. Furthermore, the solvers submitted to the International Competition on Computational Models of Argumentations (http://argumentationcompetition.org/) will be used for benchmarking the proposed solution.

A.7 The Importance of the Project:

Research focus in AI for defeasible reasoning

A.8 The Key Challenge(s) to be Overcome:

Main challenge of this project will be to design and implement universal solution for computing different semantics (complete, preferred, stable, etc.) of abstract arguments in the provided argumentation framework. Furthermore, the solution must be able to scale appropriately and be able to perform calculation on supplied argumentation framework with different structures.

Appendix B

Requirement Analysis

Table B.1: Software Requirements

ID	Requirement	MoSCoW
1	Should read provided tgf file	Must have
2	Should parse tgf file	Must have
3	Should read provided apx file	Should have
4	Should be able to determine SOME complete extensions from given argumentation framework (SE-CO)	Should have
5	Should be able to determine ALL complete extensions from given argumentation framework (EE-CO)	Should have
6	Given the argumentation framework and some argument, should decide whether the given argument is credulously inferred in complete extension (DC-CO)	Should have
7	Given the argumentation framework and some argument, should decide whether the given argument is sceptically inferred in complete extension (DS-CO)	Should have
8	Should be able to determine SOME preferred semantics from given argumentation framework (SE-PR)	Should have
9	Should be able to determine ALL preferred extensions from given argumentation framework (EE-PR)	Should have

Continued on next page

Table B.1 – continued from previous page

ID	Requirement	MoSCoW
10	Given the argumentation framework and some argument, should decide whether the given argument is credulously inferred in preferred extension (DC-PR)	Should have
11	Given the argumentation framework and some argument, should decide whether the given argument is sceptically inferred in preferred extension (DS-PR)	Should have
12	Should be able to determine SOME stable extensions from given argumentation framework (SE-ST)	Should have
13	Should be able to determine ALL stable extensions from given argumentation framework (EE-ST)	Should have
14	Given the argumentation framework and some argument, should decide whether the given argument is credulously inferred in stable extension (DC-ST)	Should have
15	Given the argumentation framework and some argument, should decide whether the given argument is sceptically inferred in stable extension (DS-ST)	Should have
16	Should be able to determine SOME semi-stable extensions from given argumentation framework (SE-SST)	Should have
17	Should be able to determine ALL semi-stable extensions from given argumentation framework (EE-SST)	Should have
18	Given the argumentation framework and some argument, should decide whether the given argument is credulously inferred in semi-stable extension (DC-SST)	Should have
19	Given the argumentation framework and some argument, should decide whether the given argument is sceptically inferred in semi-stable extension (DS-SST)	Should have
20	Should be able to determine SOME stage extensions from given argumentation framework (SE-STG)	Should have

Continued on next page

Table B.1 – continued from previous page

ID	Requirement	MoSCoW
21	Should be able to determine ALL stage extensions from given argumentation framework (EE-STG)	Should have
22	Given the argumentation framework and some argument, should decide whether the given argument is credulously inferred in stage extension (DC-STG)	Should have
23	Given the argumentation framework and some argument, should decide whether the given argument is sceptically inferred in stage extension (DS-STG)	Should have
24	Should be able to determine SOME grounded extensions from given argumentation framework (SE-GR)	Should have
25	Should be able to determine ALL grounded extensions from given argumentation framework (EE-GR)	Should have
26	Should be able to determine SOME ideal extensions from given argumentation framework (SE-ID)	Should have
27	Should be able to determine ALL ideal extensions from given argumentation framework (SE-ID)	Should have
28	Should output the result of the computation to command line	Must have
29	Should take file format as a command line argument, i.e. -fo [file format]	Must have
30	Should take type of task as a command line argument, i.e. -p [EE-CO]	Must have
31	Should take file as a command line argument, i.e. -f [./argumentationFramework.apx]	Must have

Appendix C

Project Schedule

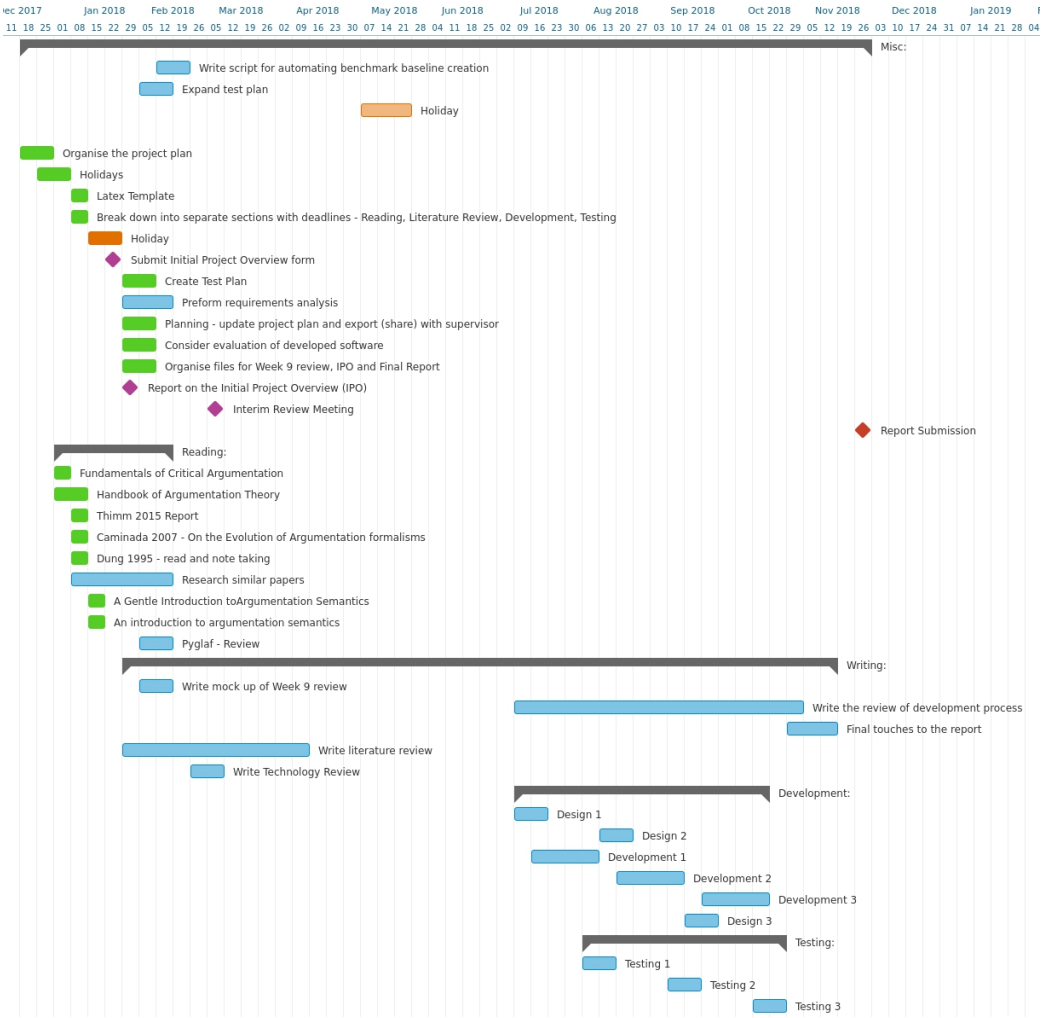

Figure C.1: Project Schedule

Appendix D

Meetings diaries

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Marcin Szczot **Supervisor:** Simon Wells

Date: 30/01/2018 **Last diary date:**

Objectives:

- Organise files for week 9 review, Initial Project Overview and Final Report
- Consider evaluation of developed software
- Planning – update project plan and export (share) with supervisor
- Write small review of each read paper (few paragraphs; useful for literature review)
- Perform requirements analysis
- Create Test Plan
- Technology review - Python

Progress:

- Files have been organised for week 9 review, IPO and Final Report, Test Plan, Requirements Analysis, Diary. All of them are backed up using git and stored in GitHub
- Evaluation of the software – benchmark testing based on the argumentation frameworks from ICCMA – correctness and performance. Baseline to be created from the winning solvers in ICCMA.
- Project plan has been updated and shared with supervisor
- Requirement analysis – created spreadsheet for requirement analysis (so far only 2 requirements). Needs to be expanded.
- Outline of the test plan has been created. Number of TODO items are inserted to expand/verify certain section of the test plan

Supervisor's Comments:

- Write mock up of the week 9 report – based on the feedback template – include the background reading/literature review
- Requirement analysis – functional and non-functional requirements – complete the list

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Marcin Szczot **Supervisor:** Simon Wells

Date: 06/02/2018 **Last diary date:** 30/01/2018

Objectives:

- Prepare a mock up of Week 9 report – this is to be based on the feedback template and include background reading
- Perform in depth requirement analysis – functional and non-functional requirements
- Expand Test Plan
- Technology review

Progress:

- Week 9 Report – document created with list of items that should be included and started writing sections for Background Reading, Goals and aims, Evaluation
- Expanded Requirements with additional requirements
- Setup environment in AWS for benchmark testing and setup couple of solvers with few argumentation frameworks from ICCMA (part of the methodology of evaluation)
- Started working on the script to automate the creation of baseline for benchmark testing

Supervisor's Comments:

- Correctness, performance and scalability to be part of the objectives
- Correctness, performance and scalability for evaluation – correctness how is it proved by other solvers especially in large graphs. Software engineering methods for proving software correctness.
- consider GPGPU for a solver
- Week 9 report – send draft to Simon once the required parts are completed
- python libraries for computing large sets of data – numpy, scipy

Appendix E

ICCMA 2017 Submissions

Table E.1: ICCMA 2017 Submissions

Solver	Author
argmat-clpb	Fuan Pu (Tsinghua University, China), Guiming Luo (Tsinghua University, China), Yucheng Chen (Tsinghua University, China).
argmat-dvisat	Fuan Pu (Tsinghua University, China), Guiming Luo (Tsinghua University, China), Ya Hang (Tsinghua University, China).
argmat-mpg	Fuan Pu (Tsinghua University, China), Guiming Luo (Tsinghua University, China), Ya Hang (Tsinghua University, China).
argmat-sat	Fuan Pu (Tsinghua University, China), Guiming Luo (Tsinghua University, China), Ya Hang (Tsinghua University, China).
ArgSemSAT	Federico Cerutti (Cardiff University, UK), Mauro Vallati (University of Huddersfield, UK), Massimiliano Giacomin (University of Brescia, Italy), Tobia Zanetti (University of Brescia, Italy).

Continued on next page

Table E.1 – continued from previous page

Solver	Author
ArgTools	Samer Nofal (German Jordanian University, Jordan), Katie Atkinson (University of Liverpool, UK), Paul E. Dunne (University of Liverpool, UK).
ASPrMin	Wolfgang Faber (University of Huddersfield, UK), Mauro Vallati (University of Huddersfield, UK), Federico Cerutti (Cardiff University, UK), Massimiliano Giacomin (University of Brescia, Italy).
cegartix	Wolfgang Dvořák (TU Wien, Austria), Matti Järvisalo (University of Helsinki, Finland), Johannes P. Wallner (TU Wien, Austria).
Chimærarg	Federico Cerutti (Cardiff University, UK), Mauro Vallati (University of Huddersfield, UK), Massimiliano Giacomin (University of Brescia, Italy).
ConArg	Stefano Bistarelli (Università di Perugia, Italy), Fabio Rossi (Università di Perugia, Italy), Francesco Santini (Università di Perugia, Italy).
CoQuiAAS	Jean-Marie Lagniez (Univ. Artois, France), Emmanuel Lonca (Univ. Artois, France), Jean-Guy Mailly (Univ. Paris Descartes, France).
EqArgSolver	Odinaldo Rodrigues (King's College London, UK).
gg-sts	Tomi Jahunen (Aalto University, Finland), Shahab Tasharrofi (Aalto University, Finland).
goDIAMOND	Stefan Ellmauthaler (Leipzig University, Germany), Hannes Strass (Leipzig University, Germany).
heureka	Nils Geilen (Universität Koblenz-Landau, Germany), Matthias Thimm (Universität Koblenz-Landau, Germany).
pyglaf	Mario Alviano (University of Calabria, Italy).

Table E.2: Tasks supported by solvers

	D3	CO				PR				ST				SST				STG				GR		ID	
		DC	DS	SE	EE	DC	DS	SE	EE	DC	DS	SE	EE	DC	DS	SE	EE	DC	DS	SE	EE	DC	SE	DC	SE
argmat-clpb		1	1	1	1					1	1	1	1									1	1		
argmat-dvisat	1	1	1	1	1	1	1	1	1	1	1	1	1									1	1	1	1
argmat-mpg	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
argmat-sat	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ArgSemSAT		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1					1	1		
ArgTools		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ASPrMin									1																
cegartix	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Chimærarg									1				1												
ConArg	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
CoQuiAAS	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
EqArgSolver	1	1	1	1	1	1	1	1	1	1	1	1	1									1	1		
gg-sts	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
goDIAMOND	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
heureka		1	1	1	1	1	1	1	1	1	1	1	1									1	1		
pyglaf	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Appendix F

Argumentation Framework Graph