FernUniversität in Hagen
Faculty of Mathematics and Computer Science

Artificial Intelligence Group

Development of Algorithms for the Elicitation of Sigma-Equivalent Abstract Argumentation Frameworks

Master's Thesis

in partial fulfillment of the requirements for
the degree of Master of Science (M.Sc.)
in Praktische Informatik

submitted by
Daniel Frahm

First examiner:	Prof. Dr. Matthias Thimm Artificial Intelligence Group
Advisor:	Isabelle Kuhlmann Artificial Intelligence Group

Statement

I hereby certify that this thesis has been composed by me and is based on my own work, that I did not use any further resources than specified – in particular no references unmentioned in the reference section – and that I did not submit this thesis to another examination before. The paper submission is identical to the submitted electronic version.

	Yes	No
I agree to have this thesis published in the library.	☐	☐
I agree to have this thesis published on the webpage of the artificial intelligence group.	☐	☐
The thesis text is available under a Creative Commons License (CC BY-SA 4.0).	☐	☐
The source code is available under a GNU General Public License (GPLv3).	☐	☐
The collected data is available under a Creative Commons License (CC BY-SA 4.0).	☐	☐

. . . .

(Place, Date) (Signature)

Zusammenfassung

Die Elizitierung ist ein neuer Ansatz zur Rekonstruktion von Argumentationsframeworks. Sie ist durch einen interaktiven Prozess gekennzeichnet, in dem ein Agent ein Argumentationsframework besitzt und Fragen hierzu beantworten kann. Das Ergebnis des Prozesses ist die Rekonstruktion eines Argumentationsframeworks, das dem des Agenten äquivalent ist. In dieser Masterarbeit wird die Elizitierung von Argumentationsframeworks unter Beschränkung auf semantische Entscheidungsfragen mit Bezug zu den klassischen Semantiken untersucht. Die daraus rekonstruierten Argumentationsframeworks besitzen unter der gegebenen Semantik die gleichen Extensionen wie jene des Agenten, d.h. sie sind σ-äquivalent. Dazu werden zunächst die Eigenschaften der klassischen Semantiken untersucht, die sich für Algorithmen im interaktiven Teil eignen. Anschließend werden die semantischen Entscheidungsfragen spezifiziert, die an den Agenten gestellt werden können. Diese orientieren sich an den bekannten Berechnungsproblemen der abstrakten Argumentation. Aus den geeigneten Eigenschaften und Fragen werden Algorithmen für den Frageteil des Elizitierungsprozesses entwickelt und anschließend hinsichtlich ihrer Leistungsfähigkeit evaluiert. Es zeigt sich, dass ohne vorherige Enumerierung für viele Semantiken außer der begründeten und der vollständigen Semantik ein naiver Ansatz mittels Durchfragen aller Extensionen besser abschneidet. Darüber hinaus wird diskutiert, inwiefern eine unmittelbare argumentkongruente Rekonstruktion eines σ-äquivalenten Argumentationsframeworks aus den gesammelten Informationen erfolgen kann. Dies ist nur bei konfliktfreien Mengen sowie vollständiger Semantik zuverlässig möglich.

Abstract

Elicitation is a novel approach to the reconstruction of argumentation frameworks. It is characterised by an interactive process in which an agent possesses an argumentation framework and is able to answer questions about it. The result of this process is the reconstruction of an argumentation framework equivalent to that of the agent. In this Master's thesis, the elicitation of argumentation frameworks is investigated under the restriction of using semantic decision questions with reference to the classical semantics. The argumentation frameworks reconstructed therefrom have the same extensions as these of the agent under the given semantics, i.e. they are σ-equivalent. For this purpose, properties of the classical semantics that are suitable for algorithms in the interactive part are first examined. Then, the semantic decision questions that can be posed to the agent are specified. These are oriented towards the well-known computational tasks of abstract argumentation. Algorithms for the question part of the elicitation process are developed from the suitable properties and questions and then evaluated with regard to their performance. It is shown that without prior enumeration, for many semantics except grounded and complete semantics, a naïve approach by means of asking for all extensions performs better. Furthermore, it is discussed to what extent an immediate argument-congruent reconstruction of a σ-equivalent argumentation framework can be done from the collected information. This is only reliably feasible for conflict-free sets as well as the complete semantics.

Contents

1. Introduction

The subject area of formal argumentation, as a subfield of Artificial Intelligence (AI), deals with approaches to the formal modelling of human argumentation and the use of these models to infer knowledge. The idea is to recreate human knowledge reasoning this way. In 1995 Phan Minh Dung founded the approach of abstract argumentation with his to this day influencing seminal paper [20]. There he introduced an abstract representation of arguments and their attack relation. For this purpose, the arguments are abstracted in such a way that their internal structure is no longer relevant for further considerations. A lot of work has been put into drawing meaningful conclusions from so-called (abstract) argumentation frameworks (AF). These AFs consist of directed graphs where the nodes represent the arguments and the edges define the attack relation. The meaning behind an attack of an argument A against an argument B is that there is a contradiction. So if an argument A is accepted, then B must be rejected. To determine the set of accepted arguments, a calculus is applied. What follows is a brief illustration using an example based on an argument about plant watering.

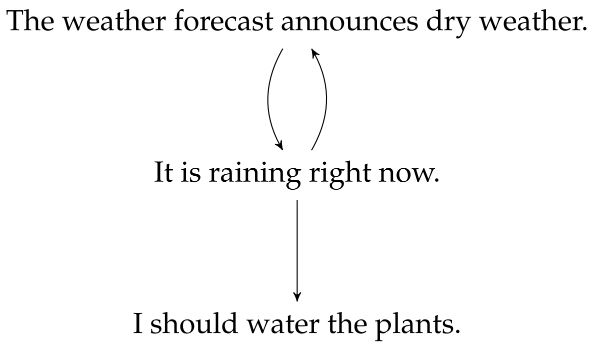
The weather forecast announces dry weather.
It is raining right now.
I should water the plants.

Figure 1: An illustration of an argument about plant watering.

Example 1. *Consider Figure 1 with its reasoning and the three statements. A decision is to be made on whether to water the plants. This result lies in the acceptance of the argument* I should water the plants. *The argument itself is attacked by the statement that* It is raining right now. *This is indicated by the arrow. So if this observation is true and the argument* It is raining right now *is accepted, then* I should water the plants *is rejected and therefore the plants do not need to be gardened. The reverse is also possible.*

However, the argument It is raining right now *is in mutual conflict with the third statement* The weather forecast announces dry weather *that has not yet been considered further. So a decision must be made beforehand between the two conflicting arguments. Only then a decision can be concluded.*

So if one of the two arguments is accepted, the other is rejected. Depending on the observation made regarding the weather, it can be reasoned whether the plants need to be watered or not.

As mentioned before, the actual contents of the arguments are abstracted away. Therefore, an abstract framework results in a structure as in Figure 2, where the arguments are identified by variable names. Without going into more detail at this point, such an AF is described by a pair of the set of arguments and the set of attacks, in formal notation (Arg, AR)*. So in this case* $F_1 = \big(\{A_1, A_2, A_3\}, \{(A_1, A_2), (A_2, A_1), (A_2, A_3)\}\big)$.

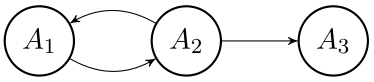

Figure 2: The abstract argumentation framework modelled in accordance with Example 1.

Although the basic idea of abstract frameworks is very simple, this approach can be used for different scenarios. For example, as in recent publications on recommendation systems [42] or persuasion dialogues [15]. As part of active research, Dung's approach was enriched in a wide range for example by adding concepts as preferences and strength [12] or by developing weighted argumentation systems [24]. New research findings continue to be published.

One such object of study that plays an important role is the calculus. It determines consistent or accepted arguments based on the attack relations and is called a semantics. The set of arguments calculated by this function is called an extension. Several semantics are already named in Dung's work, which are also commonly referred to as the classical semantics. These are called complete, grounded, preferred and stable semantics. Over the years, more semantics have been developed that represent different ideas. Nevertheless, all semantics pursue the goal of finding the accepted arguments. Conversely, it can be assumed that there is a direct or indirect conflict between the arguments that are not part of the outcome. This may indicate that these arguments could be contradicted by the extensions via a direct attack of the arguments in the extensions or that they could be rejected due to other conflicts not directly connected to the acceptable arguments. The latter explanation may indicate that there is a possibility that these arguments are compatible with the accepted arguments, but just not in the given extension.

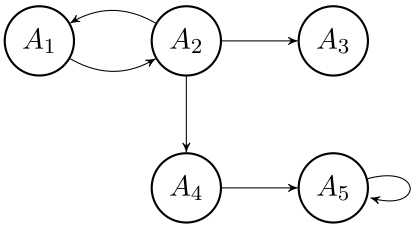

Figure 3: An example of an abstract argumentation framework with five arguments.

Example 2. *To give an example of semantics and their extensions, assume an AF like the one in Figure 3. This AF has five arguments with several attacks between themselves. The two arguments A_1 and A_2 attack each other. Also argument A_5 attacks itself and is attacked by A_4 as well. Furthermore, A_2 attacks A_3. Without going into more detail about the characterisation of semantics at this point, we choose so-called conflict-free sets as the semantics. We therefore consider a semantics in general as a function that provides us with extensions to an argumentation framework, i.e. sets of arguments that fulfil a certain condition. For conflict-free sets, the condition is that if there is no attack between two arguments, they are accepted as extensions.*

This condition is always true for the empty set $\emptyset$, i.e. if no argument is chosen, then there is no conflict. Further, all arguments except A_5 are not self-attacking, so each argument forms an extension by itself, i.e. $\{\{A_1\}, \{A_2\}, \{A_3\}, \{A_4\}\}$. Moreover, there are no attacks between A_1, A_4, A_3, so extensions can also be formed from them $\{A_1, A_3\}$, $\{A_1, A_4\}$, $\{A_3, A_4\}$ as well as all three arguments together $\{A_1, A_3, A_4\}$.

With these introductory thoughts on the broader research field in which this thesis is situated, we next touch on the research topic of elicitation and introduce the motivation and problem. Afterwards, we formulate the central research question and present the structure of the thesis.

1.1. Motivation and Problem Specification

The previously given context to abstract argumentation provides a basis for calculating extensions, i.e. the set of arguments accepted together. If one assumes that an agent possesses such a model and knows about the extensions, it is also conceivable that through previously known information and, in particular, by questioning the agent, one gathers further information in order to construct or reconstruct an abstract argumentation framework that completely or to some extent replicates the agent's model such that the same extensions arise.

Kuhlmann [34] gave an introductory elaboration of the problem and named it the elicitation of abstract argumentation frameworks. Thereby, the ability to choose questions to ask for needed information in the process can be considered unique to this approach [34]. So far, no further work based on this is known, with the exception of Kuhlmann et al.'s [35] paper on agents that hold several semantics as so-called types and the subsequent questions of which semantics can be distinguished on the basis of that type, taking into account some of the characteristics of AFs.

Let us demonstrate the usefulness of this approach with the following train of thought: We assume, for example, that an agent reads a newspaper article with arguments for and against a matter, for instance living in a suburb, and now, based on information such as the amount of accepted arguments or the like, which the agent provides, the agent's hidden AF is to be found out.

Now we take the position of an interviewer who also read the article and therefore at least knows all the arguments as well. Now we want to elicit information from the agent by asking a series of questions, in order to then produce an AF from compiled information. Two successive steps can therefore be identified. First, we conduct an interview and then we create an AF, which is a reconstruction of the agent's AF.

So far, this research topic has received little attention. Initial work has already been done on the approach of reconstructing an AF with information already given. Niskanen et al. [37] and Riveret et al. [43] have already dealt with reconstructing an AF on the basis of extensions respectively labels (a method of encoding further information beyond acceptance by means of multiple values).

These approaches correspond to the activities to be done in the second step. If, however, the information for a reconstruction is not yet available and must be collected, which corresponds to the first step, then no work exists apart from the introduction by Kuhlmann [34] described above.

Altogether, elicitation is a process based on the interaction between two parties, an agent with an AF and an interviewer who may have partial information and may ask questions. The fundamental problem we address in this Master's thesis is to find an elicitation process in which all arguments and a semantics are revealed to the interviewer and the interviewer now poses questions to the agent, who answers them truthfully, as it would be the case with an oracle. In the process, the interviewer uses the answers given to decide which further questions to ask. Once he has gathered all the information, the interviewer constructs an AF that corresponds to the desired notion of equivalence.

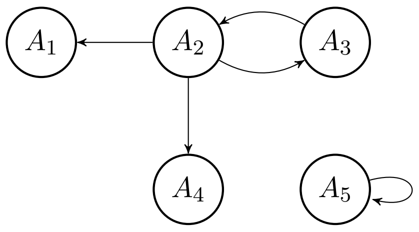

Figure 4: An example of a reconstructed argumentation framework with five arguments.

For this purpose, it must be understood that a set of extensions in many cases can be realised by more than one argumentation framework. We revisit Example 2 for this purpose. Here, nine extensions $\{\{\}, \{A_1\}, \{A_2\}, \{A_3\}, \{A_4\}, \{A_1, A_3\}, \{A_1, A_4\}, \{A_3, A_4\}, \{A_1, A_3, A_4\}\}$ were identified as conflict-free sets. The interviewer knows all the arguments $\{A_1, A_2, A_3, A_4, A_5\}$ and also comes up with these nine extensions through clever questioning. Figure 3 represents the AF of the agent. Now, if our goal as the opposing party is to construct an AF that produces the same extensions, we can obtain an AF like the one in Figure 4 by guessing the attack relations. To do this, we just need to preserve the conflicts between the extensions so that they arise. For example, A_5 is not in any extension and so can be removed from all extensions by only attacking itself. As a result, it is obvious that the attack relations are different, but nevertheless, the same extensions with respect to the conflict-free sets are produced.

This also demonstrates a problem of reconstruction: Reconstructing a syntactically identical AF requires a lot of information beyond the knowledge of the extensions. Therefore, standard equivalence is only described by the semantics σ. Two AFs are considered equivalent if they produce the same extensions under the same semantics. It is also called σ-equivalence in the following. This concept guarantees that the desired result, i.e. the accepted arguments, is achieved by both compared argumentation frameworks. If however new information, that is attacks and new arguments, is added to the two AFs, then the two may no longer be equivalent under σ-equivalence.

As a broader restriction, if σ-equivalence is used as a condition for the generated AFs to be accepted, then there is no guarantee that only the syntactically identical AF will be found. That is, there is a set of AFs that are equivalent in terms of extensions under the semantics and thus can be called a σ-equivalent class. Then the agent's AF is a member of it, but if there is more than one AF, then it is unclear which of these syntactically corresponds to that of the agent's AF.

In this Master's thesis, we intend to fundamentally develop algorithms for the first interview part that go beyond Kuhlmann's [34] naïve approach. Since there are already approaches for σ-equivalent reconstructions, the second step is based on these. The implication of this for the first step is that only the extensions need to be elicited. Therefore, we set the restriction that only questions with relation to the extensions are asked.

However, the approaches of this thesis are chosen in such a way that further work based on this body of work can examine the algorithms under a stricter variant of equivalence, such as strong equivalence described by Oikarinen and Woltran [38].

1.2. Objective and Research Question

What follows is the presentation of the overall objective of the Master's thesis, which aspect of the research area of elicitation will be dealt with and the research question derived from it. In addition, the further structure of this thesis is presented.

The objective is, that the Master's thesis should contribute to the overall goal of developing algorithms for the elicitation of abstract argumentation frameworks. The algorithms to be developed contribute to the subfield of eliciting σ-equivalent AFs by asking only extension-related questions.

From this objective we derive the following research question:

Research Question. *What are meaningful properties of the semantics and the possible extensions of an AF in respect of elicitation and how can they be formed into algorithms only using semantic questions for eliciting sigma-equivalent AFs under admissible and conflict-*

free extensions[1] *as well as the complete, grounded, preferred and stable semantics?*

To check whether the properties found actually lead to better algorithms, we will benchmark the algorithms against the naïve approach in an experiment. One of the performance indicators here is the CPU runtime of the different algorithms developed compared to the naïve approach (see Section 4.3.1). For a valuable algorithm the runtime should be lower in favour of the algorithms developed.

In order to develop such algorithms that subsequently allow for the reconstruction of an AF by asking questions first, some restrictions or assumptions have to be made to sufficiently narrow down the scope of the problem. As mentioned before, the notion of σ-equivalence is used to determine equivalency between the found AF and the hidden AF of the agent. In addition, a reasonable assumption must be made about the agent's ability to answer questions. Therefore the kind of problems the agent can solve and subsequently the set of questions must be specified. The exact set will be defined later in this thesis, but, already anticipating later analyses, the restriction is that only decision questions with a relation to extensions are answered by the agent. These are polar questions that can only be answered truthfully with a yes or no.

So in order to answer the research question, a literature review is conducted first. This is done in order to get an overview of the problem of elicitation and related approaches such as learning AFs. Next, it is investigated which questions could usefully be asked to an agent in such a scenario. With this question pool in hand, algorithms for the classical semantics are developed to elicit σ-equivalent AFs. In addition, we give an insight into the possibilities of reconstruction from the collected information. The algorithms are then implemented and evaluations are carried out on the basis of this implementation to examine the performance of the approach.

The contributions to the subject area of elicitation of this Master's thesis are:

1. Finding meaningful boundaries regarding the agent's ability to answer questions, and building on this, establishing a question pool that the interviewer can use to elicit the agent's AF.

2. The inspection and identification of properties of classical semantics, that are useful for narrowing the search space.

3. The development of algorithms that couples the found properties with the possible questions.

4. The implementation of the algorithms and the evaluation of their performance.

The Master's thesis is therefore organised as follows: Section 2 gives the necessary background for abstract argumentation frameworks and their semantics. In

[1]For the sake of simplification, admissible and conflict-free extensions from now on will be seen as semantics, too.

addition, the approach of labelling instead of forming extensions is briefly introduced. In order to assess the computability of the questions to be developed, the field of computational complexity is briefly introduced. Furthermore, the problem of elicitation is revisited and the naïve approach is explained.

Following on from this, Section 3 looks at the approaches that have been taken to address the reconstruction of AFs so far.

With this background knowledge, the algorithms are developed and described in Section 4. This is the main contribution of this Master's thesis. First, the algorithmic process of elicitation is described, and then the questions for the agent are defined. After that, the algorithms for the classical semantics are developed. Finally, the reconstruction from the collected information is discussed and what limitations it has.

The proposed algorithms were implemented on this basis. These were then evaluated and in Section 5 the results are discussed in terms of their performance. Finally, Section 6 provides a summary of the topic and identifies further directions for future work.

2. Background

This section is intended to provide a basis for further elaboration on the topic. For this purpose, literature important for further understanding is cited and an introduction to those approaches is given. This background section begins with the concept of abstract argumentation frameworks [20] followed by a description of the idea of argumentation semantics [20] and its realisation in the form of extensions [20]. In order to be able to understand similar approaches to the reconstruction of argumentation frameworks, the alternative approach of labellings [14] is also briefly described. Before turning to elicitation, the topic of computational complexities [1] will be touched upon. This will be necessary later in order to evaluate different types of elicitation questions in terms of their tractability. This is finally followed by a more formal description of the approach of eliciting argumentation frameworks [34] than what was described in the introductory section.

2.1. Abstract Argumentation Frameworks

First of all, the essential concept of Dung's (abstract) argumentation frameworks [20], which is fundamental to this Master's thesis, is outlined. Therefore, the following definitions are taken from or based on Dung's work [20].

An argumentation framework consists of two components. One is the set of arguments Arg and the other is the attack relation, a binary relation AR over Arg. The word abstract in abstract argumentation frameworks comes from the fact that the inner structure of the argument is not important for the reasoning. It is based purely on the relations between the arguments.

Definition 1. *An abstract argumentation framework (AF) is a pair $\langle Arg, AR \rangle$, where Arg is a set of arguments and $AR \subseteq Arg \times Arg$ is a binary relation.*

In the context of this Master's thesis, it is assumed that all argumentation frameworks are finite.

Definition 2. *Given an AF $F = (Arg, AR)$, it is that $Arg \subseteq \mathcal{U}$ is finite. Further $\mathbb{F}$ denotes the set of all argumentation frameworks over $\mathcal{U}$.*

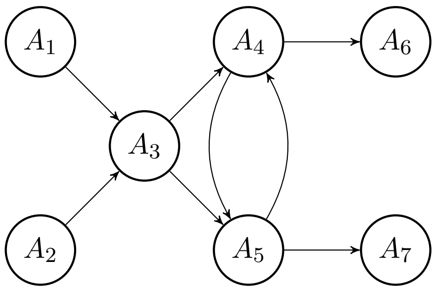

Figure 5: An example argumentation framework F_2.

To represent that an argument $a \in Arg$ attacks an argument $b \in Arg$ the notation $a \hookrightarrow b$ is used; so that a attacks an argument b if $a \hookrightarrow b$ holds. Moreover, this notion can also be extended to sets, so that a set $S \subseteq Arg$ of arguments attacks an argument b if b is attacked by an argument in $a \in S$. Conversely, an argument a can also attack a set if an argument $b \in S$ from the set is attacked by the argument a.

Another important notion is defence. A set E defends an argument a if every attacker of a is attacked by an argument from E.

However, sets of arguments cannot be considered only in terms of their attacks on any other arbitrary argument and vice versa. An important property is conflict-freeness, which means that the arguments in a set do not attack each other.

Definition 3. *A set S of arguments is conflict-free if no argument in S attacks an argument in S.*

Example 3. *Consider the argumentation framework in Figure 5. Several conflict-free sets can be determined. An example is $\{A_1, A_2\}$. The two arguments A_1 and A_2 of this set do not attack each other and are hence conflict-free.*

The set $\{A_1, A_2, A_3\}$ can be taken as a counterexample. It is not conflict-free because A_3 is attacked by both A_1 and A_2.

Now, when choosing a set of arguments of an argumentation framework, the question arises of how to determine their acceptance. A rational agent will only accept a set of arguments if it can defend itself against all attacks (see Definition 4). This set can then be said to be admissible (see Definition 5).

Definition 4. *Let $F = (Arg, AR)$ be an argumentation framework. Then an argument $a \in Arg$ is* acceptable with the respect to *$S \subseteq Arg$ iff for each argument $b \in Arg$: if $b \hookrightarrow a$ then there is a $c \in S$ such that $c \hookrightarrow b$.*

Definition 5. *An set of arguments S is* admissible *if S is conflict-free and each argument in S is acceptable with respect to S.*

Example 4. *Consider Figure 5 a second time. In this example, a set is looked for that covers Definition 4 and Definition 5 on acceptability and admissibility. The set $\{A_1, A_2\}$ from Example 3 satisfies these, although this is a rather straightforward example. Another set is, for instance, $\{A_1, A_2, A_4, A_7\}$. The set is conflict-free because none of the arguments attack each other. Moreover, arguments A_1 and A_2 attack argument A3 (in fact only one of the two is needed) and so argument A_4 is accepted by the defence. Since A_5 is attacked by A_4, argument A_7 is likewise defended and is therefore accepted. Thus all arguments of the set are accepted in respect of the set and the set is therefore not only conflict-free but also admissible.*

Finally, further notions for extensions and the characteristic function shall be described. E^+ denotes a set containing all attacked arguments of E, i.e. $E^+ = \{a \in Arg \mid E \text{ attacks } a\}$. Moreover, the range is a set $E^\oplus$ containing the extension and all arguments attacked by the extension, i.e. $E^\oplus = E \cup E^+$. Finally, F^E is called an E-reduct of an AF F such that $F^E = (E^*, AR \cap (E^* \times E^*))$ with $E^* = Arg \setminus E^\oplus$.

And lastly, the characteristic function Γ, which is defined as follows:

Definition 6. *Let $F = (Arg, AR)$ be an argumentation framework. Then $\Gamma_F(E)$ is a function that returns all arguments that are defended by the extension E, i.e. $\Gamma_F(E) = \{a \in Arg \mid E \text{ defends } a\}$.*

2.2. Extension-based Semantics

A semantics is a calculation method for evaluating the arguments. This is necessary because some arguments have to be rejected based on the attack relation. It must therefore be determined whether an argument is justified or not. The different semantics have their nuances and therefore each provides a particular evaluation methodology that describes under which attacks an argument is still justified or not.

Already in the fundamental paper by Dung [20], the complete, grounded, preferred and stable semantics are described. They are therefore also called classical semantics.

The function for evaluating a semantics is generally described by the extension-based or the labelling-based approach. In the following, the extension-based approach will be examined in closer detail to begin with. In the succeeding subsection, the labelling approach is briefly described.

An extension is a subset of Arg in an argumentation framework (Arg, AR) that includes the arguments that can be accepted together. A semantics in this case then defines a set of extensions that can be derived from an argumentation framework according to the methodology of semantics. The formal account is given in Definition 7 and is based on the definition for semantics from [49].

Definition 7. *An extension-based* semantics *is a function σ such that for every argumentation framework $F = (Arg, AR)$, $\sigma(F) \in 2^{2^{Arg}}$ with 2^S as the notation for the power set of S. The elements of $\sigma(F)$ are called* extensions*.*

Although the two concepts of conflict-freeness and admissibility are not semantics in the ordinary sense, they will be considered among the classical semantics for the sake of simplicity for the rest of the Master's thesis. To complete the picture, the remaining, ordinary classical semantics will therefore be described next.

All the subsequent semantics are based on the idea of admissibility. A first stricter selection criterion for justified arguments is provided by the complete semantics. It restricts the derived extensions so that each extension only contains exactly those arguments that it also defends.

Definition 8. *Let $F = (Arg, AR)$ be an AF. A set of arguments $E \subseteq Arg$ is a* complete extension *if E is admissible and each argument $a \in Arg$ that is acceptable w.r.t. E belongs to E, so that $a \in E$.*

Example 5. *Consider the argumentation framework $F_2 = (Arg, AR)$ in Figure 5. To find the complete extensions, extensions that are admissible can be formed and examined to see if all arguments are also contained in the extension it defends. Therefore, the empty set $\emptyset$, which is always admissible, should be checked first. But since there are arguments that are not attacked by any other, namely A_1 and A_2, and are thus always defended by conflict-free sets, the empty set is not a complete extension. However, the set $\{A_1, A_2\}$ is complete by having both arguments. So the remaining extensions always have $\{A_1, A_2\}$ as a subset. Since A_3 is always attacked if A_1 or A_2 occur in a set, this argument for set formation is dropped. It is not part of any admissible set. The attack relation of A_4 and A_5 represents a decision. If one of the two arguments is selected in either case, A_7 or A_6 is defended respectively. Thus the complete extensions are $\{A_1, A_2\}$, $\{A_1, A_2, A_4, A_7\}$ and $\{A_1, A_2, A_5, A_6\}$.*

Based on the complete semantics, two further semantics can be defined, which in turn restrict the choice of arguments based on the attack relation. The two semantics are the grounded and preferred semantics. First, the grounded semantics is defined. It is characterised by containing that complete extension which is minimal with respect to set inclusion. Since there is always exactly one minimal extension, this means that the grounded extension is always unique. Informally speaking, it therefore contains those arguments that are not attacked by any other argument.

Definition 9. *Let $F = (Arg, AR)$ be an AF. A set of arguments $E \subseteq Arg$ is a* grounded extension *if and only if E is admissible and E is the minimal complete extension with respect to set inclusion.*

Example 6. *Consider once again the argumentation framework $F_2 = (Arg, AR)$ in Figure 5. As already shown in Example 5, the complete extensions are $\{A_1, A_2\}$, $\{A_1, A_2, A_4, A_7\}$*

and $\{A_1, A_2, A_5, A_6\}$. The minimal extension in terms of set inclusion is, therefore, $\{A_1, A_2\}$. It is easy to see that these are the arguments that are not attacked by anyone else.

Now be the preferred semantics defined. The preferred extensions are defined as those admissible extensions that are maximal with respect to set inclusion.

Definition 10. *Let $F = (Arg, AR)$ be an AF. A set of arguments $E \subseteq Arg$ is a preferred extension if and only if E is admissible and E is a maximal admissible set with respect to set inclusion.*

Example 7. *As in the previous example, consider the argumentation framework $F_2 = (Arg, AR)$ in Figure 5. As a reminder, the complete extensions are $\{A_1, A_2\}$, $\{A_1, A_2, A_4, A_7\}$ and $\{A_1, A_2, A_5, A_6\}$. Therefore, $\{A_1, A_2, A_4, A_7\}$ and $\{A_1, A_2, A_5, A_6\}$ are the preferred extensions in F, since there is no complete superset of both of them.*

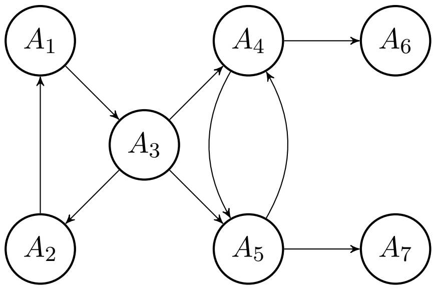

Figure 6: An modification of the argumentation framework in Figure 5.

The last remaining classical semantics is the stable semantics. It also further restricts the complete one. I.e. a stable extension is a complete extension, but it also attacks every other argument of the argumentation framework outside of the extension. As the only semantics of the described ones, it can be that there is no stable extension at all. This is known to occur when there are odd attack cycles.

Definition 11. *Let $F = (Arg, AR)$ be an AF. A set of arguments $E \subseteq Arg$ is a stable extension, if and only if E is complete and for every arguments $a \in Arg \setminus E$, $E \hookrightarrow a$.*

Example 8. *Consider the argumentation framework $F_2 = (Arg, AR)$ in Figure 5. As a reminder, the complete extensions are $\{A_1, A_2\}$, $\{A_1, A_2, A_4, A_7\}$ and $\{A_1, A_2, A_5, A_6\}$. Since extensions $\{A_1, A_2, A_4, A_7\}$ and $\{A_1, A_2, A_5, A_6\}$ attack all arguments outside themselves, they are also stable. A modification F_3 of F_2 can be seen in Figure 6, which shows an odd attack cycle between arguments A_1, A_3 and A_2. In this case, there is no stable extension because the only complete extension is the empty set $\emptyset$ and this obviously does not attack all other arguments.*

The semantics are generally expressed by the function $\sigma(F)$ for any argumentation framework F. The actual semantics, on the other hand, are expressed by replacing σ with the corresponding function name of the semantics.

Definition 12. *The classical semantics are denoted by $\sigma \in \{cf, ad, co, gr, pr, st\}$ where*

- cf *is the set of conflict-free sets;*
- ad *is the set of admissible sets;*
- co *is the set of complete extensions;*
- gr *is the set containing the grounded extension;*
- pr *is the set of preferred extensions;*
- st *is the set of stable extensions.*

Finally, the relationships between the semantics should be discussed. The semantics are based on the conflict-free nature of the sets. Based on this, these sets must also be admissible, i.e. every admissible set is also conflict-free. Since the complete semantics is a stronger concept than the admissible semantics, every complete extension is also an admissible one. With regard to grounded semantics, it must be said that, by definition, every grounded extension is also a complete extension. The same can be said of preferred semantics. An interesting case is the stable semantics. Every stable extension is also a complete extension, but moreover, every stable extension is also a preferred one. Figure 7 illustrates this once again graphically.

$$\begin{tikzcd}
 & & \text{stable semantics} \arrow[d, "\text{is a}"] \\
\text{grounded semantics} \arrow[dr, "\text{is a}"] & & \text{preferred semantics} \arrow[dl, "\text{is a}"'] \\
 & \text{complete semantics} \arrow[d, "\text{is a}"] & \\
 & \text{admissible semantics} \arrow[d, "\text{is a}"] & \\
 & \text{conflict-free semantics} &
\end{tikzcd}$$

Figure 7: The relationships between the classical semantics.

2.3. Labelling-based Semantics

This Master's thesis focuses on the extension-based approach. However, there is another commonly used approach in the literature that makes use of argument labelling. Since related work uses labellings, besides extensions, to reconstruct abstract argumentation frameworks, these will be briefly introduced. Caminada and Gabbay [14] gave a good overview of formal definitions for the classical semantics using the labelling concept in their article. The approach was already described in 1999 by Jakobovits and Vermeir [31] for Dung's argumentation frameworks and is based on an earlier work by Pollock [41]. Extensions include those accepted arguments that successfully argue against contesting arguments. Furthermore, they implicitly convey the information that all arguments that are not in an extension have been rejected. Not only can this also be expressed through argument labelling, but additional information about the contested arguments becomes clear. Thus, explicitly rejected arguments can be identified, but also those that are withheld from explicit assessment.

The following definitions are taken from or based on [14].

Definition 13. *Let $\Lambda = \{\mathsf{in}, \mathsf{out}, \mathsf{undec}\}$ be the labels and $F = (Arg, AR)$ be an AF. A labelling is a total function $\mathcal{L} : Arg \to \Lambda$.*

There are three labels associated to the argument labelling. The labelling-based semantics is a function producing a set of labellings.

The next aim is to illustrate the complete labelling.

Definition 14. *Let $F = (Arg, AR)$ be an AF and $\mathcal{L}$ be a labelling on F. $\mathcal{L}$ is a complete labelling if and only if for every $a \in Arg$:*

- *if a is labelled* in *then all its attackers are labelled* out*;*
- *if a is labelled* out *then one of its attackers is labelled* in*;*
- *if a is labelled* undec *then not all its attackers are labelled* out *and none of its attackers is labelled* in.

in($\mathcal{L}$), out($\mathcal{L}$) *and* undec($\mathcal{L}$) *denote the set of arguments labelled* in, out, undec *respectively.*

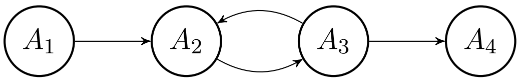

Figure 8: An argumentation framework with four arguments.

Example 9. *Consider the argumentation framework $F_4 = (Arg, AR)$ in Figure 8. For F, there is only one complete labelling based on the criteria. Starting with argument A_1, this is always to be labelled as* in*. Subsequently, A_2 is to be labelled* out*. Since A_2 is* out*, the mutual conflict with A_3 is resolved and A_3 is also* in *with the same consequence as for A_1 and a_2 that A_4 is* out*. None of the arguments is* undec*. That is,* in$(\mathcal{L}_1) = \{A_1, A_3\}$ *and* out$(\mathcal{L}_1) = \{A_2, A_4\}$ *for the one complete labelling $\mathcal{L}_1$.*

A parallel is evident, for every AF F the set of complete extensions is exactly the set of arguments of the complete labelling which are in, i.e., in set builder notation $\{\mathsf{in}(\mathcal{L}) | \mathcal{L} \text{ is a complete labelling}\}$.

The other classical semantics can furthermore be described by restrictions for the complete labelling. For this purpose, Caminada and Gabbay [14] have established several intuitive definitions and theorems that can be summarised as in the following definition. The reader is referred to their work for detailed descriptions.

Definition 15. *Let $F = (Arg, AR)$ be an AF and $\mathcal{L}$ be a labelling on F.*
$\mathcal{L}$ is a

- *grounded labelling if $\mathcal{L}$ is a complete labelling where either*
  - $\mathsf{in}(\mathcal{L})$ *is minimal with respect to set inclusion,*
  - $\mathsf{out}(\mathcal{L})$ *is minimal with respect to set inclusion or*
  - $\mathsf{undec}(\mathcal{L})$ *is maximal with respect to set inclusion.*
- *preferred labelling if $\mathcal{L}$ is a complete labelling where either*
  - $\mathsf{in}(\mathcal{L})$ *is maximal with respect to set inclusion or*
  - $\mathsf{out}(\mathcal{L})$ *is maximal with respect to set inclusion.*
- *stable labelling if $\mathcal{L}$ is a complete labelling such that* $\mathsf{undec}(\mathcal{L}) = \emptyset$.

2.4. Signatures and Realisability

One perspective on semantics and the abstract argumentation frameworks is the concept of realisability and their signatures. Given a set $\mathbb{S}$, it is checked whether it is realisable under a semantics σ, i.e. it is determined whether there is a set of σ-extensions that match $\mathbb{S}$. The concept of signatures is the collection of all realisable sets.

A lot of basic work on this can be found in the research paper by Dunne et al. [23], which is why many of the following definitions and examples are based on it. For proof of the statements, the reader is referred to the referenced articles.

2.4.1. Signatures of Extension-Based Semantics

First of all, the realisable sets are to be characterised and their properties described. For this purpose, a formal Definition 16 of signatures is to be given first.

Definition 16. *The signature Σ_σ of a semantics σ is defined as $\Sigma_\sigma = \{\sigma(F) \mid F \in \mathbb{F}\}$.*

Next are some definitions of notation used to characterise the signatures of semantics in this subsection.

Definition 17. *Given $\mathbb{S} \subseteq 2^{\mathcal{U}}$,*

- *$Args_{\mathbb{S}}$ denotes $\bigcup_{S \in \mathbb{S}} S$ and $||\mathbb{S}||$ for $|Args_{\mathbb{S}}|$,*
- *$Pairs_{\mathbb{S}}$ denotes $\{(a,b) \mid \exists S \in \mathbb{S} : \{a,b\} \subseteq S\}$ and*
- *$dcl(\mathbb{S})$ denotes $\{S' \subseteq S \mid S \in \mathbb{S}\}$, also referred to as downward-closure.*

The set $\mathbb{S}$ is also called an extension-set if $Args_{\mathbb{S}}$ is finite, i.e. in the following this term is used as only finite AFs are considered in this thesis. Therefore, it should be noted that for all semantics σ considered, each element $\mathbb{S} \in \Sigma_\sigma$ is an extension-set.

It should also be taken into consideration that for any $a \in Args_{\mathbb{S}}$, $(a,a) \in Pairs_{\mathbb{S}}$ holds for all extension-sets $\mathbb{S}$.

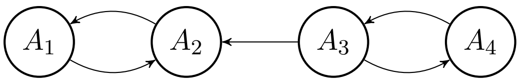

Figure 9: An argumentation framework for Example 10.

Example 10. *To familiarize the reader with Definition 17, suppose the extension-set $\mathbb{S} = \{\{A_1,A_3\},\{A_1,A_4\},\{A_2,A_4\}\}$. The terms can be applied as follows:*

- *$Args_{\mathbb{S}} = \{A_1,A_2,A_3,A_4\}$ and $||\mathbb{S}|| = 4$,*
- *$Pairs_{\mathbb{S}} = \{(A_1,A_1),(A_2,A_2),(A_3,A_3),(A_4,A_4),(A_1,A_3),(A_1,A_4),(A_2,A_4),$ $(A_3,A_1),(A_4,A_1),(A_4,A_2)\}$ and*
- *$dcl(\mathbb{S}) = \{\emptyset,\{A_1\},\{A_2\},\{A_3\},\{A_4\},\{A_1,A_3\},\{A_1,A_4\},\{A_2,A_4\}\}$.*

The AF in Figure 9 produces under the complete semantics the extension-set $\mathbb{S}$ and therefore $\mathbb{S} \in \Sigma_{co}$.

As in the subsection on semantics, conflict-free sets are considered first. A property of these sets is that their subsets are also all conflict-free. This corresponds to the fact that these sets are downward-closed, i.e. that for a given AF the downward-closure does not affect the set of conflict-free sets.

Moreover, it is obvious, too, that for semantics based on admissibility, such as the stable or preferred semantics, $\sigma(F)$ is incomparable for any AF.

Definition 18. *Given $\mathbb{S} \subseteq 2^{\mathcal{U}}$, $\mathbb{S}$ is*

- *downward-closed if $\mathbb{S} = dcl(\mathbb{S})$ and*
- *incomparable if all elements $S \in \mathbb{S}$ are pairwise incomparable, i.e. for each $S, S' \in \mathbb{S}$, $S \subseteq S'$ implies $S = S'$.*

Nevertheless, incomparability is not sufficient as a property, as Dunne et al. [23] show in the following example:

Example 11. *Consider the incomparable extension-set $\mathbb{S} = \{\{A_1, A_2\}, \{A_1, A_3\}, \{A_2, A_3\}\}$ and a semantics σ that preserves conflict-freeness, i.e. $\sigma(F) \subseteq cf(F)$ for any AF F. Now suppose there exists an AF F with $\sigma(F) = \mathbb{S}$. Then F must not contain attacks between A_1 and A_2, A_1 and A_3, and respectively A_2 and A_3. But then $\sigma(F)$ typically contains $\{A_1, A_2, A_3\}$.*

In order to exclude sets like $\mathbb{S}$ from the example, different approaches can be discussed. For the stable semantics, as well as for the stage and naive semantics not considered in the thesis, a strong condition can be made to characterise them. This is called tightness.

Informally, it can be said that if an argument does not occur in an extension, then there are reasons for this, such as a direct conflict. So for incomparable sets, if there is a set $S \in \mathbb{S}$ and an argument a that does not occur in S, so that for each $s \in S$ there is another set S' with a and s as members, then the incomparable set is not tight. As [23] points out, for incomparable sets $\mathbb{S}$ the premise of the condition $S \cup \{a\} \notin \mathbb{S}$ is always fulfilled. Therefore, it is only necessary to check whether for all $S \in \mathbb{S}$ and $a \in Args_{\mathbb{S}} \setminus S$ there is an $s \in S$ such that $(a, s) \notin Pairs_{\mathbb{S}}$.

Definition 19. *An extension-set $\mathbb{S} \subseteq 2^{\mathcal{U}}$ is tight if for all $S \in \mathbb{S}$ and $a \in Args_{\mathbb{S}}$ it holds that if $S \cup \{a\} \notin \mathbb{S}$ then there exists an $s \in S$ such that $(a, s) \notin Pairs_{\mathbb{S}}$.*

Example 12. *Consider once again the extension set $\mathbb{S} = \{\{A_1, A_2\}, \{A_1, A_3\}, \{A_2, A_3\}\}$ from Example 11. This extension-set is incomparable but not tight, since there is no reason to, for instance, exclude A_3 from the extension $\{A_1, A_2\}$. This is because (A_1, A_3) and (A_2, A_3) are both contained in $Pairs_{\mathbb{S}}$. On the other hand, the extension-set $\mathbb{S}' = \{\{A_1, A_2\}, \{A_1, A_3\}, \{A_2, A_4\}, \{A_3, A_4\}\}$ is easily checked to be tight.*

Two statements can be made about the properties of tightness according to Dunne et al. [23]:

Lemma 1 (Dunne et al. [23, Lemma 2])**.** *For a tight extension-set $\mathbb{S} \subseteq 2^{\mathcal{U}}$ it holds that*

1. *the $\subseteq$-maximal elements in $\mathbb{S}$ form a tight set, and*
2. *if $\mathbb{S}$ is incomparable then each $\mathbb{S}' \subseteq \mathbb{S}$ is tight.*

It is worth noting that the second statement of the lemma implies that if the downward-closure of an incomparable extension-set $\mathbb{S}$ is tight, then $\mathbb{S}$ itself is tight, too.

With these insights, the characteristics for the conflict-free and stable signatures[2] can already be described.

Theorem 1 (cf. Dunne et al. [23, Theorem 1])**.** *For a set $\mathbb{S} \subseteq 2^{\mathcal{U}}$ it holds that*

- $\Sigma_{cf} = \{\mathbb{S} \neq \emptyset \mid \mathbb{S}$ *is downward-closed and tight*$\}$,
- $\Sigma_{st} = \{\mathbb{S} \mid \mathbb{S}$ *is incomparable and tight*$\}$.

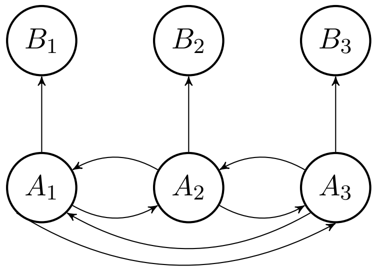

Figure 10: An argumentation framework F used in Example 13 in accordance with [23]

Example 13. *Consider the AF F_5 in Figure 10. There is an extension-set $\mathbb{S} = stb(F_5) = \{\{A_1, B_2, B_3\}, \{A_2, B_1, B_3\}, \{A_3, B_1, B_2\}\}$. One can check that $\mathbb{S}$ is tight, for instance, $E = \{A_1, B_2, B_3\}$: For each argument t not in E (i.e. B_1, A_2, A_3) there is an argument $s \in E$ such that $(s, t) \notin Pairs_{\mathbb{S}}$. In this instance A_1 plays this role for each t, since neither (A_1, B_1), (A_1, A_2), nor (A_1, A_3) is contained in $Pairs_{\mathbb{S}}$. The two remaining extensions are to be approached symmetrically.*

It is worth pointing out that the downward-closure of the extension-set is not tight, i.e. $dcl(\mathbb{S})$ is not tight. In fact, it is given that $\{B_2, B_3\} \in dcl(\mathbb{S})$, but for B_1 it is that $\{B_1, B_2, B_3\} \notin dcl(\mathbb{S})$, but (B_1, B_2) and (B_1, B_3) are contained in $Pairs_{dcl(\mathbb{S})} = Pairs_{\mathbb{S}}$.

For the remaining classical semantics, weaker properties must be found for the extension-sets. One of these is conflict-sensitivity. It checks for the absence of the union of any pair of the extensions in an extension set, whether it is justified by a conflict raised by $\mathbb{S}$.

Definition 20. *A set $\mathbb{S} \subseteq 2^{\mathcal{U}}$ is called conflict-sensitive if for each $A, B \in \mathbb{S}$ such that $A \cup B \notin \mathbb{S}$, it holds that $\exists a, b \in A \cup B : (a, b) \notin Pairs_{\mathbb{S}}$.*

Two known facts are that, on the one hand, for $a, b \in A$, $(a, b) \in Pairs_{\mathbb{S}}$ holds by definition and therefore conflict-sensitivity is determined only by arguments $A \in A \setminus B, b \in B \setminus A$ for $A, B \in \mathbb{S}$ and, on the other hand, that for incomparable $\mathbb{S}$ the property reduces to check for each $A, B \in \mathbb{S}(A \neq B)$ whether $a, b \in A \cup B$ exists so that $(a, b) \notin Pairs_{\mathbb{S}}$.

As with tightness (cf. Lemma 1), the following statements can be made about the property of conflict-sensitivity according to Dunne et al. [23]:

Lemma 2 (Dunne et al. [23, Lemma 4]). *For a conflict-sensitive extension-set $\mathbb{S} \subseteq 2^{\mathcal{U}}$,*

1. *the $\subseteq$-maximal elements in $\mathbb{S}$ form a conflict-sensitive set,*
2. *if $\mathbb{S}$ is incomparable then each $\mathbb{S}' \subseteq \mathbb{S}$ is conflict-sensitive, and*

[2]In fact, Dunne et al. also described characteristics for the naïve and stage signatures not considered in this thesis.

3. $\mathbb{S} \cup \{\emptyset\}$ *is conflict-sensitive.*

This allows the admissible and preferred signatures to be characterised[3]:

Theorem 2 (cf. Dunne et al. [23, Theorem 1]). *For a set $\mathbb{S} \subseteq 2^{\mathcal{U}}$ it holds that*

- $\Sigma_{ad} = \{\mathbb{S} \neq \emptyset \mid \mathbb{S}$ *is conflict-sensitive and contains* $\emptyset\}$,
- $\Sigma_{pr} = \{\mathbb{S} \neq \emptyset \mid \mathbb{S}$ *is incomparable and conflict-sensitive*$\}$.

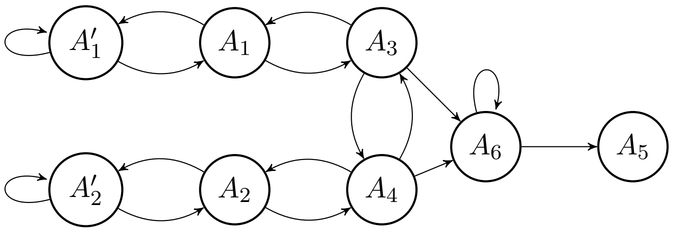

Figure 11: An argumentation framework for Example 14.

Example 14. *Consider the AF F_6 in Figure 11 in accordance with example 4 in [23]. Let $S_1 = \{A_1, A_2\}$, $S_2 = \{A_1, A_4, A_5\}$ and $S_3 = \{A_2, A_3, A_5\}$ with $\mathbb{S} = \{S_1, S_2, S_3\}$. Then $\mathbb{S} = pref(F_6)$. So $\mathbb{S}$ is conflict-sensitive, since for each pair of extensions, there exists a pair of arguments not contained in $Pairs_{\mathbb{S}}$. Still, since $S_1 \cup \{A_5\} \notin \mathbb{S}$ but (A_1, A_5) and (A_2, A_5) are contained in $Pairs_{\mathbb{S}}$, the extension-set $\mathbb{S}$ is not tight.*

Next, the complete semantics and its signature need to be looked at. For it, a more attenuated property is needed than for the previous ones. For this purpose, the completion-sets $\mathbb{C}_{\mathbb{S}}(E)$ is to be defined first.

Definition 21. *For an extension-set $\mathbb{S} \subseteq 2^{\mathcal{U}}$ and $E \subset \mathcal{U}$, the completion-sets $\mathbb{C}_{\mathbb{S}}(E)$ of E in $\mathbb{S}$ is defined as the set of $\subseteq$-minimal sets $S \in \mathbb{S}$ with $E \subseteq S$.*

The necessary property for the characterisation of the complete signature is called com-closed. Given an extension-set $\mathbb{S}$ and elements $\mathbb{T}$ thereof and there is no evidence of a conflict between the arguments in $Args_{\mathbb{T}}$, then $\mathbb{S}$ has to contain a unique superset of $Args_{\mathbb{T}}$ which is the completion-set. This is the opposite of the case when $\mathbb{S}$ would be conflict-sensitive, because then $Args_{\mathbb{T}}$ would have to be in $\mathbb{S}$.

Definition 22. *A set $\mathbb{S} \subseteq 2^{\mathcal{U}}$ is called com-closed if for each $\mathbb{T} \subseteq \mathbb{S}$ the following holds:*
if $(a, b) \in Pairs_{\mathbb{S}}$ for each $a, b \in Args_{\mathbb{T}}$, then $Args_{\mathbb{T}}$ has a unique completion-set in $\mathbb{S}$, i.e. $|\mathbb{C}_{\mathbb{S}}(Args_{\mathbb{T}})| = 1$.
For a com-closed extension-set $\mathbb{S} \subseteq 2^{\mathcal{U}}$ and $E \subseteq Args_{\mathbb{T}}$, the unique element of $\mathbb{C}_{\mathbb{S}}(E)$ is denoted by $C_{\mathbb{S}}(E)$.

[3]Dunne et al. [23] also characterised the semi-stable signature that is not considered in this thesis.

It has already been worked out that for incomparable sets the notions conflict-sensitivity and com-closed agree.

Since no precise characterisation has been given so far, only the properties found in [23] are described below:

Proposition 1 (Dunne et al. [23, Proposition 4])**.** *For each AF F, $co(F)$ is a non-empty, com-closed extension-set with $(\bigcap_{S \in co(F)} S) \in co(F)$.*

So it can just be said that the complete signature is probably a subset of these properties, i.e. $\Sigma_{co} \subset \{\mathbb{S} \mid \mathbb{S} \text{ is com-closed and } (\bigcap_{S \in \mathbb{S}}) \in \mathbb{S}\}$.

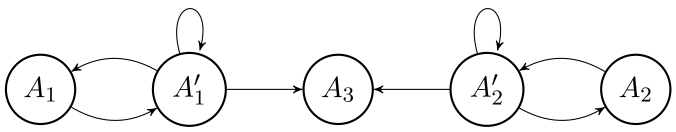

Figure 12: An argumentation framework for Example 15.

Example 15. *Consider the AF F_7 in Figure 12 in accordance with example 5 in [23]. There is $co(F) = \{\emptyset, \{A_1\}, \{A_2\}, \{A_1, A_2, A_3\}\}$, which is com-closed. This can be checked by having a look on the completion-set where $\mathbb{C}_{co(F)}(\{A_1\} \cup \{A_2\}) = \{\{A_1, A_2, A_3\}\}$.*

Also $co(F)$ is not conflict-sensitive as $\{A_1, A_2\} \notin co(F)$, but $(A_1, A_2) \in Pairs_{co(F)}$.

The characterisation of the grounded signature can be directly inferred from the fact that every argumentation framework F has exactly one grounded extension, i.e. $|grd(F)| = 1$, and every extension-set with $|\mathbb{S}| = 1$ is realisable by an AF with $(Args_{\mathbb{S}}, \emptyset)$ under the grounded semantics.

Theorem 3 (cf. Dunne et al. [23, Theorem 1])**.** *For a set $\mathbb{S} \subseteq 2^{\mathcal{U}}$ it holds that $\Sigma_{gr} = \{\mathbb{S} \mid |\mathbb{S}| = 1\}$.*

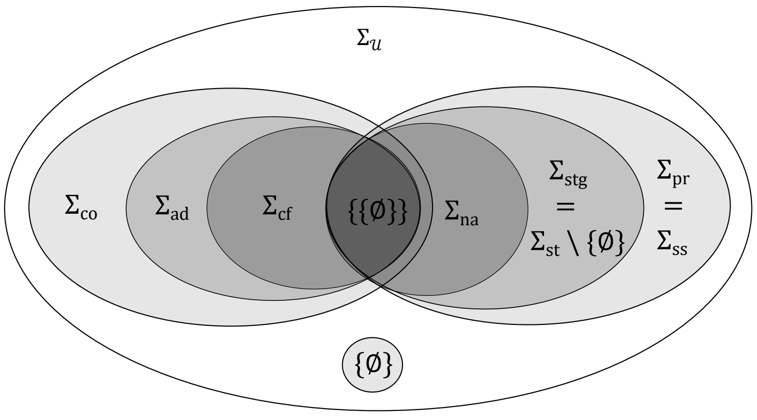

Figure 13: Relationship between the subsets of all signatures considered in [23].

Before briefly reviewing findings on the realisability of the signatures, it should be mentioned that there is a relationship between the signatures similar to that of the semantics. For this purpose, the properties of Lemma 3 and 4 can be considered.

Lemma 3 (Dunne et al. [23, Lemma 3])**.** *Every tight extension-set is also conflict-sensitive.*

Lemma 4 (Dunne et al. [23, Lemma 5])**.** *Each conflict-sensitive extension-set is com-closed.*

Moreover, Dunne et al. [23] show that there is a subset relationship between the considered signatures. The Venn diagram in Figure 13 illustrates these relations. The outer ellipse contains all extension sets over $\mathcal{U}$ and is therefore notated as $\Sigma_{\mathcal{U}} = \{\mathbb{S} \subseteq 2^{\mathcal{U}} \mid \mathbb{S} \text{ is an extension-set}\}$. The singleton $\{\emptyset\}$ belongs to the stable semantics, since it is only realisable by the stable semantics. However, $\{\{\emptyset\}\}$ is the set consisting of the extension-set $\{\emptyset\}$ and can be realised by all considered semantics. It is noted that signatures for further, non-classical semantics are included here, which have not been considered in detail so far. These are denoted by Σ_σ where $\sigma \in \{il, eg, na, stg, ss\}$.

The right side shows those signatures that consist only of incomparable sets. Furthermore, the intersection of the signatures with Σ_{co} is exactly the same with Σ_{gr} (plus the not further specified Σ_{eg} and Σ_{il}), which contain all extension-set $\mathbb{S}$ with $|\mathbb{S}| = 1$. Also, these semantics only have the empty extension set with conflict-free and admissible sets in common.

2.4.2. Realisability of Extension-Sets

For the previously gained insights into the characteristics of extension sets for the considered semantics, except for the complete one, it shall now be shown that they are sufficient. The concept of realisability will serve this purpose. It is to be understood as follows: An extension-set $\mathbb{S} \subseteq 2^{\mathcal{U}}$ is realisable under the semantics σ if there is an AF $F \in \mathbb{F}$, such that $\sigma(F) = \mathbb{S}$.

This is to be accomplished by a canonical argumentation framework that has an attack relationship between all arguments that do not appear together in any set of the extension set.

Definition 23. *Given an extension set $\mathbb{S}$, the canonical argumentation framework for $\mathbb{S}$ is defined as*

$$F^{cf}_{\mathbb{S}} = (Args_{\mathbb{S}}, (Args_{\mathbb{S}} \times Args_{\mathbb{S}}) \setminus Pairs_{\mathbb{S}}).$$

For conflict-free sets, this type of AF is sufficient, as described in the following proposition.

Proposition 2 (Dunne et al. [23, Proposition 5])**.** *For each extension set $\mathbb{S} \neq \emptyset$, which is downward-closed and tight,*
$cf(F^{cf}_{\mathbb{S}}) = \mathbb{S}$.

In order to work for the stable semantics, unwanted sets must be removed from the previously defined canonical argumentation framework. These can be removed by adding new arguments, which are attacked by all other sets of $\mathbb{S}$ but not by the unwanted extensions.

Definition 24. *Given an extension-set $\mathbb{S}$ and its canonical framework $F^{cf}_{\mathbb{S}} = (Arg^{cf}_{\mathbb{S}}, AR^{cf}_{\mathbb{S}})$, let $\mathbb{X} = st(F^{cf}_{\mathbb{S}}) \setminus \mathbb{S}$. It is defined that*

$$F^{st}_{\mathbb{S}} = (Arg^{cf}_{\mathbb{S}} \cup \{\bar{E} \mid E \in \mathbb{X}\}, AR^{cf}_{\mathbb{S}} \cup \{(\bar{E}, \bar{E}), (a, \bar{E}) \mid E \in \mathbb{X}, a \in Args_{\mathbb{S}} \setminus E\}).$$

Hence the following Proposition 3 [4].

Proposition 3 (Dunne et al. [23, Proposition 7]). *For each non-empty, incomparable and tight extension-set $\mathbb{S}$,*

$$st(F^{st}_{\mathbb{S}}) = \mathbb{S}.$$

For the other semantics based on admissibility, the previously defined canonical framework is not sufficient. It turns out, however, that such semantics can still be generated by means of so-called defence formulae. The defence formula specifies conditions for an argument to be in an extension. Therefore, $\mathcal{D}^{\mathbb{S}}_a \wedge a$ represents a superset of some acceptable positions.

Definition 25. *Given an extension-set $\mathbb{S}$, the* defence formula *$\mathcal{D}^{\mathbb{S}}_a$ of an argument $a \in Args_{\mathbb{S}}$ in $\mathbb{S}$ is defined as:*

$$\mathcal{D}^{\mathbb{S}}_a = \bigvee_{S \in \mathbb{S},\, s.t.\ a \in S} \ \bigwedge_{s \in S \setminus \{a\}} s.$$

$\mathcal{D}^{\mathbb{S}}_a$ given as a logically equivalent CNF is called CNF-defence-formula $\mathcal{CD}^{\mathbb{S}}_a$ of a in $\mathbb{S}$.

As it turns out, the procedure is divided into two parts. First, a canonical framework is created. Based on this, it is modified so that only elements from $\mathbb{S}$ are allowed. In the second step, self-attacking arguments are added, which are then attacked in turn by the arguments with new attack relations.

Definition 26. *Given an extension-set $\mathbb{S}$, the* canonical defence-argumentation-framework *$F^{def}_{\mathbb{S}} = (Arg^{def}_{\mathbb{S}}, AR^{def}_{\mathbb{S}})$ extends the canonical AF $F^{cf}_{\mathbb{S}} = (Args_{\mathbb{S}}, AR^{cf}_{\mathbb{S}})$ as follows:*

- $Arg^{def}_{\mathbb{S}} = Args_{\mathbb{S}} \cup \bigcup_{a \in Args_{\mathbb{S}}} \{\alpha_{a\gamma} \mid \gamma \in \mathcal{CD}^{\mathbb{S}}_a\}$*, and*
- $AR^{def}_{\mathbb{S}} = AR^{cf}_{\mathbb{S}} \cup \bigcup_{a \in Args_{\mathbb{S}}} \{(b, \alpha_{a\gamma}), (\alpha_{a\gamma}, \alpha_{a\gamma}), (\alpha_{a\gamma}, a) \mid \gamma \in \mathcal{CD}^{\mathbb{S}}_a, b \in \gamma\}$.

Next, the proposition that the characteristics are sufficient for admissibility is given.

Proposition 4 (Dunne et al. [23, Proposition 8]). *For each conflict-sensitive extension-set $\mathbb{S}$ where $\emptyset \in \mathbb{S}$, it holds that $ad(F^{def}_{\mathbb{S}}) = \mathbb{S}$.*

[4] Moreover, it holds that $st(F^{st}_{\mathbb{S}}) = stg(F^{st}_{\mathbb{S}})$ for the not considered stage semantics.

The canonical defence-argumentation-framework is also sufficient for the preferred and the complete semantics. For the preferred semantics, additional self-attacking arguments a' are added for each argument a in $Args_{\mathbb{S}}$ with an attack relation $a \hookrightarrow a'$.

Proposition 5 (Dunne et al. [23, Proposition 9]). *For each non-empty, incomparable and conflict-sensitive extension-set $\mathbb{S}$ there exist F with $pr(F) = \sigma$.*[5]

To close the concept of realisability, it should be mentioned that the complete semantics can be realised if the extension-set $\mathbb{S}$ is realisable under the admissible semantics.

Finally, it should be noted that Dunne et al. [23] also provided initial results for the question of the maximum number of extensions that can be achieved for an AF under a semantics. Before that, Baumann and Strass [9] conducted a detailed analytical and empirical study on the maximal and average numbers of stable extensions in abstract argumentation frameworks. As a result, they proposed a function that gives the maximum number of stable extensions for an AF with n arguments. This result can be understood to some extent as an answer to how much quantitative disagreement a semantics can express according to [23]. The key result of Baumann and Strass is the following Theorem 4. The interested reader is referred to their paper [9] for a detailed proof.

Theorem 4 (Baumann and Strass [9, Theorem 1]). *For any natural number n, it holds that*

$$\max_{F=(Arg,AR)\in\mathbb{F},|Arg|=n} |st(F)| = \Lambda(n)$$

where the function $\Lambda : \mathbb{N} \to \mathbb{N}$ is defined by

$$\Lambda(n) = \begin{cases} 1, & \textit{if } n = 0 \vee n = 1, \\ 2^s, & \textit{if } n \geq 2 \wedge n = 3s, \\ 4 \cdot 3^{s-1}, & \textit{if } n \geq 2 \wedge n = 3s+1, \\ 2 \cdot 3^s, & \textit{if } n \geq 2 \wedge n = 3s+2. \end{cases}$$

In order to gain insights about the extent of structural diversity a semantics can express and therefore to determine the number of extensions given a fixed amount of arguments, the diversity function is to be defined according to the results in [23].

Definition 27. *Given a semantics σ, the diversity function is defined as:*

$$\Delta_\sigma(n) = \max_{F\in\mathbb{F},||\sigma(F)||=n} |\sigma(F)|$$

[5]As shown in the proof to this proposition (Proposition 9 in [23]) it holds that $ss(F^{def}_{\mathbb{S}}) = \mathbb{S}$

The first results reveal for the conflict-free and admissible sets $\sigma \in \{cf, ad\}$ that $\Delta_\sigma(n) = 2^n$ is applicable[6]. This is easy to demonstrate because with an extension set $\mathbb{S}$ with $||\mathbb{S}|| = n$, the AF $(Args_\mathbb{S}, \emptyset)$ has 2^n conflict-free and admissible sets.

In their work, Dunne et al. [23] were able to prove that the results for the function Λ can be transferred to Δ_σ for all incomparable semantics considered so far as well as the not addressed naïve, semi-stable and stage semantics (i.e. $\sigma \in \{na, stg, ss\}$). For a proof, the reader is also asked to read this in [23], as before.

Theorem 5 (Dunne et al. [23, Theorem 5]). *For $\sigma \in \{na, st, stg, pr, ss\}$ and any natural number n, it holds that*

$$\Delta_\sigma(n) = \Lambda(n).$$

For the complete semantics, a conjecture was brought up by Baumann and Strass [10] in 2015, which lacked formal proof. It was not until a fair six years later that formal proof was provided by Ulbricht [48]. Ulbricht used findings about AFs for the proof instead of classical graph problems as for the other semantics before.

The proof uses a bi-cover of an AF and so-called proper guesses E' about subsets of arguments $\{a_1, ..., a_t\}$. These guesses are proper if no $1 \leq i \leq t$ such that $X_i \subseteq \bigcup_{i \neq j} X_j$. They are then applied to the characteristic function of an AF $E^* = \Gamma(E')$ and together with a complete extension of the reduct of the AF $E'' \in co(F^{E'})$ using the guesses yields a complete extension $E \in co(F) = E^* \cup E''$ (cf. Lemma 5.2 in [48]).

Another important finding is that if the guessed subset is proper, then all the arguments are in conflict of at least as many bi-cover sets as there are countable arguments in that guessed subset (cf. Lemma 5.3 in [48]).

With this knowledge, a good starting point for calculations is to find the size of the set of all proper sets that give complete extensions i.e. $|\{E \in co(F) \mid E \cap S = \Gamma(E')\}|$ with E' being the proper guess. For even numbers of arguments, the maximum number of extensions can be easily computed using the number of bi-cover sets and the number of arguments that are covered.

However, for odd numbers, different cases of attacks have to be distinguished in order to arrive at the desired inequality for odd numbers with some calculations. A complete proof can be found in [48].

The results in [48] can be formulated as the following function:

Theorem 6 (Ulbricht [48]). *For $\sigma = co$ and any natural number $n \geq 2$, it holds that*

$$\Delta_\sigma(n) = \begin{cases} 3^{n/2}, & \text{if } n \text{ is even}, \\ 4 \cdot 3^{(n-3)/2}, & \text{if } n \text{ is odd}. \end{cases}$$

As this subsection is comparatively long, important findings are summarised once again. A foundation has been created that allows the set of extensions which can

[6]Notice that Dunne et al. [23] also applied this upper bound to complete semantics, but newer results show a more precise upper bound. They showed that an AF $(\{a, a' \mid a \in Args_\mathbb{S}\}, \{(a, a'), (a', a), (a', a') \mid a \in Args_\mathbb{S}\})$ has 2^n complete extensions.

be expressed by an AF to be investigated. For this purpose, characteristics were named which describe under which specific conditions argumentation frameworks exist. Except for the grounded semantics, statements can be made in this regard. In addition, results were named for the more specific realisation of these AFs. In the end, the number of maximal extensions for a given set of arguments was examined for these semantics.

2.5. Computational Complexity

In this section, the topic of computational complexity will be briefly touched upon. To put it simply, this field of computer science deals with complexity classes and assigns problems to these classes. The different complexity classes are characterised by requirements for computation time and memory size. A problem that can be assigned to a complexity class has an algorithm that solves the problem within the requirements of the complexity class. Models have been created to consider problems without real-world systems and their inherent constraints. The best known model and the most widely used are Turing machines.

In the following, some basics are given in order to be able to describe and classify problems of abstract argumentation in the course of this thesis. Since this is only a brief outline of the subject, the interested reader is encouraged to refer to standard publications such as [40] and [1] for further details on this topic.

The first thing to establish is what an algorithm is. Basically and informally, an algorithm can be described as computing a function f and processing a set of finite rules so that a valid and finite input is processed into a finite output. Time is needed to process each step and space is needed to hold information in between.

For such an algorithm, its efficiency can be measured. This is typically measured by the number of operations of the function based on the length of the input. Another function T can capture the efficiency by returning the maximum number of operations for the input length of the algorithm. Since the fine-grained details of a calculation are often not relevant, the Big-Oh notation was introduced. A Definition 28 according to [1] is given.

Definition 28. *If $f, g : \mathbb{N} \to \mathbb{N}$, then i) $f = O(g)$ if there exists a constant c such that $f(n) \leq c \cdot g(n)$ for every sufficiently large n; ii) $f = \Omega(g)$ if $g = O(f)$; iii) $f = \Theta(f)$ is $f = O(g)$ and $g = O(f)$; iv) $f = o(g)$ if for every $\epsilon > 0$, $f(n) \leq \epsilon \cdot g(n)$ for every sufficiently large n; v) $f = \omega(g)$ if $g = o(f)$.*

Next, the complexity classes in this thesis should be briefly mentioned. These are mainly ones for decision problems as well as one for counting problems.

The first thing that should be noted is, that decision problems in class P are those that have a polynomial-time algorithm and have an answer or solution for an input i with length $|i|$ after $|i|^k$ many steps. Here, regardless of the size of k, an algorithm in class P is efficient. Large values for k, such as 200, do not seem efficient, but are so compared to other classes. Before briefly giving the definitions for the classes, it should also be noted that the class L contains those problems that can be solved in logarithmic size and polynomial time. P and L are therefore considered tractable, besides obviously trivial algorithms, while other classes are considered intractable.

Definition 29. *For a given input i with size $|i|$, the class* trivial *contains algorithms of k time, where k is independent of the input i.*

Definition 30. *For a given input i with size $|i|$, the class* L *contains algorithms of* $\log(|i|)$ *space.*

Definition 31. *For a given input i with size $|i|$, the class* P *contains algorithms of $|i|^k$ time with $k > 0$.*

In addition to the classes already introduced before, the first intractable class now follows. Problems in EXPTIME can be solved in exponential time, which is equivalent to a brute-force search.

Definition 32. *For a given input i with size $|i|$, the class* EXPTIME *contains algorithms of $k^{|i|}$ time with $k > 0$.*

Besides P and EXPTIME, there are problems that fall in the class of NP that are solvable in non-deterministic polynomial time. Here, there is a set of witnesses[7] for each instance of the problem. A witness is characterised by being a function that has polynomial size in $|i|$ and thus an element from this witness function $w \in W(i)$ can be verified in polynomial time. The result of a decision problem is **yes** if and only if there is at least one element of the witness function also a witness for instance i.

In more straightforward terms, for a given input i, it is easy to check whether it returns **yes** as a decision, but finding a valid input is considered difficult.

For the following Definition 33, borrowed from [1], the concept of the Turing machine is used. A Turing machine has an alphabet of symbols that can be stored on k tapes. In addition, the Turing machine has a set of possible states in which registers can be located. Finally, there is a so-called transition function that describes how the Turing machine has to execute each step.

Definition 33. *A decision problem $L \subseteq \{0,1\}^*$ is in* NP *if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time Turing machine M such that for every $x \in \{0,1\}^*$,*

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} \text{ such that } M(x,u) = 1$$

If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$, then u is a witness for x (with respect to L and M).

The complement of a decision problem Q is denoted by $\bar{Q}$. Therefore the following short definition can be given. However, it is important to note that the class coNP itself is not the complement of the class NP (cf. [1])!

[7]often called a certificate

Definition 34. *For a decision problem Q, the class* coNP *is defined as* *coNP* $= \{Q \mid \bar{Q} \in$ *NP*$\}$.

Before briefly touching on the polynomial hierarchy (PH), reductions and completeness as well as the complexity of counting problems, the class DP should be mentioned. DP is the difference class of decision problems that have an intersection of instances i of both a problem $Q_a \in$ NP and also $Q_b \in$ coNP are accepted. The Definition 35 is in accordance with the work in [39]. An example of a problem in DP is the SAT-UNSAT problem where two boolean formulae are tested to see if one is satisfiable and the other is not.

Definition 35. *Given two decision problems Q_a and Q_b, the class* DP *is defined as DP* $= \{Q_a \cap Q_b \mid Q_a \in NP, Q_b \in coNP\}$.

There is a hierarchy for the classes of polynomial runtime that consists of several levels. At the bottom level is P. In the levels above it are the two well-known NP and coNP. However, there are levels beyond this which are described by the classes Σ^P_{i+1} and Π^P_{i+1} for $i \geq 0$. One way to define them is to use a so-called oracle machine. It has a special tape (the so-called oracle tape) on which it can be checked whether there is $o \in O$ for some problem O. This access to such a check is called an oracle.

Definition 36. *For the polynomial hierarchy, P is equal to Σ^P_0, Π^P_0 and Θ^P_0.*

For all $i \geq 0$ there is $\Sigma^P_{i+1} = NP^{\Sigma^P_i}$, $\Pi^P_{i+1} = coNP^{\Sigma^P_i}$ and $\Theta^P_{i+1} = P^{\Sigma^P_i}$. With $Q \in \{P, NP, coNP\}$, the problem Q^O is then decided by the oracle machine with access to an oracle O that is complete.

Next, hardness describes problems that are at least as hard as all others in the class. To show that a problem Q_b is at least as hard as another problem Q_a in the class C, one uses the so-called reduction $\leq_p$, where there is a function f that can be computed polynomial-time that transforms every element in the domain to the codomain if and only if for $x \in Q_a$ there is $f(x) \in Q_b$. Moreover, the problem Q_b is complete if it is part of the class C.

Definition 37. *For a given complexity class C, a problem Q is hard for C if for all problems Q' in C there is a (polynomial-time) reduction $Q' \leq_p Q$.*

Furthermore, if $Q \in C$, then Q is complete for C, also abbreviated as C-c.

Finally, counting problems should be dealt with briefly. In this type of problem, one is not interested in finding only one witness, which is the class NP, but the actual number of them. The class #P describes these problems. The closeness of #P to NP can be seen in Definition 38 which is taken from [1]. However, #P is a problem class where a function f returns a natural number.

Definition 38. *A function $f : \{0,1\}^* \to \mathbb{N}$ is in #P if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time Turing machine M such that for every $x \in \{0,1\}^*$:*

$$f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid M(x,y) = 1\}|.$$

This lays all the groundwork for considering the computational complexity in abstract argumentation.

2.6. Eliciting Argumentation Frameworks

This Master's thesis explores the idea of the elicitation of argumentation frameworks under the condition that only questions related to the extensions of the system are asked. Therefore, the concept of eliciting argumentation frameworks is an important prerequisite for deeper investigation. In the elicitation scenario, there is an agent that has an argumentation framework and does not reveal at least parts of it. The goal is to uncover this abstract argumentation framework by asking questions. Kuhlmann [34] introduces this new line of reconstruction of argumentation frameworks and provides some preliminary insights.

The elicitation of abstract argumentation frameworks is thus an interactive scenario in which an agent is questioned by an interviewer. In doing so, it is permissible and intentional to choose the further questions based on the answers in order to approach the agent's hidden AF. That makes this problem unique compared to previous related work such as the problem of learning AFs based on labellings [43] or synthesising AFs [37] based on example extensions.

In the scenario described so far, it is unclear whether the agent hides the entire argument framework or only parts of it. Previous work on the topic assumes that the set of arguments is known and only the attacks between the arguments are hidden by the agent. Also, it is important to note that the agent in the scenario answers every question truthfully, like an oracle.

Kuhlmann [34] already noted that finding the exact identical AF, especially the attack relation, is difficult because although even a few questions can significantly reduce the amount of possible solutions, it is often not enough to obtain an unique solution as it is possible that the same scenario can be modelled differently resulting in multiple AFs that represent that scenario. In other words, the arguments are the same between two AFs, but the notion of attack differs. In fact, it can be assumed that the reconstruction of the syntactically identical abstract framework is very complicated. An example of this has already been shown in the motivational part of the introduction in Subsection 1.1. As a consequence, the goal is to find an equivalent solution with respect to a semantics. σ-equivalence describes the notion of standard equivalence in accordance with Oikarinen and Woltran [38] and states that if one AF has the same extensions under a given semantics as an AF to be compared, then they are considered equivalent.

This concept guarantees that the desired result, i.e. the accepted arguments, is achieved by both compared argumentation frameworks. If however new information, that is attacks and new arguments, is added to the two AFs, then the two may no longer be equivalent under σ-equivalence. Nevertheless, stricter requirements for equivalence can also be made, as will be clarified below.

So if reconstructing the exact AF seems too complicated and, considering the scenario, several AFs describe the same, then it needs a concept of when the AFs express the same outcome. That is why Oikarinen and Woltran [38] introduced another type of equivalence. The concept of strong equivalence, as they named it, compares the relation between argumentation frameworks e.g. F and F'. The two AFs are equivalent under any semantics, if both AFs are still equivalent under the standard concept of equivalence when introducing new arguments and attacks from a third AF G. So $F \cup G$ and $F' \cup G$ are equivalent.

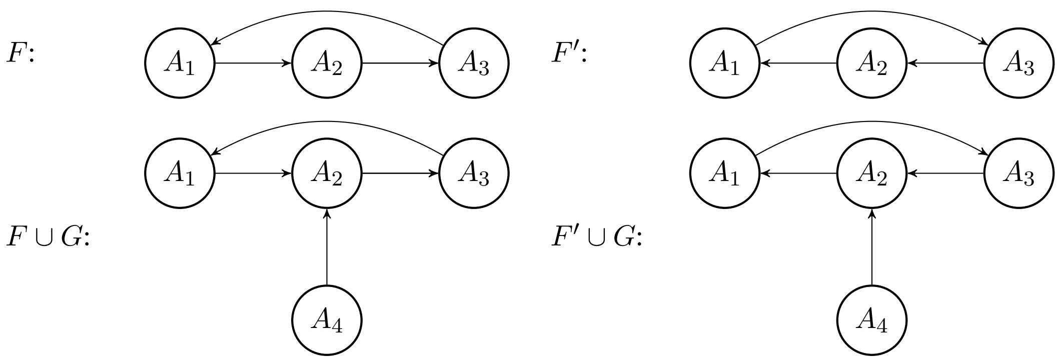

Figure 14: The abstract argumentation frameworks modelled in accordance with Example 16.

Example 16. *To illustrate the two different notions of equivalence, this example is intended to show the differences in accordance with [38]. Consider the following two argumentation frameworks* $F = (\{A_1, A_2, A_3\}, \{(A_1, A_2), (A_2, A_3), (A_3, A_1)\})$ *and* $F' = \big(\{A_1, A_2, A_3\}, \{(A_3, A_2), (A_2, A_1), (A_1, A_3)\}\big)$ *as illustrated in Figure 14.*

These two AFs produce the same extensions under most known semantics. Thus, both have the same unique extension, the empty set, using the preferred semantics. They are therefore σ-equivalent with σ being in this case the preferred semantics (so $\sigma = \{pr\}$). But if new information of an AF $G = (\{A_3, A_4\}, \{(A_4, A_3)\})$ *is added, so as the union of F and G, then a different picture emerges.*

The unique preferred extension is now $\{A_3, A_4\}$ for $F \cup G$ and while for $F' \cup G$ it is now $\{A_1, A_4\}$. The two AFs are thus not strongly equivalent, because the implicit information that did not come into play before with σ-equivalence now actually makes a difference between the AFs.

The formulation of the strong equivalence notion is therefore motivated by the fact that it makes it decidable as a property whether two AFs have the same implicit information. It also brings the ability of local simplification, i.e. without looking at an entire framework. Some subframeworks F and F' are replaceable if they are strongly equivalent, whereas under σ-equivalence the ability to replace one AF with the other cannot be guaranteed.

For the further work of the thesis, the notion of σ-equivalence will nevertheless be used, as approaches for the (re-)construction of AFs under this concept are already available. However, this Master's thesis is structured in such a way that further types of equivalence can be investigated in the reconstruction based on it. The choice of semantics for equivalence is also limited to the classical ones:

Definition 39. *Let $\sigma \in \{ad, cf, co, gr, pr, st\}$, then two AFs $F = (Arg, AR)$ and $F' = (Arg', AR')$ are σ-equivalent if they possess the same σ-extensions, i.e., $\sigma(F) = \sigma(F')$.*

Next, the questions will be dealt with on the surface before they are addressed in greater depth in the later part of this thesis. The questions to be asked can be divided into two categories. On the one hand, there are syntactical and on the other semantic questions. Syntactical questions refer to the arguments and their attack relationship. These questions can concern the whole AF, such as "Does the AF contain any self-attacks?" or "How many attacks are there?", as well as individual arguments, such as "Does argument a attack argument b?" or "Is there an attack between arguments c and d?". In contrast, semantic questions have a connection to the extensions and can be "Is $E \subseteq Arg$ a σ-extension of the AF?", "What are the grounded extensions of your AF" or "How many extensions exist with respect to σ?", for example.

Another distinctive feature is already clear from the example questions. On the one hand, there are decision questions, i.e. those that are polar and can be answered with yes or no, and there are queries that are functional and are answered, for example, a set, an attack pair or the number of arguments in an extension. In the context of this Master's thesis, the restriction is made that the agent may only be asked decision questions that are semantic questions, too. In addition, it will be determined later which exact questions the agent can answer. Nevertheless, it is possible to pose queries in the same way and this could be the subject of further research.

That elicitation is possible in principle is demonstrated by the two-step naïve approach. It can be described as follows [34]:

1. Ask the questions of "Is $E \subseteq Arg$ a σ-extension of F_A" with respect to every possible subset of arguments in given Arg. As a result, the σ-extensions of F_A are known.
2. Compute all possible AFs from Arg. For every possible AF G, compute $\sigma(G)$ and check whether $\sigma(F_A) = \sigma(G)$. If this is the case, a solution was found and an equivalent AF $F_E := G$ is returned.

This naïve approach makes use of a single decision question, namely whether a subset of arguments is an extension of the agent's argumentation framework with respect to a given semantics. Already with this one question, σ-equivalent argumentation frameworks can be elicited. However, this approach is considered very inefficient and therefore hardly practicable for larger AFs. However, it shows that in principle semantic decision questions can be used to solve the problem.

3. Related Work

The work done so far on the reconstruction of argumentation frameworks will be presented in this section. The reconstruction approaches are based on the fact that information or parts of it are already given, from which the AF is (re-)constructed. From the perspective of elicitation, where sufficient information must first be acquired, this procedure is the second stage. Therefore, the following elaborations are a building block for the realisation of the elicitation approach of argumentation frameworks.

First, a look at the work on synthesising argumentation frameworks from examples by Niskanen et al. [37] will be taken. The synthesis problem here is an extension of realisability. In cases of non-realisability, e.g. those occurring in the dynamics of argumentation as in the revision of AFs, the semantically closest AF is to be generated from existing knowledge. Thus, cases are particularly considered in which no AF can actually represent the knowledge.

After that, the work on learning attacks from labellings is briefly examined. Riveret et al. [43] provide an algorithm for grounded labellings as input. This differs from the approach in [37]. In particular, the semantics for the input are restricted and additional information is added with labellings compared to extensions.

Last, Kido and Liao's [33] work on what they call the inverse problem in the construction of argumentation frameworks is briefly reviewed. Here, a conflict between arguments is to be detected from a noisy set of acceptable arguments. They propose a Bayesian solution that allows for subjective beliefs about the existence of an attack and also handles uncertainties about the probability that an attack actually arises.

3.1. Synthesising AFs from Examples

The AF synthesis problem is introduced in the paper by Niskanen et al. [37] This problem is an extension of the realisability previously explained in the background section (cf. Subsection 2.4). The pursued goal is to always generate an AF from existing knowledge even if none could actually be constructed from the knowledge. At the same time, the generated AF should also be the semantically closest to the given knowledge. In addition to the implementation of this problem, they provide MaxSAT-based algorithms as well as encodings for Answer-Set Programming (ASP) in which the AF can be generated from positive and negative examples of the extensions and the associated semantics. The algorithm synthesises an AF whose cost is minimal over all unsatisfied examples.

First of all, it must be clarified what an example actually is. Examples represent semantic information that may be weighted.

Definition 40. *An example $e = (S, w)$ is a pair with $S \subseteq A$ where A is a set of arguments and w as an positive integer. S_e represents the set of an example e and w_e the weight respectively.*

Similar to realisability, $\mathbb{S}_E = \{S_e \mid e \in E\}$ expresses the set that contains all sets of arguments from the example sets.

Next, an instance of the AF synthesis problem is to be defined. It consists of a set of arguments A, positive and negative examples E^+ and E^- respectively, and a semantics σ.

Definition 41. *The quadruple $P = (A, E^+, E^-, \sigma)$ is an instance of the AF synthesis problem where A is a non-empty set of arguments, E^+ and E^- are two sets with positive and negative examples respectively, and a semantics σ with $(\mathbb{S}_{E^+} \cup \mathbb{S}_{E^-}) \subseteq 2^A$.*

With respect to an argumentation framework F, F satisfies all positive examples e if $S_e \in \sigma(F)$. Likewise, F satisfies all negative examples if $S_e \notin \sigma(F)$.

The cost associated with an AF with respect to an instance is denoted by $cost(P, F)$ and is the sum of all weights of examples not satisfied by the AF. The cost function is given as follows:

$$\sum_{e \in E^+} w_e \cdot I(S_e \notin \sigma(F)) + \sum_{e \in E^-} w_e \cdot I(S_e \in \sigma(F))$$

The indicator function $I(\cdot)$ returns 1 if the property of membership in a set is satisfied and otherwise 0.

In summary, the task can be given in a formal representation, where an input of arguments, positive and negative examples, and semantics is given as a precondition, and the output is to ensure that an argumentation framework is found whose cost is minimal.

The Task of Argumentation Framework Synthesis

`REQUIRE:`	$P = (A, E^+, E^-, \sigma)$ such that $(\mathbb{S}_{E^+} \cup \mathbb{S}_{E^-}) \subseteq 2^A$
`ENSURE:`	An AF F^*
`TASK:`	Find an AF F^* with $F^* \in arg\ min_{F=(Arg,AR^*)-}(cost(P, F))$

The properties of the synthesis problem that have to be fulfilled are described by Niskanen et al. [37]. The relation to Dunne et al.'s work [23] on realisability is established by working out conditions for which 0-cost solutions exist and which properties fulfil 0-cost solutions. The interested reader is invited to read the details in section 4 in [37].

At this point, the reader is also invited to study the MaxSAT-based algorithms for conflict-free, admissible, stable and complete semantics, as well as MaxSAT-based counterexample-guided abstraction refinement (CEGAR) approach for preferred semantics and also the answer-set programming algorithm in Section 6 of [37].

In summary, this approach can be used to find argumentation frameworks that have as many positive examples as possible as extensions while producing as few negative examples as possible. Pre-determined weights among the examples are taken into account. However, since only one AF is generated at minimal cost, it is the case that in cases where new knowledge is added, an AF is generated again. It is therefore not possible to generate a set of AFs and only continue searching in this set when new knowledge is added. Therefore, for elicitation, the synthesis of AFs can only take place when all the necessary information for finding an AF is given. Since the agent in this thesis always answers truthfully, this approach is suitable for generating AFs from the extensions elicited.

3.2. Learning Attacks in Probabilistic AFs

Another approach to constructing argumentation frameworks comes from Riveret and Governatori [43]. In contrast to the previously mentioned synthesis problem (cf. Subsection 3.1), four-valued grounded labellings are assumed here as input knowledge. The aim here is to learn from these the attacks between the arguments and thus to reconstruct them. The algorithm proposed by Riveret and Governatori has a weighted argumentation graph as a substructure for this and uses rules that adjust the weights in such a way that a decision can be made about whether an attack occurs or not. In the end, an argumentation framework according to Dung can then be extracted from the weighted argumentation framework. An overview of the algorithm is therefore provided below, with definitions taken from [43]. For a detailed description, the reader is referred to paper [43].

Although the proposed algorithm will be described in a concise manner, a few preliminary definitions must nonetheless be made. For subsequent work, for instance, the concept of a sub-framework[8] is needed.

Definition 42. *A sub-framework H of an argumentation framework $F = (Arg, AR)$ is an argumentation framework (Arg_H, R_H), where $Arg_H \subseteq Arg$ and $\forall a, b \in Arg_H, (a \hookrightarrow b) \in AR$ iff $(a \hookrightarrow b) \in AR_H$.*

As mentioned at the beginning, a four-valued labelling is expected as input for the algorithm. The labels themselves are the three known labellings in, out, undec and additionally the label off, which signifies arguments that do not exist in the semantics. The labelling itself works as known from the three-value labelling.

Definition 43. *Let $\Lambda = \{\mathsf{in}, \mathsf{out}, \mathsf{undec}, \mathsf{off}\}$ be the labels and $F = (Arg, AR)$ be an AF. A labelling is a total function $\mathcal{L} : Arg \to \Lambda$.*

There is always only one unique grounded $\{\mathsf{in}, \mathsf{out}, \mathsf{undec}\}$ labelling for an argumentation framework, but several grounded $\{\mathsf{in}, \mathsf{out}, \mathsf{undec}, \mathsf{off}\}$ labellings can be generated for the sub-frameworks. Those arguments that are not expressed by the sub-framework are then labelled off. The other arguments are labelled according to the grounded $\{\mathsf{in}, \mathsf{out}, \mathsf{undec}\}$ labelling as known.

[8]It may be noted that [43] terms this concept a sub-graph.

Definition 44. *Let $F = (Arg, AR)$ be an argumentation framework and $H = (Arg_H, AR_H)$ an induced sub-framework of F. A grounded $\{\mathsf{in}, \mathsf{out}, \mathsf{undec}, \mathsf{off}\}$ of F is a $\{\mathsf{in}, \mathsf{out}, \mathsf{undec}, \mathsf{off}\}$ labelling such that:*

- *every argument in Arg_H is labelled according to the grounded $\{\mathsf{in}, \mathsf{out}, \mathsf{undec}\}$ labelling of H, and*
- *every argument in $Arg \setminus Arg_H$ is labelled* off*.*

The next step is to define the weighted argumentation framework. It is used within the algorithm and extends Dung's argumentation framework with weights for each attack. The article [24] gives a good overview of the topic and the reader is referred there for details.

Definition 45. *A* weighted argumentation framework *is a triple $W = (Arg, AR, w)$ where (Arg, AR) is an abstract argumentation framework according to Dung and $w : AR \to \mathbb{R}$ a function assigning weights to every attack.*

On the basis of the assigned weights, the algorithm calculates those attacks that are necessary to construct an AF in which the input labellings can be generated. For this purpose, the weights have different interpretations: An attack $a \hookrightarrow b$ is

- discarded if $w_{a \hookrightarrow b} < 0$,
- confirmed if $w_{a \hookrightarrow b} > 0$ and
- undecided if $w_{a \hookrightarrow b} = 0$.

Finally, the so-called credit rules are required, according to which the algorithm progressively decides with each labelling to be processed whether an attack is confirmed or discarded. For this purpose, four rules are applied according to which the attack relations are weighted depending on the labelling of two arguments. The interested reader can delve into the exact rules in the work of Riveret and Governatori [43].

Algorithm 1 Algorithm for learning attacks from labellings in accordance with [43]

```
Require: $Arg$ as a set of arguments, $\mathfrak{L}$ as a set of grounded $\{\mathsf{in}, \mathsf{out}, \mathsf{undec}, \mathsf{off}\}$ labellings
Ensure: An argumentation framework $F$
  $C \leftarrow (Arg_C, AR_C, w)$
  For any $a, b \in A_C$, initialise $w_{a \hookrightarrow b}$
  while (there is computational budget) $\vee$ (an attack is undecided) do
      Get a labelling instance $\mathcal{L}$ from $\mathfrak{L}$
      Apply credit rules $R_i(\mathcal{L}, C)$ where $i = 1..4$
      Optionally prune discarded attacks in $C$
  $F \leftarrow prune(C)$
```

As before for the synthesis of AFs from examples (cf. Subsection 3.1), the task of the algorithm can be specified as follows:

	The Task of Learning Attacks from Examples
`REQUIRE:`	Arg as a set of arguments,
	$\mathfrak{L}$ as a set of grounded $\{\mathsf{in}, \mathsf{out}, \mathsf{undec}, \mathsf{off}\}$ labellings
`ENSURE:`	An argumentation framework F
`TASK:`	Find an AF F that yields as many labellings as it can.

The algorithm proposed by Riveret and Governatori [43] iteratively assigns weights according to the credit rules for the attack relations as long as a computational budget has not yet been reached or as long as each attack has not yet been confirmed or discarded. As soon as the condition is fulfilled, the AF induced so far is returned. Algorithm 1 illustrates this. The reader is asked to study the source literature [43] for a in-depth explanation of how it works.

Lastly, Riveret and Governatori [43] showed that, given an inexhaustible computational budget, a argumentation framework was almost certainly found that was indistinguishable from the labelling source. However, the algorithm does not support any other of the classical semantics. Moreover, a four-valued labelling forms a special case. In contrast to synthesising with extensions, not only are classic three-valued labellings used here, but extra information is introduced with an additional label, which must first be extracted by an agent, for example. This means that significantly more information has to be revealed or discovered.

3.3. A Bayesian Approach to the Direct and Inverse Problem

The work of Kido and Liao [33] introduces the concept of the inverse problem. Here, a noisy set of acceptable arguments is assumed and then attack relations are sought that explain the sets well in terms of semantics. The inverse problem is the converse of the direct problem, where sets of acceptable arguments are to be found with respect to a semantics for a given attack relation. The abstract probabilistic model proposed by Kido and Liao works with both the direct and inverse problems when the attack relations are probabilistically distributed. The following description and definitions are based on Kido and Liao's work [33].

First of all, both the direct and inverse problem are to be specified according to [33]:

Direct problem	Given an attack relation between arguments of an argumentation framework, a direct problem aims to find sentiments regarding acceptability of the arguments, i.e. for example extensions, defined with respect to a semantics.
Inverse problem	Given noisy sentiments regarding acceptability of arguments, an inverse problem aims to find an attack relation between the arguments explaining the sentiments well in terms of the semantics.

The inverse problem is based on finding an attack relation for a given acceptability, e.g. a set of extensions. For this purpose, both the semantics and the arguments are already known and, if necessary, already known attack relations can be included.

Definition 46. *An* inverse problem *is defined as follows: Given an acceptability acc, an attack relation is to be found that satisfies $acc = \sigma(Arg, Arg^k, AR)$ where σ is the semantics, Arg are the fixed arguments, Arg^k are already known arguments and AR is the attack relation.*

The noise mentioned before in the specification of the problem are thereby effect irrelevant for the given semantics or can even be false or inaccurate observations. This noise must also be taken into account when solving the problem. It extends the previous equation by η, which denotes the noise: $acc = \sigma(Arg, Arg^k, AR) + \eta$. To solve the inverse problem, a probabilistic model is formulated, given a probabilistic distribution over the attacks and the acceptance. The generative model of abstract argumentation is taken for the inverse problem to trace the causalities back to the actual argumentation framework starting from acceptance. This is achieved by computing the posterior distribution over attack relations with the given acceptability variables. For more specific details, the reader is referred to the paper [33]. There, theoretical and empirical evaluations of the correctness of the model are also carried out.

The probabilistic model formulated by Kido and Liao [33] is suitable to consider uncertainty due to missing data or noisy observations and still construct an argumentation framework. In the scenario of this Master's thesis, all information received is undistorted, since it is assumed as a prerequisite that the interviewed agent always answers truthfully. Nevertheless, for further research on the topic, e.g. elicitation using noisy sentiments of multiple agents may be of interest.

4. Algorithms for Eliciting Abstract Argumentation Frameworks

The elicitation is a two-step procedure in which the agent is questioned in the first stage by means of the input information on the arguments and the semantics. For this purpose, a question is formulated, which the agent answers. Based on this answer, a new question is formulated. Once all the information has been collected, i.e. in the scenario of this thesis that all the extensions have been found, the procedure moves on to the second stage, in which an equivalent argumentation framework is reconstructed from the collected information. This argumentation framework is the result of the procedure. Figure 15 visualises this approach.

In the following work, we focus primarily on the first interviewing stage of the procedure. For the second stage, called the reconstruction, there is existing work on canonical AFs and synthesising argumentation frameworks (see also Subsections 2.4 and 3.1). Therefore, we only briefly discuss the extent to which the information collected allows for a more immediate reconstruction.

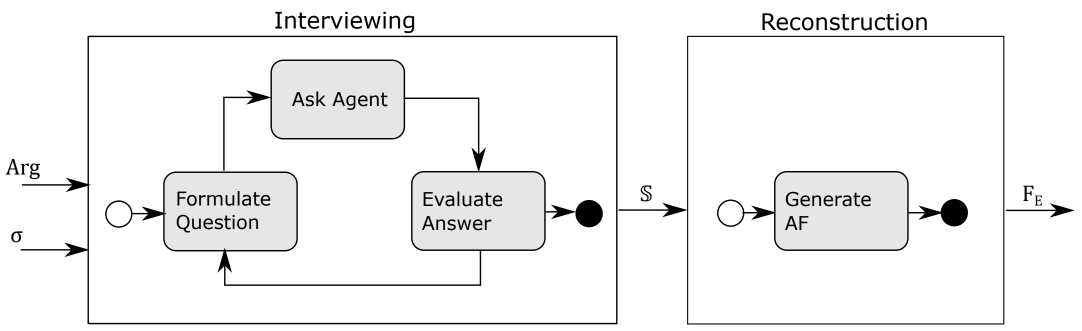

Figure 15: Illustration of the two stages of the elicitation procedure.

This section hence deals with the development of the algorithms for the interviewing stage in eliciting argumentation frameworks, taking into account that only questions are asked that are also related to semantics. For this purpose, we present the general structure of the algorithms in more detail. Next, the questions that can be asked of the agent are narrowed down and then formulated. The collected properties of the investigated semantics and their extensions are the basis for this. Subsequently, the developed algorithms are introduced and we describe for each semantics their functioning.

Finally, we discuss in further depth the reconstruction and its limitations with the collected information.

4.1. Detailed Specification of the Elicitation Procedure and Algorithms to Be Developed

The short introductory description of the two stages of the elicitation procedure should first be deepened. For this purpose, we present the procedure as a more formal description in Algorithm 2. The general goal of the elicitation algorithm is to take given information and the possibility of collecting further information through questioning and then construct an argumentation framework that satisfies the given equivalence conditions. In the case of this Master's thesis, this is the σ-equivalence.

Thus, an interaction takes place between an interviewer and an agent, whereby the interviewer can ask those questions that the agent can also answer. An agent in turn has an argumentation framework F_A and a question pool $\mathfrak{Q}$ that it can answer. We therefore define an agent that answers all questions truthfully as $Agent = (F_A, \mathfrak{Q})$. An agent has a type or configuration, which is described by indices. The upper indices restrict the question pool and the lower indices describe further properties, e.g. extensions already known before the interview. We say an agent can solve or answer a question $Q \in \mathfrak{Q}$ if for each question input x there exists an answer y.

The interview itself takes then place between the interviewer and the agent as a first stage. It is a question cycle that depends on the semantics under consideration. The first question to start with must be defined for each semantics. It is asked in the

Algorithm 2 Eliciting procedure with its two distinct stages

```
Require: $F_A = (Arg, AR)$ as the agent's AF, $Agent = (F_A, \mathfrak{Q})$ as the agent,
         $\sigma$ as the semantics
Ensure: An argumentation framework $F_E$ such that $\sigma(F_E) = \sigma(F_A)$
  $\mathbb{S} \leftarrow$ INTERVIEW AGENT$(Agent, Arg, \sigma)$
  $F_E \leftarrow$ RECONSTRUCT AF$(\mathbb{S}, \sigma)$
```

initial state and therefore cannot be selected based on previous answers. This question does not have to be the same for all semantics. For example, it seems reasonable to ask about the existence of any extension for the stable semantics. However, this question would be worthless for some of the other classical semantics, since they always have at least one extension.

For all later cycles, in turn, previous answers of the agent can be used in the selection of the question to be asked.

The selected question is then asked to the agent and the agent gives his answer to it. The answer can then be evaluated. On the one hand, the result of the questioning can be used to further restrict the search space by removing impossible extension sets. On the other hand, it can be decided whether enough questions have been asked to construct an equivalent argumentation framework of the agent. If continued, the question cycle takes place again from the beginning.

However, if it is decided that there are no further questions to be asked, then this stage is finished. For this first stage, we develop algorithms in the Subsection 4.3. There, different from what is depicted in Algorithm 2, we keep the agent implicit as a parameter.

The result of this interviewing stage is an extension set $\mathbb{S}$. We transfer this together with the semantics into the reconstruction stage. We will discuss later in Subsection 4.4 which options for reconstruction are possible from the information obtained.

With this approach established, all that remains is to specify the questions that can be asked in order to develop algorithms that perform the filtering, the questioning process per semantics and the discussion on the generation of σ-equivalent argumentation frameworks for the different semantics.

4.2. Question Forming

An essential part of the process is the questions to be asked to the agent. For this purpose, we examine the scenario in more detail, in order to be able to define some restrictions.

An agent has an argumentation framework that is hidden to the outside. In a real-world example, the agent may have read a newspaper article that places arguments for and against a topic. An interviewer would then like to question the agent to find out which arguments the agent accepts, in order to also construct an argumentation framework that comes to the same conclusion under the same acceptability semantics as that of the agent.

In the considered scenario of this Master's thesis, the agent always answers truthfully to the interviewer's questions. Nevertheless, it is assumed that the agent is reluctant to disclose information. This will become important for the selection of question types again soon.

In such a scenario, basically an infinite number of questions could be asked. Therefore, some restrictions and assumptions have to be made in order to sufficiently narrow down the scope of the problem. Thus, the categorisation of questions should be taken up again.

It is helpful to look at existing work on the analysis and the definition of meaningful categories for questions, as for example, in Graesser and Person's work [30] in the analysis of tutoring or Lehnert's work [36] in computational models for question answering. The distinction between yes-no questions and wh-questions, previously called decision questions and queries, already described in the section on elicitation, can be found there in the form of the question category verification for yes-no questions and other categories such as casual consequences, concept completion or quantification and many more for queries. However, Singer [44] suggests that yes-no questions are not one-dimensional but two-dimensional. For this purpose, there are two question acts, namely request reports (wh-questions or also called queries before) and request verification (yes-no questions or decision questions). These can then be assigned to the categories according to [44].

Inspired by the question types from [36] and the multidimensionality from [44], the dichotomy between decision questions and queries will be presented once again and the two-dimensionality will be considered with regard to questions types an agent can answer in the context of elicitation of abstract argumentation frameworks.

On the one hand, questions can also be categorised according to the question result similar to Singer's question acts. Decision questions are those that are posed as yes-no questions, i.e. are polar in that they only allow two possible answers. In addition, there are queries (also called wh-questions) that are asked as open questions and are functional in the sense that they provide an outcome to the initial question that is more complex than the decision for 'yes' or 'no'. For example, such results can be sets with arguments in the context of formal argumentation.

On the other hand, a distinction can be made between syntactic and semantic questions as two broader question classes. The former relate to the structure of the graph, i.e. to the arguments and their attack relation, while the latter have a connection to the extensions. In the context of the Master's thesis, these are therefore properties and arguments of the extensions.

Table 1 outlines both dimensions of the question types and gives examples. In contrast to Singer, in addition to the question acts (decision question and query),

	Question Class	
Question Act	Semantic Questions	Syntactic Questions
Query	Which arguments are credulous accepted?	Is there an odd-cycle attack relation?
Decision Question	Is $E \subseteq \sigma(F)$?	Is an Argument a not in an Extension E and not attacked by E?

Table 1: Two-dimensional representation of the question types with example questions from the subordinate question categories.

there is the dimension of question classes instead of question categories. The reasoning behind this is that question classes are superordinate to question categories. Within a question class, different question categories can be grouped, e.g. in the naïve procedure, the question about the existence of an extension has already been addressed (cf. Subsection 2.6), which can be included in the category of existence questions.

After answering the question of which question acts and question classes an agent can answer and the subsequent compilation of possible questions, the question categories for the scenario of the Master's thesis are compiled.

4.2.1. Selection of Question Types and the Compilation of Questions

Since only questions related to extensions are to be asked in this Master's thesis, primarily semantic questions are of interest. Therefore, the algorithms builds on this question class. So we will not allow for syntactic questions, even if they have a relation to extensions. I.e. the attacks between arguments may not be asked, but also for example whether an argument is not part of the extension because it is attacked by the latter is not permitted to be asked.

With regard to the distinction between decision questions and queries, it can be assumed that queries would oversimplify the algorithms, since the agent answers truthfully to them. For example, if an agent answered truthfully to the query "What are all the extensions?", then the whole interview part of the elicitation would be superfluous. At this point, one must return to the agent's reluctance to reveal information. This is because it seems more natural for the agent to answer only decision questions in a real world scenario, since he is in a passive role similar to an examination.

In summary, this means that for the given setting, questions are sought that are semantic and have the form of a decision question. In addition, syntactic decision questions are to be found that are related to extensions.

Formulating Questions A starting point for formulating the questions are computational problems in abstract argumentation frameworks with respect to semantics. Dvořák and Dunne published an article [26] on this that deals with the computational problems and their complexity. Next, these problems are to be outlined and the questions concerning them are to be collected.

First of all, there are computational problems on the acceptance status of the arguments within the extension set. They represent different degrees of scepticism and are called credulous and sceptical reasoning respectively. Arguments that occur in all extensions are accepted sceptically. Arguments that are not found in all but at least one extension are accepted credulously. Given the agent's AF $F_A = (Arg, AR)$ and an argument $a \in Arg$, the following questions can be formulated:

- Credulous Acceptance $Cred_\sigma$: Is a contained in some $E \in \sigma(F_A)$?
- Sceptical Acceptance $Scept_\sigma$: Is a contained in each $E \in \sigma(F_A)$?

Another problem is extension verification, which asks whether a set of arguments is an extension of the agent and, given the arguments Arg of the agent's AF, can be formulated as follows for a set $E \subseteq Arg$:

- Verification of an extension Ver_σ: Is $E \in \sigma(F_A)$?

As already explained in the background subsection on extensions, it may be that the stable semantics has no extension at all. For the other semantics, it may be of interest to know if there are extensions other than the empty set. Again, let the agent AF be given to formulate this into questions:

- Existence of an extension $Exists_\sigma$: Is $\sigma(F_A) \neq \emptyset$?
- Existence of a non-empty extension $Exists_\sigma^{\neg\emptyset}$: Does there exist a set $E \neq \emptyset$ such that $E \in \sigma(F_A)$?

As a final problem, Dvořák and Dunne state the decision whether a semantics has a unique extension for a given argumentation framework. Again, let the AF of the agent be given:

- Uniqueness of the solution $Unique_\sigma$: Is there a unique set $E \in \sigma(F_A)$, i.e., is $\sigma(F) = \{E\}$?

Another relevant reasoning problem, as listed e.g. in [16], is the question of how many extensions are produced by a semantics for an argumentation framework ($Count_\sigma = |\sigma(F)|$). This does not fall under the category of decision questions and thus gets omitted. However, a similar decision question can be formulated. Given the agent's AF:

- Actual count question $ActualCount_\sigma^n$: Is $|\sigma(F_A)| = n$?

With that, the semantic decision questions are determined. As previously introduced, no syntactical questions are allowed and therefore none are searched for.

At this point, as explained at the beginning of this subsection, the questions asked can be classified into different categories. The question categories result from the decision problems considered. Table 2 summarises the question categories for the question act of decision questions. However, since we formulated no questions for queries due to the scenario, just as non-extension related questions, as well as syntactic questions, are not allowed in this scenario, too, it is not possible to determine categories for them and thus shall be part of future work.

Question class	Question category	Decision questions
Semantic Questions	Acceptance	$Cred_\sigma$, $Scept_\sigma$
	Verification	Ver_σ
	Existence	$Exists_\sigma$, $Exists_\sigma^{\neg\emptyset}$
	Uniqueness	$Unique_\sigma$
	Counting	$ActualCount_\sigma^n$

Table 2: Grouping of the semantic decision questions into question categories.

The collection of questions an agent can answer is denoted by the set $\mathfrak{Q}$. For an agent who is only able to answer decision questions a subset $\mathfrak{Q}_{Dec}$ is used. Further, if the agent is only able to answer semantic questions, then this can be noted with a superscript $\mathfrak{Q}_{Dec}^{Sem}$. This results in a subset relationship $\mathfrak{Q}_{Dec}^{Sem} \subseteq \mathfrak{Q}_{Dec} \subseteq \mathfrak{Q}$.

The semantic questions are $\mathfrak{Q}_{Dec}^{Sem} = \{Cred_\sigma, Scept_\sigma, Ver_\sigma, Exists_\sigma, Exists_\sigma^{\neg\emptyset}, Unique_\sigma, ActualCount_\sigma^n\}$.

4.2.2. On the Complexity of Questions

Next, we evaluate the complexity of the selected questions for each semantics. This is important for making the elicitation algorithms as efficient as possible.

With the restriction to decision questions, this includes the reformulation of the counting question into a decision question, a far-reaching research basis for the complexity of decision problems can already be drawn on. Moreover, the choice of decision questions is analogous to the well-known computational problems for semantics is favourable, since here, in principle, all upper bounds on the complexities are already known for the classical semantics. Different authors contributed for different problems and (classical) semantics. The works were summarised in [26] by Dvorák and Dunne which can be used as an overview of the subject.

Brief outlines of the findings are presented in the following. In addition, Table 3 presents the complexity classes for the classical semantics. It is borrowed from [26].

σ	$Cred_\sigma$	$Scept_\sigma$	Ver_σ	$Exists_\sigma$	$Exists_\sigma^{\neg\emptyset}$	$Unique_\sigma$
cf	in L	trivial	in L	trivial	in L	in L
ad	NP-c	trivial	in L	trivial	NP-c	coNP-c
gr	P-c	P-c	P-c	trivial	in L	trivial
co	NP-c	P-c	in L	trivial	NP-c	coNP-c
st	NP-c	coNP-c	in L	NP-c	NP-c	DP-c
pr	NP-c	Π_2^P-c	coNP-c	trivial	NP-c	coNP-c

Table 3: Complexity of computational problems for AFs (cf. [26])

In his seminal paper [20], Dung already gave the first insights into the classical semantics. It follows from his considerations that $Scept_{ad}$, $Exists_{ad}$, $Exists_{pr}$ and $Exists_{grd}$ are trivial in their complexity. Their properties make this evident. Likewise, for the grounded semantics, the acceptance problems $Cred_{gr}$ and $Scept_{gr}$ are in P, and the verification problem Ver_{gr} is also in P. For the latter, Dvorák and Woltran [29] also showed that they are P-hard. From the properties in Dung's paper [20] it can be inferred for the grounded semantics that $Exists_{gr}^{\neg\emptyset}$ is in L. Furthermore, it can be inferred for Ver_{ad} and Ver_{stb} that they are in L. In general, the distinction between problems in L and those which are P-complete can be attributed to Dvorák et al. [27] respectively Dvorák and Woltran [29].

Furthermore, it can be concluded from the properties that it is sufficient for $Cred_{cf}$ to check whether the argument in question attacks itself, so it follows that this problem is in L. This efficient check carries over to Ver_{cf}. For $Scept_{cf}$ and $Exists_{cf}$ it can be deduced from the properties that they can be solved trivially, since, among other things, the empty set always exists since it is conflict-free.

Already in 1996 Dimopoulos and Torres [19] explored abstract argumentation by using graph theoretical structures as well as logic programs and default logic. Consequently, the following complexity statements can be deduced from this: $Scept_{st}$ and Ver_{pr} are coNP-complete. Furthermore, the following problems are complete for NP: $Cred_{ad}$, $Exists_{ad}^{\neg\emptyset}$ for the admissible semantics, $Exists_{co}^{\neg\emptyset}$ for the complete semantics, $Cred_{st}$, $Exists_{st}$ and $Exists_{st}^{\neg\emptyset}$ for the stable as well as $Cred_{pr}$ and $Exists_{pr}^{\neg\emptyset}$ for the preferred semantics.

Later, Dunne and Bench-Capon [22] could show for sceptical acceptance of preferred semantics $Scept_{pr}$ that this problem is Π_2^P-complete.

Also Coste-Marquis et al. [17] gave further insights into the complexity of complete semantics in their work on symmetric AFs. According to their research $Cred_{co}$ is NP-complete, $Scept_{co}$ P-complete, $Exists_{co}$ is trivial, too, and that Ver_{co} can be solved efficiently (as it is known today Ver_{co} is in L).

Finally, the complexity of uniqueness must be determined. In his technical report, Dvorák [25] gives the complexity classes for all classical semantics. The uniqueness problem $Unique_{gr}$ for the grounded semantics is trivial because one of its properties is that it always has exactly one extension. $Unique_{cf}$ is in L, while $Unique_{ad}$, $Unique_{co}$ and $Unique_{pr}$ are coNP-complete. The stable semantics stands out in that one of its properties is that it does not guarantee that an extension exists at all. This means that an extra check is necessary to determine whether an extension exists, and $Unique_{st}$ is therefore DP-complete among the randomised reductions.

This sets the upper bound of the complexity of all classical semantics and the decision problems or the questions to the agent. The remaining question to be examined is therefore the number of extensions. The question $ActualCount_{\sigma}^{n}$ takes a number n and checks whether $n = Count_{\sigma}$. $Count_{\sigma}$ is the counting problem for enumerating the extensions, i.e. how many extensions there are in the end.

Baroni et al. [3] studied the complexity of extension counting problems and showed that in the general case #P-completeness prevails. Nevertheless, they show that there are a few tractable cases.

4.3. Algorithms for Interviewing

The next step is to look at the algorithms to be developed. To do this, we first deal with the division of the computational complexity of the scenario and the selection of the agent type. After we have set these, we proceed to design the interviewing algorithms in the subsections.

Regarding the complexity there are two parts to the first stage. On the one hand the interviewer has to do calculations on what question to ask as well as on deciding whether to continue or halt as all extensions have been found. Another view on this is, that the interviewer does some preprocessing before asking the agent that acts as some kind of oracle. The response or result of the agent is then post-processed.

This two-part work also means that the complexity is not only made up of the interviewer's part but also that of the agent. Therefore before designing elicitation algorithms it is necessary to have a look on the agent. It is established that the agent answers truthfully such that the agent acts as an oracle. This is the first condition imposed on the agent which is meaningful not only to the design of the algorithm, but also on the overall complexity as no additional computations are required on e.g. the probability of the answer being true or not.

Next the agent is only able to answer decision questions with the restriction that i) there is a relation to extensions and ii) that they are only semantic, so the set $\mathfrak{Q}_{Dec}^{Sem}$ is used. As all questions apart from $ActualCount_{\sigma}^{n}$ are directly also computational decision problems in AFs results from studies of the complexity directly apply to them.

For the later implementation of the algorithms, additional properties are of interest, such as whether the agent has already enumerated the extensions or not. These have no influence on the scope of questions that the agent can answer, as they relate to the implementation of the calculation of the answers. Therefore, these properties are added as a subscript. In Section 5 on the experiments, this distinction is discussed again. In the following, however, it is assumed for the sake of illustration that the agent has no state, which means that every calculation of the answer is carried out by the agent without prior knowledge of the previous one and thus also that the agent cannot fall back on any enumeration in advance or during the interview round. We denote this with the previous restrictions as $Agent_{\neg\mathbb{E}}^{Dec,Sem}$.

One effect of this decision, however, is the complexity of the agent's calculation of answers. If an agent can already access previously calculated information, answers can be calculated more easily if necessary. For example, if the agent enumerated all extensions in advance, i.e. solved the computational problem $Enum_\sigma$ giving $Enum_\sigma(F) = \mathbb{E} = \sigma(F)$ and stored the result, then the complexity of the considered questions in $\mathfrak{Q}_{Dec}^{Sem}$ boils down to search problems in $\mathbb{E}$.

Second, the answering of the $ActualCount_\sigma^n$ question depends i) on whether the agent already computed $\mathbb{E}$, because then $|\mathbb{E}| = Count_\sigma(F)$ meaning counting is a by-product of enumeration, and ii) if an agent internally solves $ActualCount_\sigma^n$ by computing $Count_\sigma$ once and storing this result, then the computational complexity differs between the first time the question is asked and any further questions about the count.

Yet the behaviour of an $Agent_{\neg\mathbb{E}}^{Dec,Sem}$ is in line with the complexities considered in the previous section on the questions to be asked and their complexity. There it is also assumed that there is no prior knowledge, i.e. no state in a process, to answer the question. With regard to the development of the algorithms, it can thus be assumed that these upper bounds are applicable. In the experiments in Section 5, the effect on the runtimes for agents without state and with state is examined in more detail.

Figure 16 establishes the connection. $\mathcal{A}$ contains all possible types of an agent. An agent that always answers truthfully as an oracle is denoted as $Agent$. Within this type of agent, the one that can only answer decision questions $Agent^{Dec}$ is chosen. In the case of this thesis, an $Agent^{Dec,Sem}$ is chosen that can only answer questions related to semantics. Finally, there is the restriction that this agent $Agent_{\neg\mathbb{E}}^{Dec,Sem}$ can neither enumerate nor keep a count internally.

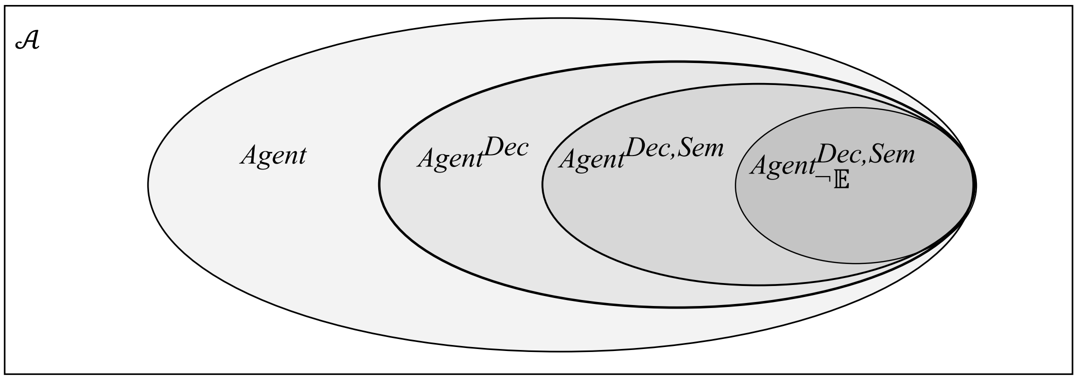

Figure 16: A visualisation of the narrowing of the agent.

At this point, the feasibility of asking questions to an $Agent_{\neg\mathbb{E}}^{Dec,Sem}$ with complexity classes such as NP, coNP and higher in PH should also be discussed. In general, it could be assumed that a few questions can be asked in these complexity classes without impacting the overall performance too much. In principle, there is a standard implementation for these questions and the identified questions in classes L or P are improvements of these. On top of that, the standard implementations of the questions respectively calculation tasks are often obviously inefficient. In addition, many of the questions can only be used meaningfully in iterations or even have iterations in their standard implementation. We specify these as follows (apart from $Exists_\sigma$ as it is only for $\sigma = st$ relevant):

- The standard procedure to $Cred_\sigma$ is to non-deterministically guess an extension containing the occurring argument. After that, it is verified and in case of answering yes, the argument is accepted credulously.
- For $Scept_\sigma$, on the other hand, it has to be shown that among all guessed extensions the argument is not accepted sceptically, i.e. the extension is verified and does not contain the argument to be checked. If the answer is no for all extensions guessed in this way, the argument is accepted sceptically.
- For $Exists_\sigma^{\neg\emptyset}$ a non-empty extension is to be guessed non-deterministically. If this is verified, then $Exists_\sigma^{\neg\emptyset}$ is to be answered yes.
- For $ActualCount_\sigma^n$, it is verified that the cardinality matches the given number. For this, all extensions must be enumerated. All extensions are to be verified and compared with the number n to be checked. If both are the same, the answer is yes.
- Finally, for $Unique$, all possible extensions must be verified (or chosen non-deterministically and verified). If only one is found, the question must be answered yes; if a second is found or none at all, the question must be answered no.

It is apparent that in these algorithms the questions must already (partially) verify the extensions. This makes their use within the algorithms to be developed questionable in terms of overall performance. We argue they are therefore not feasible to use in the algorithms[9].

Turning to the interviewer side, the complexity here consists of, on the one hand, the number of questions to be asked in order to arrive at the final extension set. On the other hand, the calculation of which question to ask next and the decision whether all extensions have been found must be included in the complexity. These considered components correspond to those in Figure 15.

[9]See also Appendix B where we additionally evaluated the performance with another kind of solver implementation using these questions in modified versions of the developed algorithms.

The complexity can only be determined together with the response complexities of the agent due to the cyclical process on the part of the interviewer. This is because the interviewer tries to find all extensions and therefore works through all the necessary steps to obtain a solution. However, since the interviewer cannot determine the answer to partial problems themselves, they ask the agent. The agent then calculates the answer to the question. With this result, the interviewer can ask for the next steps. Within this cycle, the complexity of the questions to the agent is decisive for the overall performance of the interviewing part of the elicitation process.

Based on this knowledge, algorithms for the classical semantics can now be described. For this purpose, on the one hand, the algorithm for the interview part per semantics will be outlined and, on the other hand, the generation of σ-equivalents AF with the collected information will be discussed.

4.3.1. Naïve Approach as a Baseline

Before an algorithm is actually shown for each classical semantics, we should establish a baseline first. This serves to compare the algorithms with an approach that works for all classical semantics. This approach is the naïve one from Section 2.6 and was already developed in the introductory paper by Kuhlmann [34] to show the overall feasibility of elicitation.

For the interviewing stage it relies on asking the agent whether a set of arguments is accepted, i.e. whether it represents an extension. This procedure corresponds to forming the power set 2^{Arg} for all known arguments Arg and asking the agent via Ver_σ for each set whether this is a valid extension.

Algorithm 3 Naïve Approach for Interviewing Part

```
Require: $Arg$ as a set of arguments, $\sigma$ as the semantics
Ensure: An extension set $\mathbb{S}$
  $\mathbb{S} \leftarrow \emptyset$
  $\mathcal{S} \leftarrow 2^{Arg}$
  for all $s \in \mathcal{S}$ do
      $r \leftarrow$ Ask agent $Ver_\sigma(s)$
      if $r = \top$ then
          $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$          ▷ Add $s$ to the extension set $\mathbb{S}$
```

The decisive disadvantage can be seen quickly. The number of extensions increases exponentially with the number of arguments and with them the number of questions to be asked. This procedure is an exhaustive search, also called the brute-force search. In this case, all possible extensions not only have to be tried out, they all have to be confirmed or rejected by the agent via the verification question. It is obvious that this is not an efficient procedure.

4.3.2. Conflict-Free Sets

We start with the examination of conflict-free sets. To do this, we first collect feasible questions and then describe and prove the algorithm. The questions and their complexity can be seen in Table 3.

Questions For conflict-free sets, the picture is that all questions are either in complexity class L or are even trivial. Since there is always a conflict-free set, namely the empty set, it is of interest whether there is another one besides the empty set. For $Exists_{cf}^{\neg\emptyset}$ it is sufficient to check whether an argument does not attack itself. A similar approach can be taken to answer $Cred_{cf}$. If an argument is not self-attacking, then it is accepted. Therefore, the question $Exists_{cf}^{\neg\emptyset}$ can also be replaced by checking all arguments for their credulous acceptability. If one is credulously accepted, then $Exists_{cf}^{\neg\emptyset}$ is to be answered in the positive. In the case that none is accepted, it is to be negated. The same applies to the question $Unique_{cf}$. Only if all arguments attack themselves, there is only one existence, i.e. the empty set. This can also be compensated by asking $Cred_{cf}$ for all arguments. In addition, Ver_{cf} is included in the question pool, since it is in L and is necessary in order to have a suspected extension confirmed by the agent respectively to have the extension rejected by the agent. Lastly, using $ActualCount_{\sigma}^{n}$ is not an option because of its default implementation by finding all extensions and counting them, this is also true for all following algorithms. The interested reader can check the results from supplementary experiments in Appendix B, which support this.

Algorithm 4 Conflict-free Sets Algorithm for the Interviewing Part

```
Require: $Arg$ as a set of arguments, $\sigma = cf$
Ensure: An extension set $\mathbb{S}$
  $Args_{\mathbb{S}} \leftarrow \emptyset$
  for all $a \in Arg$ do                                        ▷ Step I
      if $Cred_{cf}(a) = \top$ then
          $Args_{\mathbb{S}} \leftarrow Args_{\mathbb{S}} \cup \{a\}$
  $\mathbb{S} \leftarrow \{\{\}\}$
  $\mathcal{S} \leftarrow 2^{Args_{\mathbb{S}}} \setminus \mathbb{S}$
  for all $s \in \mathcal{S}$ do                                ▷ Step II
      if $\exists s' \in \mathbb{S} : s \subseteq s'$ then
          $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$
      else
          $r \leftarrow$ Ask agent $Ver_{cf}(s)$
          if $r = \top$ then
              $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$
```

Description For conflict-free sets, i.e. $\sigma = cf$, recall from Definition 3 that a set is conflict-free, if there is no attack between the arguments in this set. From the signature of conflict-freeness Σ_{cf} it is also known that it is downward-closed and tight. This reflects on the one hand the fact that if e.g. three arguments $\{A_1, A_2, A_3\}$ are accepted, then all their subsets will be too, e.g. $\{A_1, A_3\}$ as one possible subsets. Furthermore, this implies the fact that the empty set $\{\}$ is always conflict-free and therefore the extension-set contains at least the empty set. On the other hand tightness represents the condition, that a conflict-free set cannot contain arguments that are attacked by the set.

From the Subsection 2.4 on signatures and realisability it is known, that the maximal number of sets in a conflict-free extension-set is the same as the cardinality of sets of the power set of all arguments, i.e. $\Delta_{cf}(n) = 2^n$.

With this knowledge, the algorithm for finding conflict-free sets can be drawn. As for a stop condition, there is no meaningful one apart from having all possible extensions enumerated. However, the enumeration can be optimised as the empty set $\{\}$ is always a valid conflict-free set and therefore is verified upfront. In addition, for a found conflict-free set all the subsets can be accepted by the downward-closure condition.

In the case of conflict-free sets, tightness does not provide any benefit. This is due to the fact, that the extension-set $\mathbb{S}$ can be checked for tightness, but this demands that all extensions already have been found as no further property restricts the possible extensions[10].

Up to this point the described algorithm uses only the Ver_{cf} question, similar to the naïve approach. As Ver_{cf} has an upper bound in L and $Cred_{cf}$, too, it is feasible to ask for all known arguments if they are credulously accepted. In case there is any self-attacking argument, it won't be in any conflict-free set and therefore the number of arguments for the power set reduces, i.e. $n = |\{a \mid Cred_{cf}(a) = \top, a \in Arg\}|$.

As for implementing Algorithm 4, on a deterministic computer it is a good idea to start with the superset of all other sets in the power set, that is the set where every other set is a subset, and then testing it and all subsets of it. This top-down approach makes heavy use of the downward-closure condition as for each found set, all the subsets can be accepted.

Example 17. *Let the agent have an hidden AF* $F_A = (\{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}, \{(A_1, A_3), (A_2, A_3), (A_3, A_4), (A_3, A_5), (A_4, A_5), (A_5, A_4), (A_4, A_6), (A_5, A_7)\})$. *This AF is identical to the AF* F_2 *in Figure 5 used in several examples.*

With $\sigma = cf$ *and* $\sigma(F_A) = \{\{\}, \{A_1\}, \{A_2\}, \{A_3\}, \{A_4\}, \{A_5\}, \{A_6\}, \{A_7\},$
$\{A_1, A_2\}, \{A_1, A_4\}, \{A_1, A_5\}, \{A_2, A_4\}, \{A_1, A_6\}, \{A_2, A_5\}, \{A_1, A_7\}, \{A_2, A_6\},$
$\{A_2, A_7\}, \{A_3, A_6\}, \{A_3, A_7\}, \{A_4, A_7\}, \{A_5, A_6\}, \{A_6, A_7\},$
$\{A_1, A_2, A_4\}, \{A_1, A_2, A_5\}, \{A_1, A_2, A_6\}, \{A_1, A_2, A_7\}, \{A_1, A_4, A_7\}, \{A_1, A_5, A_6\},$
$\{A_2, A_4, A_7\}, \{A_2, A_5, A_6\}, \{A_1, A_6, A_7\}, \{A_2, A_6, A_7\}, \{A_3, A_6, A_7\}, \{A_1, A_2, A_4, A_7\},$
$\{A_1, A_2, A_5, A_6\}, \{A_1, A_2, A_6, A_7\}\}$. *The interviewer only knows* Arg *and* σ.

First, the interviewer asks the agent about the credulous acceptance of all arguments and

[10] E.g. the stable semantics are tight and incomparable and therefore it is possible to make use of the tightness property, since if s is verified and $a \notin s$, then there will be no $s \cup \{a\}$.

collects these in the set $Args_{\mathcal{S}} = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$. It can be seen that no argument is missing, as none of the arguments attack themselves.

Next, the empty set $\{\}$ is added to $\mathbb{S}$, since it is always acceptable (the same could be done with all credulously accepted arguments, but is omitted from the algorithm for simplicity).

Now all possible extensions are tried out. For reasons of comprehensibility, not all possible extensions are shown, but only two positive examples and one negative example.

Let us start with a positive example. Let the set $s = \{A_1, A_2, A_4, A_7\}$ be given and $\mathbb{S}$ contains only the empty set. Since $s \nsubseteq \{\}$, the second case applies and the agent verifies s as an extension. Therefore s is added to $\mathbb{S}$.

Now we suppose a negative example. For this $s = \{A_1, A_2, A_3, A4\}$. $\mathbb{S}$ contains the previously found extension and the empty set. So it has to be checked whether s is a subset of a set in $\mathbb{S}$. This is obviously not the case, since $s \nsubseteq \{\}$ and $s \nsubseteq \{A_1, A_2, A_4, A_7\}$. Since the agent also rejects $Ver_{cf}(s)$, s is not in $\mathbb{S}$.

Finally, the second positive example. For this we take the set $s = \{A_1, A_4, A_7\}$. $\mathbb{S}$ contains the previously found extension and the empty set. Again, it is necessary to check whether s is a subset. This is the case since $s \subseteq \{A_1, A_2, A_4, A_7\}$ and therefore s can be added to the extension set without asking the agent for verification.

We conclude this example with these results. For all other possible sets in $\mathcal{S}$, proceed in the same way. In the end, after all possible sets have been tried, the extension set is equal to the result of the semantic function using cf for the AF F_A, i.e. $\mathbb{S} = \sigma(F_A)$.

Proof of Corectness for Algorithm 4 In order to prove the correctness, the individual steps are to be proven. The steps can be taken from the comments in Algorithm 4.

Proof of step I. Let Arg be all arguments of the agent's AF and σ the semantics and let us prove, that $Args_{\mathcal{S}}$ only contains credulously accepted arguments. Asking the agent $\forall a \in Arg : Cred_{\sigma}(a)$ by definition gives for each argument whether it is credulously accepted or not. Collecting those where the question is positive results in $Args_{\mathcal{S}}$ containing only credulously accepted arguments by definition. □

Proof of step II. Let $Args_{\mathcal{S}}$ be a set of all credulously accepted arguments, $\mathbb{S}$ the extension set, $\mathcal{S}$ the power set of $Args_{\mathcal{S}}$ and F_A be the hidden AF of the agent.

As the step II is iterative, we will give a proof of correctness by a loop invariant and induction.

Invariant: At the start of each iteration of $\mathcal{S}$, $\mathbb{S}$ should contain only those extensions from $\mathcal{S}$ such that $\mathbb{S}$ is a subset of the extension-set of the agent's AF.

Initialisation: At the start of the first iteration the invariant states that $\mathbb{S}$ should contain only those extensions from $\mathcal{S}$, where at this point no extension has been chosen from, such that $\mathbb{S}$, which contains only the empty set at this point, is a subset of the extension-set of the agent's AF. As $\mathbb{S} = \{\{\}\}$ and $\{\{\}\} \subseteq \sigma(F_A)$ with F_A being the AF of the agent is always true by Definition 3, this holds.

Maintenance: Assume that the invariant holds at choosing $s \in \mathcal{S}$. Then it must be that $\mathbb{S}$ must only contain extensions that the agent would verify to be part of their AF, too.

So for adding s to the extension-set $\mathbb{S}$ the following must be valid.

If there exists a set s' in $\mathbb{S}$ such that s is a subset of s' then s is also a valid extension. This is correct because Theorem 1 states that the extension-set is downward-closed so that Definition 18 and Definition 17 hold. That is, for any $\mathbb{S} \in \Sigma_{cf}$ it holds that $\{S' \subseteq S \mid S \in \mathbb{S}\}$. Therefore s can be added in this case.

In the case that s is not a subset, the agent must verify the extension and in the positive case, it is added to the extension set $\mathbb{S}$. This holds because the agent always answers Ver_{cf} truthfully.

Termination: When the loop terminates, $\mathbb{S} = \sigma(F_A)$ where F_A is the AF of the agent. The loop terminates when there is no remaining set of arguments in $\mathcal{S}$ to inspect. Since $\mathcal{S}$ is finite, this is trivial. $\square$

4.3.3. Admissible Sets

Now that an algorithm for the conflict-free sets has been developed, one for the admissible sets follows. Here, too, we first find questions that are tractable for an $Agent_{\neg\mathbb{E}}^{dec,sem}$ and develop the algorithm based on this. A proof follows as before.

Questions For admissible sets, according to Table 3, only the question Ver_{ad} is tractable, since it lies in complexity class L. $Scept_{ad}$ and $Exists_{ad}$ are trivial due to the properties of $\sigma = ad$ (cf. also Definition 5). The verification can be implemented by the agent by checking the arguments. The necessary check corresponds to that from the Definition 5. Since each argument to the extension must also be accepted, it can simply be checked for each argument of the set of arguments to be checked whether this argument is accepted, i.e. each attacker is in turn attacked by another argument.

Other useful questions like $Cred_{ad}$, $Exists_{ad}^{\neg\emptyset}$ and $Unique_{ad}$ are in NP or coNP. Asking large numbers of these questions is therefore probably detrimental to performance (cf. experimental results in Section 5), which is why we do not include them in the basic algorithm. Nevertheless, a modified algorithm using these questions is presented in Appendix A, and in Appendix B we briefly review the performance using additionally another solver.

Description The semantics of admissibility has few properties. With respect to its signature Σ_{ad}, it can be inferred from Theorem 2 that the empty set always forms an extension and that all extensions are conflict-sensitive to each other.

First is the verification of whether s is in the extension set $\mathbb{S}$, the agent can verify s and in the case that s could not only theoretically be part but actually forms an extension, it can be included in the extension set.

Algorithm 5 Admissible Sets Algorithm for the Interviewing Part

```
Require: $Arg$ as a set of arguments, $\sigma$ as the semantics
Ensure: An extension set $\mathbb{S}$
  $Args_{\mathcal{S}} \leftarrow Arg$
  $\mathbb{S} \leftarrow \{\{\}\}$                                                    ▷ Step I
  $Pairs_{\mathbb{S}} \leftarrow \{\}$
  $\mathcal{S} \leftarrow 2^{Args_{\mathcal{S}}} \setminus \mathbb{S}$
  for all $s \in \mathcal{S}$ do
                                                                        ▷ Step II
      if $s \in \mathbb{S}$ then
          continue
                                                                        ▷ Step III
      $r \leftarrow$ Ask agent $Ver_{\sigma}(s)$
      if $r = \top$ then
          $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$                  ▷ Add $s$ to the extension set $\mathbb{S}$
          for all $p \in s \times s$ do
              $Pairs_{\mathbb{S}} \leftarrow Pairs_{\mathbb{S}} \cup \{p\}$   ▷ Add pairs of arguments appearing together

          for all $S \in \mathbb{S}$ do                                 ▷ Step IV
              if $S \cup s \notin \mathbb{S}$ then
                  $c \leftarrow \top$
                  if $\forall a, b \in S \cup s : (a, b) \in Pairs_{\mathbb{S}}$ then
                      $c \leftarrow \bot$
                  if $c \neq \top$ then                 ▷ No conflict means potential for existing
                      if $Ver_{ad}(S \cup s)$ then
                          $\mathbb{S} \leftarrow \mathbb{S} \cup \{S \cup s\}$
```

Once this is the case, the latter property of conflict-sensitivity can be used as a criterion in advance to check whether a set of arguments to be checked, s, has further compatible extensions. For this, for all arguments that exist in the union of s and the extension set $\mathbb{S}$, there is a pair that is in $Pairs_{\mathbb{S}}$. If this is the case, one can let the agent verify if the union set is indeed a valid extension[11].

In the case of admissibility, we make no use of the diversity function as a stop condition, since $\Delta_{ad} = 2^{|Args_{\mathcal{S}}|}$, i.e. the upper limit corresponds to the size of the search space of extensions.

Example 18. *We also give an example for the algorithm for admissible sets. For this we again use AF $F_2 = F_A$ from Figure 5. The hidden AF of the agent is $F_A = (\{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}, \{(A_1, A_3), (A_2, A_3), (A_3, A_4), (A_3, A_5), (A_4, A_5), (A_5, A_4), (A_4, A_6), (A_5, A_7)\})$ and $\sigma = ad$.*

[11]The disadvantage of this method is that some extensions cannot be found in advance at an early stage because the necessary pairs have not yet been formed. This is because $\mathbb{S}$ is still incomplete and grows with the iterations.

For $\sigma(F_A)$ we get the following extensions-set $\{\{\}, \{A_1\}, \{A_2\}, \{A_1, A_4\}, \{A_1, A_5\}, \{A_2, A_4\}, \{A_1, A_2\}, \{A_2, A_5\}, \{A_1, A_2, A_4, A_7\}, \{A_1, A_2, A_5, A_6\}, \{A_1, A_2, A_4\}, \{A_1, A_2, A_5\}, \{A_1, A_4, A_7\}, \{A_1, A_5, A_6\}, \{A_2, A_4, A_7\}, \{A_2, A_5, A_6\}\}$. The interviewer only knows Arg and σ. Due to the number of extensions, we will only select a few examples to demonstrate how the algorithm works.

The algorithm first accepts $\{\}$ and then forms the power set without the empty set. We start with the first iteration with the set of arguments $s = \{A_1\}$. Step II is skipped because $\mathbb{S}$ is still empty. The agent verifies $s = \{A_1\}$ and we add this to $\mathbb{S}$ so that $\mathbb{S} = \{\{A_1\}\}$. We also form the pair $Pairs_{\mathbb{S}} = \{(A_1, A_1)\}$.

We proceed and choose $s = \{A_2\}$. Since $\mathbb{S} = \{\{A_1\}\}$, step II is performed, but $s \notin \mathbb{S}$. Therefore we continue with step III. Here the agent verifies first that $s \in \mathbb{S}$. As this is the cas ewe get $\mathbb{S} = \{\{A_1\}, \{A_2\}\}$. For $Pairs_{\mathbb{S}}$ we now get $\{(A_1, A_1), (A_2, A_2)\}$. We proceed with step IV. Here $\{A_1\} \cup \{A_2\}$ do not have no pairs for $\{(A_1, A_2), (A_1, A_2)\}$, so we do not add it for the moment. However, there will be an invariant where $s = \{A_1, A_2\} \in \mathcal{S}$ and then it will get verified by the agent.

Finally, we show a negative example. Suppose $s = \{A_3\}$. We do not find $s \in \mathbb{S}$ as well as that the agent does not verify s to be an extension. In this case, we do not add to $\mathbb{S}$ and also do not perform step IV.

Proof of Correctness for Algorithm 5 In order to prove the correctness, the individual steps are to be proven. The steps can be taken from the comments in Algorithm 5.

Proof of step I. Adding the empty set $\{\}$ (or $\emptyset$) is trivial since this is already proven by, among other things, the Theorem 2. □

Proof of the main part. Let $Args_{\mathcal{S}}$ be a set of all arguments, $\mathbb{S}$ the extension set, $\mathcal{S}$ the power set of $Args_{\mathcal{S}}$ without the empty set and F_A be the hidden AF of the agent.

As the main part is iterative, we will give a proof of correctness by a loop invariant and induction. The steps II, III and IV within the loop maintenance and the termination are considered.

Invariant: At the start of each iteration of $\mathcal{S}$, $\mathbb{S}$ should contain only those extensions from $\mathcal{S}$ such that $\mathbb{S}$ is a subset of the extension-set of the agent's AF.

Initialisation: At the start of the first iteration the invariant states that $\mathbb{S}$ should contain only those extensions from $\mathcal{S}$, where at this point no extension has been chosen from, such that $\mathbb{S}$, which contains only the empty set as an extension at this point, is a subset of the extension-set of the agent's AF. As $\mathbb{S} = \{\{\}\}$ and $\{\{\}\} \subseteq \sigma(F_A)$ with F_A being the AF of the agent is always the case according to Theorem 2, this holds.

Maintenance: Assume that the invariant holds at choosing $s \in \mathcal{S}$. Then it must be that $\mathbb{S}$ must only contain extensions that the agent would verify to be part of their AF, too.

Step II: Starting with step II, it is checked whether $s \in \mathbb{S}$ exists. If this condition is fulfilled, e.g. because previously for an s' the condition existed that $S \in \mathbb{S} : Ver_{ad}(S \cup s') = \top$ and $S \cup s' = s$, so intuitively speaking s could already be formed from a union set before, then s need not be checked further.

It is easy to show that this is correct. At a point in time T_0, $\mathbb{S}_{T_0} = \{\{A_0\}, \{A_1, A_2\}, \{A_3\}\}$ is given. Since $s_0 = \{A_1, A_2\}$ is already given, it does not need to be checked. Now, if $s_1 = \{A_1, A_3\}$ occurs at some later point in time T_1, then $s_1 \notin \mathbb{S}_{T_1}$ with $\mathbb{S}_{T_1} = \mathbb{S}_{T_0}$. Therefore s_1 needs to be checked. This corresponds to the condition of step II.

Step III: Next comes step III. The question Ver_{ad} holds by definition, so that if the agent s answers in the affirmative, it is added to the extension set $\mathbb{S}$ and the pairs are formed from all the arguments, which are then added to $Pairs_{\mathbb{S}}$.

Step IV: We continue with step IV only if the case in step III applies. Here we check whether we can already form further extensions from both the extension set $\mathbb{S}$ and the extension s to be checked, which the agent does not have to verify further. The case occurs when there is no conflict for each extension in $\mathbb{S}$ together with the extension to be verified. The contrary case is given in the Definition 20 for conflict-sensitivity.

Therefore, for each $S \in \mathbb{S}$, we check whether $S \cup s$ does not yet occur in $\mathbb{S}$. If this is the case, this union set should be examined more closely. In the case that for all constructible pairs (a, b) from $a, b \in S \cup s$ there is also a pair in $Pairs_{\mathbb{S}}$, there is no conflict. That is, the arguments could occur together. This case is verified by Ver_{ad} and, if verified positive, $S \cup s$ is included in $\mathbb{S}$. No pairs need to be formed here, as they all already exist.

That this step works can be shown by means of Definition 20. It states that if $A \cup B \notin \mathbb{S}$ for $A, B \in \mathbb{S}$, then $\exists a, b \in A \cup B : (a, b) \notin Pairs_{\mathbb{S}}$. We assume $A = s$ and $S \in \mathbb{S} : B = S$, then in order for conflict-sensitivity to be preserved, $S \in \mathbb{S}$ are in principle admissible for all $S \cup s$ only if $\forall a, b \in S \cup s : \nexists (a, b) \notin Pairs_{\mathbb{S}}$. If such a case exists for an $S \in \mathbb{S}$, it is worthwhile to have the agent $S \cup s$ verified. Verification holds by definition.

Suppose the converse, that $S \cup s \notin \mathbb{S}$ but $\forall a, b \in S \cup s : (a, b) \in Pairs_{\mathbb{S}}$ holds. This would be caught by verification so that no false extensions are included in $\mathbb{S}$.

The invariant is finished as there is no further step.

Termination: When the loop terminates, $\mathbb{S} = \sigma(F_A)$ where F_A is the AF of the agent. The loop terminates when there is no remaining set of arguments in $\mathcal{S}$ to inspect. Since $\mathcal{S}$ is finite, this is trivial. □

4.3.4. Complete Semantics

We continue with the first classical semantics in the narrower sense. The complete semantics is especially of interest as the grounded and the preferred semantics, among others, build on the complete semantics, as shown in the Figure 7 at the end of Subsection 2.2 on extension-based semantics.

As before, we start by selecting questions based on complexity as shown in Table 3. This is followed by a description of the Algorithm 6. The algorithm was created using the previously selected questions. Afterwards, a proof of correctness follows.

Questions For the complete semantics it can be seen from Table 3 that only $Scept_{co}$ and Ver_{co} are in P and L respectively as complexity class. For $Scept_{co}$ it can be said that it is in P, since it is only necessary to check whether an argument is in the grounded semantics, which has $Cred_{gr}$ in P. For Ver_{co}, on the other hand, it is sufficient for the agent to check whether the set E to be checked is conflict-free and whether the characteristic function applied to that set E also has E as its result, i.e. $\Gamma_{F_A}(E) = E$ with the agent's AF F_A. This makes E admissible and ensures that any argument accepted in E is also in E.

With the assumption made earlier that large numbers of questions in classes outside L and NP are not beneficial for the runtime, further questions are not considered in this basic version of the algorithm. A modified version with the additional questions is briefly presented in Appendix A.

Description The complete semantics is special in the respect that no precise characterisation and thus signature has been found for it up to the present. I.e. the properties found can lead to an extension-set being com-closed and also satisfying $(\bigcap_{S \in \mathbb{S}} S) \in \mathbb{S}$, but yet no AF F exists for which $co(F) = \mathbb{S}$ (see also example 8 in [23]).

Nevertheless, at least some unwanted extensions can be excluded and thus the number of questions that the agent has to answer can be presumably reduced. For this purpose, it is verified at the beginning of the algorithm whether the empty extension exists. The background is that if it exists, the intersection over all arguments in the extensions is empty[12].

On the other hand, if the empty extension is not part of the extension set, then after two sets have been found, the intersection can be formed, thus creating a condition according to which unwanted extensions can be sorted out. For example, if $\mathbb{S}$ is $\{\{A_1, A_2, A_4\}, \{A_4\}\}$, a set of arguments such as $\{A_1, A_3\}$ need not be checked, since the argument A_4 is the current intersection of all extensions in $\mathbb{S}$ and the intersection of this set $\{A_4\}$ with the set to be checked, $\{A_1, A_3\}$, is no non-empty subset.

[12]This also means that the unique grounded extension is empty.

Algorithm 6 Complete Semantics Algorithm for the Interviewing Part

```
Require: $Arg$ as a set of arguments, $\sigma = co$
Ensure: An extension set $\mathbb{S}$
  $Args_{\mathcal{S}} \leftarrow Arg$
  $\mathbb{S} \leftarrow \{\}$
  $z \leftarrow$ Ask agent $Ver_{co}(\{\})$                                ▷ Step I
  if $z = \top$ then
      $\mathbb{S} \leftarrow \mathbb{S} \cup \{\{\}\}$
  $e \leftarrow \emptyset$
  $\mathcal{S} \leftarrow 2^{Args_{\mathcal{S}}} \setminus \{\{\}\}$
  for all $s \in \mathcal{S}$ do
      if $\Delta_{co}(|Args_{\mathcal{S}}|) = |\mathbb{S}|$ then                   ▷ Step II
          break
      if $z = \bot \wedge |\mathbb{S}| \geq 2 \wedge |(s \cap e)| \neq 0 \wedge (s \cap e) \nsubseteq e$ then   ▷ Step III
          continue                     ▷ Intersection of all arguments not given
      $r \leftarrow$ Ask agent $Ver_{co}(s)$                                ▷ Step IV
      if $r = \top$ then
          $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$
          if $z = \bot \wedge |\mathbb{S}| \geq 2$ then                          ▷ Step V
              $e \leftarrow \bigcap_{S' \in \mathbb{S}} S'$           ▷ Find the intersecting arguments
```

Unfortunately, the check for com-closed is not meaningful to check within an invariant, because all further possible extensions are missing. Thus, it is not possible to tell whether two arguments from $\mathbb{S} \cup s$, with s the set of arguments to be checked, actually forms a unique completion set (cf. Definition 22).

In addition, the diversity function is again used as an upper limit for the number of extensions. It therefore serves as a stop condition.

Finally, the omission of the question $Scept_{co}$ may be noticeable. In fact, this has to do with the fact that knowing about sceptically accepted arguments does not reduce the set of arguments of the search space $Args_{\mathcal{S}}$. Moreover, additional complexity is added for checking whether an argument has already been accepted sceptically or not. This version of the algorithm therefore omits this.

Example 19. *For the current example, let us consider repeatedly the AF F_2 from Figure 5. For $\sigma = co$ and $F_2 = F_A$ we get here $\sigma(F_A) = \{\{A_1, A_2\}, \{A_1, A_2, A_5, A_6\}, \{A_1, A_2, A_4, A_7\}\}$ with $Arg = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$. We show three positive examples and two negative examples below.*

The algorithm starts with step I by verifying the empty set. Here, the agent answers negatively since $\{\} \notin gr(F_A)$. Therefore, it is not added to $\mathbb{S}$. We note this result in the variable z. Then we iterate over the power set without the empty set.

Let us start with a positive example for $s = \{A_1, A_2, A_5, A_6\}$. Since we have the upper bound at $\Delta_{gr}(|Arg|) = 9$, step II does not stop us. For step III, $z = \bot$, however, the *cardinality of the extension set is at $|\mathbb{S}| = 0$, thus this check is also to be ignored. We ask the agent for $Ver_{co}(s)$ and the agent accepts this. We add s to $\mathbb{S}$. We do not perform step V because $|\mathbb{S}| = 1$ and the condition is not satisfied.*

We continue with $s = \{A_1, A_2\}$. Again, the stop condition is not fulfilled. The condition of step III still does not apply either. So we let the agent verify s. Since it decides positively, we add s to $\mathbb{S}$. Now that $z = \perp$ and $|\mathbb{S}| = 2$ are satisfied, we perform step V. We form the intersection of $\{A_1, A_2, A_4, A_7\} \cap \{A_1, A_2\}$ and obtain $e = \{A_1, A_2\}$.

Next, two negative examples are to follow with the first one being $s = \{A_1, A_3\}$. Again, the stop condition does not apply, but step III does, since $z = \perp$ and $\mathbb{S} \geq 2$. Now we have to check whether s is a superset of $e = \{A_1, A_2\}$. Since this is not the case, we proceed to the next possible extension.

Now we take $s = \{A_1, A_2, A_3\}$. Again, the stopping condition does not apply, but step III needs to be examined more closely because $z = \perp$ and $\mathbb{S} \geq 2$. s is indeed a superset of e, so we proceed to step VI. However, since the agent does not verify s, we do not add it to the extension set $\mathbb{S}$.

Let us conclude this example with a positive example. Let $s = \{A_1, A_2, A_5, A_6\}$. Again, the stop condition does not apply, but once more, step III. We note that $s \supseteq e$, so the agent can verify s. He confirms s and so we add this to $\mathbb{S}$. The condition from step V is not satisfied because of $|\mathbb{S}| > 2$.

Once all possible extensions have been tried, we again get $\mathbb{S} = co(F_A) = \{\{A_1, A_2\}, \{A_1, A_2, A_5, A_6\}, \{A_1, A_2, A_4, A_7\}\}$ in this example.

Proof of Correctness for Algorithm 6 In order to prove the correctness, the individual steps are to be proven. The steps can be taken from the comments in Algorithm 6.

Proof of step I. Let Ver_{co} be the computational problem to verify that a given set of arguments is an extension of the agent's AF. As Ver_{co} holds by definition, this step is trivial. □

Proof of the main part. Let $Args_{\mathcal{S}}$ be a set of all arguments, $\mathbb{S}$ the extension set, $\mathcal{S}$ the power set of $Args_{\mathcal{S}}$ without the empty set and F_A be the hidden AF of the agent.

As the main part is iterative, we will give a proof of correctness by a loop invariant and induction. All steps II to V within the invariant and the termination are considered.

Invariant: At the start of each iteration of $\mathcal{S}$, $\mathbb{S}$ should contain only those extensions from $\mathcal{S}$ such that $\mathbb{S}$ is a subset of the extension-set of the agent's AF.

Initialisation: At the start of the first iteration the invariant states that $\mathbb{S}$ should contain only those extensions from $\mathcal{S}$, where at this point no extension or the empty set has been chosen from, such that $\mathbb{S}$, which contains no extension or the empty set at this point, is a subset of the extension-set of the agent's AF. As $\mathbb{S} = \{\}$ and

$\{\} \subseteq \sigma(F_A)$ or $\mathbb{S} = \{\{\}\}$ and $Ver_{co}(\{\}) = \top$ with F_A being the AF of the agent, this holds.

Maintenance: Assume that the invariant holds at choosing $s \in \mathcal{S}$. Then it must be that $\mathbb{S}$ must only contain extensions that the agent would verify to be part of their AF, too.

We start with step III, since step II is a termination condition and is covered in the termination part.

Step III: Given e which holds a set of arguments, s which holds the chosen set of arguments to be checked, $\mathbb{S}$ which holds the current state of the extension set and z which is either true or false and contains the result of $Ver_{co}(\{\})$. In the case that the empty set is not part of the extension set and two or more extensions have already been found, then $s \cap e$ must be a subset of s and not empty, otherwise s can be discarded. To prove this, we assume the contrary, that $s \cap e$ is not a subset of e, but s is a valid extension and the empty set is not part of the extension set. Moreover, two extensions S' and S'' have already been found, so that $\{S', S''\} \subseteq \mathbb{S}$. In this case $e = (\bigcap_{S \in \mathbb{S}} S)$. In order that $\mathbb{S} \cup \{s\}$ is valid, however, there must be no intersection between $s \cap e$ and e. But since $\{e\} \subseteq co(F_A)$, this would mean that there must be $e = \{\}$ so that there is no intersection. But this is a contradiction, since the condition of step III only comes into play if $\{\} \notin \mathbb{S}$ respectively $\{\} \notin co(F_A)$.

Step IV: We continue with step IV. Here we ask the agent whether the set s to be checked is indeed an extension after all the previous criteria have been met. If this is the case, we add the extension to the extension set.

Finally, in case the empty set is not part of the extension set and we have now found at least two extensions, in step V we form the intersection that must be the superset for the intersection of itself and all further possible extensions.

Step V: For step V, let the extension set be $\mathbb{S} = \{S', S''\}$ and the intersection of it be $(\bigcap_{S \in \mathbb{S}} S) = I$. I corresponds to the grounded extension as it is the unique minimal extension with respect to set inclusion. So the intersection of S' and S'' is the empty set, if and only if there are no common arguments in both sets S' and S'', but then this contradicts the condition that $\{\} \notin \mathbb{S}$. Next we show that for a third $E = S' \cup \{a\}$ it still holds $I = (\bigcap_{S \in \mathbb{S} \cup \{E\}} S)$. As I has to be minimal, if $a \in E$ is also in I, then a is also in S' and S'', therefore it would have been in I, too, such that $I = (\bigcap_{S \in \mathbb{S} \cup \{E\}} S)$ holds. The same goes for more than three sets, too.

Termination: When the loop terminates, $\mathbb{S} = \sigma(F_A)$ with F_A being the AF of the agent. The loop, therefore, terminates in two cases:

1. if there is no set of arguments in $\mathcal{S}$ is left to be checked or
2. if the theoretical maximal number of extensions has been found.

The first case is trivial. The second case holds by the Definition 27 as well as Theorem 5. □

4.3.5. Grounded Semantics

Next, the grounded semantics should be addressed. It too is based on the admissibility of accepted arguments. Its special property, however, is that it is, according to Definition 9, the minimum complete extension with respect to set inclusion.

While this property is in principle only helpful if the complete extensions are known (and in the case of this Master's thesis the agent only answers to one semantics), it can be deduced that there is only one unique grounded extension. Expressed differently, it describes that $|gr(F)| = 1$ for any AF F. This simplifies the search for the extension set $\mathbb{S}$, since there must always be exactly one extension (cf. also Theorem 3).

Also with this semantics, we first check which questions can be used to then describe the Algorithm 7 and give a proof of correctness at the end.

Questions Similar to the conflict-free sets, all questions with reference to grounded semantics are also tractable. These are in the complexity classes L and P respectively or are even entirely trivial. Due to the already known property of forming only one extension, the question $Cred_{gr}$ is very suitable for finding the possible arguments of the extension. The implementation corresponds to calculating the least-fix point of the characteristic function and then checking there whether the argument occurs and is thus accepted. $Scept_{gr}$ can be implemented analogously. For Ver_{gr}, the least-fix point of the characteristic function must be compared with the set of arguments to be verified.

The question $Exists_{gr}^{\neg\emptyset}$ is also interesting, because it is suitable to check quickly and efficiently whether the extension contains any arguments at all and whether it is worthwhile to continue searching. To do this, one checks whether there is at least one argument that is not attacked by other arguments.

Description The algorithm starts by asking for the existence of a non-empty extension, i.e. $Exists_{gr}^{\neg\emptyset}$. If this is answered in the positive, the corresponding arguments can be searched for, in order to find the extension set $\mathbb{S}$.

For this, it is sufficient to ask for the acceptance of each argument. The question category of acceptance contains two decision questions, whereby the choice between the questions $Cred_{gr}$ and $Scept_{gr}$ is a matter of personal preference since both are sufficient in the following algorithm as each accepted argument can only be contained in the single extension.

If all arguments are found, the extension E formed from them is assigned to the extension set $\mathbb{S}$. This also applies if the extension is empty.

Example 20. *Once again we take the example from Figure 5 where $F_2 = F_A$ is assumed. For $\sigma = gr$ we get $\sigma(F_A) = \{\{A_1, A_2\}\}$. We now show by way of example how this* *extension can be elicited.*

To do this, the algorithm first checks $Exists_{gr}^{\neg\emptyset}$ which is answered positively by the agent so that it enters the loop body. For the arguments $Arg = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$, credulous acceptance is now tested. In the case of arguments A_1 and A_2, the agent accepts them and we can add them to E so that $E = \{A_1, A_2\}$. All other arguments are not accepted by the agent, so they are not added to E.

Finally, E is added to $\mathbb{S}$ so that $\mathbb{S} = \sigma(F_A)$ for $\sigma = gr$.

Algorithm 7 Grounded Semantics Algorithm for the Interviewing Part

```
Require: $Arg$ as a set of arguments, $\sigma = gr$
Ensure: An extension set $\mathbb{S}$
  $E \leftarrow \emptyset$
  if $Exists_{gr}^{\neg\emptyset} = \top$ then
      for all $a \in Arg$ do
          $r \leftarrow$ Ask agent $Cred_\sigma(a)$
          if $r = \top$ then
              $E \leftarrow E \cup \{a\}$                ▷ Add argument $a$ to the extension $E$
  $\mathbb{S} \leftarrow \{E\}$
```

Proof of Correctness for Algorithm 7

Proof. Proving correctness is trivial because first the agent is asked for $Exists_{gr}^{\neg\emptyset}$. This is valid by definition. There are two cases: i) If only an empty extension exists, skip the loop. ii) Run the loop because the extension is not empty.

The loop iterates over all arguments and asks for acceptance of the argument by $Cred_{gr}$. Again, this holds by definition. A formal proof of correctness by using a loop invariant is therefore omitted.

At the end, the extension is assigned to the extension set. This also holds by definition, since $\sigma = gr$ has only $|\mathbb{S}| = 1$. □

4.3.6. Preferred Semantics

The second to last classical semantics is the preferred semantics for which an algorithm has to be developed. We start again by finding suitable questions in order to describe the Algorithm 8. Afterwards, we proceed with proving the algorithm.

Questions The questions of the preferred semantics are all in NP, coNP or even higher in the polynomial hierarchy, as indicated in Table 3. That is, no tractable questions can be found for these semantics in the case of an $Agent_{\neg\mathbb{E}}^{dec,sem}$.

We therefore select only the question Ver_{pr}, which is in the complexity class coNP and is complete for it. The question can be implemented with the understanding of the Definition 10 as follows:

For the verification, the agent can first check the extension passed with the question respectively the set of arguments E to be checked for their admissibility. If this is the case, a superset of E can be guessed. If this superset is also admissible, then E cannot be a preferred extension, since E is not maximal with respect to set inclusion. If no superset can be found that is also admissible, then E is an extension of the agent's AF.

Algorithm 8 Preferred Semantics Algorithm for the Interviewing Part

```
Require: $Arg$ as a set of arguments, $\sigma$ as the semantics
Ensure: An extension set $\mathbb{S}$
  $Args_{\mathcal{S}} \leftarrow Arg$
  $\mathbb{U} \leftarrow \{\}$
  $\mathbb{S} \leftarrow \{\}$
  $Pairs_{\mathbb{S}} \leftarrow \{\}$
  $\mathcal{S} \leftarrow 2^{Args_{\mathcal{S}}}$
  for all $s \in \mathcal{S}$ do
    if $\Delta_{pr}(|Args_{\mathcal{S}}|) = |\mathbb{S}|$ then                          ▷ Step I
      break
    if $s \in \mathbb{U}$ then                                                          ▷ Step II
      continue                                        ▷ No superset of found extensions
    if $\exists s' \in \mathbb{S}$ s.t. $s \supset s' \vee s \subset s'$ then           ▷ Step III
      continue                                             ▷ Incomparability is required
    if $\exists S' \in \{S \cup s | S \in \mathbb{S}\} \forall a, b \in S' : (a, b) \in Pairs_{\mathbb{S}}$ then   ▷ Step IV
      continue                                        ▷ Conflict-sensitivity is required
    $r \leftarrow$ Ask agent $Ver_{\sigma}(s)$                                          ▷ Step V
    if $r = \top$ then
      $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$                  ▷ Add $s$ to the extension set $\mathbb{S}$
      for all $p \in s \times s$ do
        $Pairs_{\mathbb{S}} \leftarrow Pairs_{\mathbb{S}} \cup \{p\}$   ▷ Add pairs of arguments appearing together
      if $|\mathbb{S} \geq 2|$ then                                                     ▷ Step VI
        for all $S, S' \in \mathbb{S}$ do
          if $S \neq S'$ then
            $U \leftarrow S \cup S'$
            $\mathbb{U} \leftarrow \mathbb{U} \cup \{U\}$     ▷ Not allowed in an incomparable extension-set
```

Description The algorithm starts by creating all possible sets of arguments as a power set. Then each one is examined. Next to this, $\mathbb{U}$ represents the set of all supersets to found extensions in $\mathbb{S}$. As before, $\mathbb{S}$ represents the collection of all found extensions, also called extension set. In addition, the argument pairs to $\mathbb{S}$ are collected in $Pairs_{\mathbb{S}}$.

In its basic idea, the algorithm is based on the elaborations of the Theorem 2 on the signature Σ_{pr}. First, however, it is known that $s \in \mathbb{S}$ can only be valid if s is maximal with respect to set inclusion, since Definition 10 describes this. Thus, if $\mathbb{U}$ contains the set s, then s need not be considered further. How $\mathbb{U}$ comes about will be shown later.

The next criterion to be applied is incomparability. From Definition 18 it can be seen that for this purpose sets are to be compared pairwise. I.e. s must be compared with all $S \in \mathbb{S}$. It must not be the case that s is a subset of S and vice versa. Otherwise s need not be considered further.

Another criterion to be checked is that the conflict-sensitivity is given. This is described in Definition 20. Applied to the case of elicitation, this means that if for a set $S' \in \{S \cup s | S \in \mathbb{S}\}$ there exists no pair $a, b \in S'$ which is missing in $Pairs_{\mathbb{S}}$, then s need not be considered further.

If these three criteria are met, the agent can be asked whether s actually occurs as an extension, i.e. Ver_{pr} is applied. If this is the case, then s is added to the extension set $\mathbb{S}$ and all pairs are formed from the arguments in s. Furthermore, if there is more than one extension in $\mathbb{S}$, then for all $S, S' \in \mathbb{S}$ with $S \neq S'$ the union can be formed. This represents a superset which cannot occur, since otherwise for S and S' the maximality with respect to the set inclusion according to Definition 10 is not given.

Finally, the stop condition should be named. This is again the diversity function, this time with respect to the preferred semantics Δ_{pr}. If $|\mathbb{S}| = \Delta_{pr}$, the search can be stopped because the computational upper limit of extensions has been reached.

Example 21. *For this example, let us again take the AF F_2 from Figure 5. For $\sigma = pr$ and $F_2 = F_A$ we get here $\sigma(F_A) = \{\{A_1, A_2, A_5, A_6\}, \{A_1, A_2, A_4, A_7\}\}$ with $Arg = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$. We show two positive examples and two negative examples.*

We anticipate that the stop condition in step I is not satisfied in this example.

Let us start with a positive example for $s = \{A_1, A_2, A_5, A_6\}$. Since $\mathbb{U}$ does not yet contain any supersets, we skip this step II. Since $\mathbb{S}$ is still empty as an extension set, we also skip step III. The check of step IV is unnecessary for the same reason. We therefore let the agent s verify. Since it confirms s, we add this to $\mathbb{S}$. Since $|\mathbb{S}| = 1$, step VI need not be performed. We obtain the following set for $Pairs_{\mathbb{S}}$ (every x stands for a pair in $Pairs_{\mathbb{S}}$):

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
A_1	x	x	-	-	x	x	-
A_2	x	x	-	-	x	x	-
A_3	-	-	-	-	-	-	-
A_4	-	-	-	-	-	-	-
A_5	x	x	-	-	x	x	-
A_6	x	x	-	-	x	x	-
A_7	-	-	-	-	-	-	-

We continue with the second positive example of $s = \{A_1, A_2, A_4, A_7\}$. Since $\mathbb{S}$ is still empty, we skip this step. We now check whether s is incomparabile to $\mathbb{S}$. Since s is nei-

ther a subset nor a superset to $\{A_1, A_2, A_5, A_6\}$, this is given. Also, conflict for conflict-sensitivity can be established between s and $S' \in \mathbb{S}$, so in both the pair (A_3, A_3) is not present. The agent can now positively verify s. We add this extension back to the extension set $\mathbb{S}$. Since this satisfies $|\mathbb{S}| \geq 2$, we also add to $\mathbb{U}$ the union set of $\{A_1, A_2, A_5, A_6\}$ and $\{A_1, A_2, A_4, A_7\}$, which is $\{A_1, A_2, A_4, A_5, A_6, A_7\}$. For $Pairs_{\mathbb{S}}$ we get:

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
A_1	x	x	-	x	x	x	x
A_2	x	x	-	x	x	x	x
A_3	-	-	-	-	-	x	-
A_4	x	x	-	x	-	-	x
A_5	x	x	-	-	x	x	-
A_6	x	x	-	-	x	x	-
A_7	x	x	-	x	-	-	x

The first negative example is supposed to be the formed superset $\{A_1, A_2, A_4, A_5, A_6, A_7\}$. It is easy to see that in step II s is rejected and we can continue with another presumed extension.

Let us conclude this example with another negative example. For this, $s = \{A_1, A_2, A_4\}$. This set is not present in $\mathbb{U}$, which is why step II does not reject it. However, the incomparability check takes effect here, since s is a subset of $\{A_1, A_2, A_4, A_7\}$ and is therefore not valid.

If we continue in this way, $\mathbb{S} = pr(F_A)$ is given at the end.

Proof of Correctness of Algorithm 8 In order to prove the correctness, the individual steps are to be proven. The steps can be taken from the comments in Algorithm 8.

Proof of the main part. As the main part is iterative, we will give a proof of correctness by a loop invariant and induction. All steps I to step VI within the loop maintenance and the termination are considered.

Invariant: At the start of each iteration of $\mathcal{S}$, $\mathbb{S}$ should contain only those extensions from $\mathcal{S}$ such that $\mathbb{S}$ is a subset of the extension-set of the agent's AF.

Initialisation: At the start of the first iteration the invariant states that $\mathbb{S}$ should contain only those extensions from $\mathcal{S}$, where at this point no extension has been chosen from, such that $\mathbb{S}$, which contains no extension at this point, is a subset of the extension-set of the agent's AF. As $\mathbb{S} = \{\}$ and $\{\} \subseteq \sigma(F_A)$ with F_A being the AF of the agent, this holds.

Maintenance: Assume that the invariant holds at choosing $s \in \mathcal{S}$. Then it must be that $\mathbb{S}$ must only contain extensions that the agent would verify to be part of their AF, too.

Let $\mathcal{S}$ be the power set of all arguments $Args_{\mathcal{S}}$ from the agent's AF (also given as Arg). $\mathbb{U}$ is a collection of all supersets from the elements of $\mathbb{S}$ and $\mathbb{S}$ is the extension set, i.e. the collection of all verified extensions. In addition, $Pairs_{\mathbb{S}}$ contains the argument pairs, i.e. arguments that occur together in at least one extension that are already present in $\mathbb{S}$ at the time of holding.

Step I in the loop is dealt with in the proof part for the termination and is therefore skipped for the moment.

Step II: This step checks whether s is a superset to the extensions already found and continues the loop if it evaluates to true, due to the criterion of maximality with respect to set inclusion (cf. Definition 10).

That this holds is easy to see. Given two extensions $S, S' \in \mathbb{S}$ and the set of arguments s to be checked. Since it is well known that the union of two sets is a superset of both, i.e. $S \subseteq S \cup S'$ and $S' \subseteq S \cup S'$ respectively, and S as well as S' must be admissible and maximal with respect to set inclusion, $S \cup S' \nsubseteq s$ must hold, otherwise S and S' are not maximal. Since all $S \cup S' \in \mathbb{U}$ (see also step VI), s cannot be an extension of the agent's AF if $s \in \mathbb{U}$ [13].

Step III: Since step II only sorts out exact $S \cup S' = s$ for $S, S' \in \mathbb{S}$, we still have to check incomparability in general, i.e. that there is no set in $\mathbb{S}$ for which s is a superset. We check this in step III with two conditions. The first is that s is a subset of an element in $\mathbb{S}$. If this is the case, we can proceed. This holds by Definition 18.

Conversely, it must also not be the case that an $S \in \mathbb{S}$ is a superset to s, for in this case s would not be maximal with respect to set inclusion. This is checked in the second condition. For this converse, assume that $s \in \mathcal{S}$ and $\exists S \in \mathbb{S} : s \subseteq S \wedge s \neq S$. Now if s were in $\mathbb{S}$, then $\{s, S\} \subseteq \mathbb{S}$ and so there would be an element in $\mathbb{S}$ which is a subset of another element, i.e. $\exists S, S' \in \{s, S\} \subseteq \mathbb{S} : S \subseteq S' \wedge S \neq S'$. This contradicts Definition 18.

Step IV: Next, check for conflict-sensitivity in step IV. Assume that $S \in \mathbb{S}$ and that $s \cup S \notin \mathbb{S}$ (see steps before). So there must be a conflict between s and S, otherwise the condition of conflict-sensitivity would not be met. This can only be the case if there is $(a, b) \notin Pairs_{\mathbb{S}}$. If we assume the opposite, so that $(a, b) \in Pairs_{\mathbb{S}}$ as well as for all other $x, y \in s \cup S$ there is $(x, y) \in Pairs_{\mathbb{S}}$, then there would be no conflict between any two arguments and $\{a, b\} \subseteq s \cup S$. However, then again $s \cup S$ would have to be in $\mathbb{S}$, which is a contradiction to the Definition 20[14].

[13]This step is especially reasonable from a runtime point of view, since the check for an element present in a set varies between best $O(1)$ (e.g. hash sets) and worst $O(n)$ (sets implemented as arrays and having to search for the element iteratively), depending on the implementation.

[14]See also the proof of proposition 2 in [23].

Therefore, step IV checks all $S' \in \{S \cup s | S \in \mathbb{S}\}$, so that for S' there exists an argument pair (a, b) which is not present in $Pairs_{\mathbb{S}}$ so far.

If all these criteria are met, the agent is asked to verify s in step V via Ver_{pr}. In case the agent does not verify them, the iteration is finished. However, if the agent answers in the positive, in this case s is added to $\mathbb{S}$ as an element. In addition, all pairs are formed.
Also, step VI is subsequently performed in the branch of step V.

Step VI: If there are two or more extensions in $\mathbb{S}$, the superset can be formed from all $S, S' \in \mathbb{S}$. This superset can then be included in the collection $\mathbb{U}$ (cf. step II, where it is also shown that this procedure is correct).

The invariant is finished.

Termination: When the loop terminates, $\mathbb{S} = \sigma(F_A)$ with F_A being the AF of the agent. The loop, therefore, terminates in two cases:

1. if there is no set of arguments in $\mathcal{S}$ is left to be checked or
2. if the theoretical maximal number of extensions has been found.

The first case is trivial. The second case holds by the Definition 27 as well as Theorem 5. □

4.3.7. Stable Semantics

As before, we first review and select the possible questions for asking the agent. Therefore, it is useful to look at the questions and their complexity as shown in Table 3. This is followed by an intuitive description of the algorithm with reference to the formal description in Algorithm 9. This is followed by an example and finally, we prove that it works.

Questions In contrast to the preferred semantics, the question Ver_{st} is in complexity class L, i.e. it is to be expected that a guessed extension can be verified by the agent in reasonable time. The situation is different for the remaining questions, which are in the complexity classes NP, coNP and DP. Especially for the stable semantics, the question $Exists_{st}$ would be helpful because, as mentioned in the background, there is not a stable semantics for every AF. Since the typical non-deterministic way for the question is to use an algorithm based on guessing an extension and checking if it exists, this question is not tractable considering an agent that did not enumerate the extensions beforehand, and the expected advantages to the interviewing algorithm without the $Exists_{st}$ resp. $Exists_{st}^{\neg\emptyset}$ matching in this case are not large. The question $Cred_{st}$ hides a similar algorithm. Here, an extension is first guessed and then the specified argument is searched for. As it is also questionable here whether there is an improvement in the actual running time, this question is excluded for the time being[15]. The sceptical acceptance $Scept_{st}$ can be formulated as a complementary problem, i.e. it is shown that an argument is not accepted sceptically. We restrict the algorithm to Ver_{st} for the time being in order not to have both an NP-c and a coNP-c question to the agent in the algorithm. $Unique_{st}$ is also omitted due to its complexity.

Algorithm 9 Stable Semantics Algorithm for Interviewing Part

```
Require: $Arg$ as a set of arguments, $\sigma = st$ as the semantics
Ensure: An extension set $\mathbb{S}$
  $Args_{\mathcal{S}} \leftarrow Arg$
  $Pairs_{\mathbb{S}} \leftarrow \{\}$
  $\mathcal{S} \leftarrow 2^{Args_{\mathcal{S}}}$
  for all $s \in \mathcal{S}$ do
      if $\Delta_{st}(|Args_{\mathcal{S}}|) = |\mathbb{S}|$ then                ▷ Step I
          break
      if $\exists s' \in \mathbb{S} : s \supset s' \lor s \subset s'$ then          ▷ Step II
          continue                                  ▷ Incomparability is required
      $\mathcal{D} \leftarrow Args_{\mathcal{S}} \setminus s$
      $t \leftarrow \forall d \in \mathcal{D}\ \exists a \in s : (d, a) \notin Pairs_{\mathbb{S}}$       ▷ Step III
      if $t = \bot$ then
          continue                                  ▷ $\mathbb{S} \cup \{s\}$ is not tight
      $r \leftarrow$ Ask agent $Ver_{\sigma}(s)$
      if $r = \top$ then                                         ▷ Step IV
          for all $p \in s \times s$ do
              Add $p$ to $Pairs_{\mathbb{S}}$
          $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$                  ▷ Add $s$ to the extension set $\mathbb{S}$
```

Description We will describe the algorithm next. It consists essentially of four steps. The main part is the loop for iterating over the search space in $\mathcal{S}$. In step I, the stop condition is applied. When the maximum number of extensions is reached, then the loop can be terminated early. Otherwise, this loop is terminated thus the algorithm is finished, when all sets in the search space have been checked.

As long as this is not the case, it is checked in step II whether the set is incomparable to all extensions found so far. This is one of two criteria according to Theorem 1 for stable semantics. The extensions found so far are in the extension set $\mathbb{S}$. If this is not the case, it need not be considered further and the next one can be checked.

If it is incomparable, the next step III is to find out whether the set of arguments together with the extension set still fulfil the criterion of tightness in order to satisfy Theorem 1. For this purpose, for each argument that does not occur in the set to

[15]In the experiments, the algorithm is then tested with and without $Cred_{st}$

be checked, there must be a pair of this argument and one argument in the set to be checked, that is not yet in $Pairs_{\mathbb{S}}$. The idea behind this is that only those are suitable to serve as a pair for the tightness test.

If it is also ensured that the tightness criterion is fulfilled in principle, the agent can be asked to verify the extension. This is performed in step IV. If the agent answers positively, all pairs of arguments in the extension are added to $Pairs_{\mathbb{S}}$, since they occur.

Finally, the set is attached to the extension set as an extension.

Example 22. *Let the agent have an hidden AF* $F_A = (\{A_1, A_2, A_3, A_4, A_5\}, \{(A_1, A_4), (A_1, A_5), (A_3, A_4), (A_4, A_3), (A_5, A_1), (A_5, A_2)\})$.

Then $\sigma = st$ *and* $\sigma(F_A) = \{\{A_1, A_2, A_3\}, \{A_4, A_5\}, \{A_3, A_5\}\}$. *The interviewer only knows* Arg *and* σ.

First, the interviewer sets $Args_{\mathcal{S}} = \{A_1, A_2, A_3, A_4, A_5\}$. *In its initial state,* $Pairs_{\mathbb{S}}$ *has no pair of arguments that are accepted, as illustrated in the following table. As notation, an existing pair is marked with x and a non-existing pair with a dash -. The columns represent the first argument of the pair, the rows the second argument.*

	A_1	A_2	A_3	A_4	A_5
A_1	-	-	-	-	-
A_2	-	-	-	-	-
A_3	-	-	-	-	-
A_4	-	-	-	-	-
A_5	-	-	-	-	-

In the next step, extensions have to be guessed. For illustration purposes, not all different sets from the power set $\mathcal{S}$ *are gone through, instead the first two extensions are directly guessed correctly. This is followed by the missing third extension and finally an unverifiable set of arguments is to be checked.*

The maximum number of extensions according to the diversity function is 6, since $n = |Args| \wedge n = 3s + 2$ *with* $s = 1$ *and thus* $2 \cdot 3^s = 6$.

Now we start with the set of arguments $s = \{A_1, A_2, A_3\}$. *Since* $|\mathbb{S}| = 0$ *is not terminated. Moreover,* $\mathbb{S} \cup \{s\}$ *incomparable. Since for all arguments* $Args_{\mathcal{S}} \setminus s = \{A_4, A_5\}$ *pairs can be found with the arguments from* $Args_{\int}$ *that are not in* $Pairs_{\mathbb{S}}$*, such as* (A_4, A_1), (A_5, A_1)*, tightness is also given. Since the agent verifies* s, $Pairs_{\mathbb{S}}$ *can be adjusted as follows:*

	A_1	A_2	A_3	A_4	A_5
A_1	x	x	x	-	-
A_2	x	x	x	-	-
A_3	x	x	x	-	-
A_4	-	-	-	-	-
A_5	-	-	-	-	-

We continue with the set $s = \{A_4, A_5\}$. Since $|\mathbb{S}| = 1$ is not terminated here either. Again, $\mathbb{S} \cup \{s\}$ is incomparable. That $\mathbb{S} \cup \{s\}$ could also be tight is made possible by pairs like (A_1, A_4), (A_2, A_4) and (A_3, A_4) being not in $Pairs_{\mathbb{S}}$. Since tightness is given, the agent can be asked for verification. The agent answers positively and $Pairs_{\mathbb{S}}$ now looks like this:

	A_1	A_2	A_3	A_4	A_5
A_1	x	x	x	-	-
A_2	x	x	x	-	-
A_3	x	x	x	-	-
A_4	-	-	-	x	x
A_5	-	-	-	x	x

Now the third extension $s = \{A_3, A_5\}$ is to be checked. Again, the algorithm does not terminate because $|\mathbb{S}| = 2$. Also for this invariant, $\mathbb{S} \cup \{s\}$ incomparable and tightness can be checked with the pairs (A_1, A_5), (A_2, A_5) and (A_4, A_3), which all are not contained in $Pairs_{\mathbb{S}}$. Since the agent also verifies s, we get the following $Pairs_{\mathbb{S}}$:

	A_1	A_2	A_3	A_4	A_5
A_1	x	x	x	-	-
A_2	x	x	x	-	-
A_3	x	x	x	-	x
A_4	-	-	-	x	x
A_5	-	-	x	x	x

Finally, $s = \{A_1, A_5\}$ is to be checked, which is not in $\sigma(F_A)$. First, again the algorithm is not terminated because only three extensions have been found and $\Delta_{st}(|Args_{\mathcal{S}}|) = 6$. Next, the incomparability is checked again, which is also the case as $\mathbb{S} \cup \{s\}$ is given. However, it is easy to check that under the circumstances of the extensions found earlier, $\mathbb{S} \cup \{s\}$ cannot be tight. It is $Args_{\mathcal{S}} \setminus s = \{A_2, A_3, A_4\}$ to be checked. For the first argument A_2, the pair (A_2, A_5) can not be found in $Pairs_{\mathbb{S}}$. For the second argument A_3 pairs can be found, since both (A_3, A_1) and (A_3, A_5) are in $Pairs_{\mathbb{S}}$. Therefore, we can proceed with another guessed set of arguments.

In the end, after all possible sets have been tried, the extension set is equal to the result of the semantic function using st for the AF F_A, i.e. $\mathbb{S} = \sigma(F_A)$.

Proof of Correctness of Algorithm 9 In order to prove the correctness, the individual steps are to be proven. The steps can be taken from the comments in Algorithm 9.

Proof. Let $Args_{\mathcal{S}}$ be a set of all arguments, $\mathbb{S}$ the extension set, $Pairs_{\mathbb{S}}$ be the set of argument pairs that occur in at least one extension, S the power set of $Args_{\mathcal{S}}$ and F_A be the hidden AF of the agent.

As the main part of the algorithm is iterative, we will give a proof of correctness by a loop invariant and induction.

Invariant: At the start of each iteration of $\mathcal{S}$, $\mathbb{S}$ should contain only those extensions from $\mathcal{S}$ such that $\mathbb{S}$ is a subset of the extension-set of the agent's AF.

Initialisation: At the start of the first iteration the invariant states that $\mathbb{S}$ should contain only those extensions from $\mathcal{S}$, where at this point no extension has been chosen from, such that $\mathbb{S}$, which contains no extension at this point, is a subset of the extension-set of the agent's AF. As $\mathbb{S} = \{\}$ and $\{\} \subseteq \sigma(F_A)$ with F_A being the AF of the agent, this holds.

Maintenance: Assume that the invariant holds at choosing $s \in \mathcal{S}$. Then it must be that $\mathbb{S}$ must only contain extensions that the agent would verify to be part of their AF, too.

Step II: First we check if the incomparability requirement is satisfied in step II. Remember from Definition 18 that all elements of $\mathbb{S}$ have to pairwise incomparable, i.e. for each $S, S' \in \mathbb{S}$, $S \subseteq S'$ implies $S = S'$. If the chosen s is a subset of any found extension in the extension-set $\mathbb{S}$, then it must not be considered further. To prove this, assume that s is a subset of any $s' \in \mathbb{S}$ and $s \neq s'$, then due to incomparability the agent will not verify s for their AF. Therefore s must not be considered.

Step III: Next, we check if s is tight with respect to current state $\mathbb{S}$ in step III. Different to Definition 19 it is not clear if $\mathbb{S}$ is complete or not. However, due to incomparability we know that if the agent verifies s, then there can be no subset nor superset. Therefore we can assume for now that $\mathbb{S} \cup \{s\}$ must be tight. To be tight under this circumstances means that for $a \in Args_{\mathcal{S}}$ there is $s \cup \{a\} \notin \mathbb{S}$ and therefore there exists an $a_s \in s$ such that $(a, a_s) \notin Pairs_{\mathbb{S}}$. As we do not need to consider arguments in s itself, we only need to check whether pairs of $Args_{\mathcal{S}} \setminus s$ and arguments in s are not in $Pairs_{\mathbb{S}}$ correspondingly, because for all these arguments a pair is needed, which is not contained in $Pairs_{\mathbb{S}}$. Assume the contrary that there is a $s \in \mathbb{S} \wedge s \in \mathcal{S}$ such that there is an argument $a_s \in s$ so there is $(a, a_s) \in Pairs_{\mathbb{S}}$ for all $a \in Args_{\mathcal{S}}$. Then $s \subseteq \mathbb{S}$ is not tight, as by the definition of tightness there must be a pair that is not in $Pairs_{\mathbb{S}}$. Contradiction!

Step IV: Following these checks, in step IV, the agent is asked to verify s, there are two cases: i) The agent verifies s to be an extension, then first all pairs of arguments are added to $Pairs_{\mathbb{S}}$, because they occur together.

Second, the set s is added to the extension-set $\mathbb{S}$.

In case ii) the agent does not verify the set of arguments as an extension of their AF and therefore s is not considered further.

Termination: When the loop terminates, $\mathbb{S} = \sigma(F_A)$ with F_A being the AF of the agent. The loop, therefore, terminates in two cases:

1. if there is no set of arguments in S is left to be checked or
2. if the theoretical maximal number of extensions has been found.

The first case is trivial. The second case holds by the Definition 27 and the Theorem 4 as well as Theorem 5.

□

4.3.8. Discussion

Finally, the algorithms developed for the interview part will be discussed. For this purpose, the available information, the questions and the running time will be covered below.

First, about the available information. Given in the algorithms are both the semantics and the arguments of the agent's hidden argumentation framework. With the restriction that the agent can only answer semantic decision questions with an extension relation, a narrow boundary has been set. The computational problems considered ($Cred_\sigma$, $Scep_\sigma$, Ver_σ, $Exists_\sigma$, $Exists_\sigma^{\neg\emptyset}$ and $Unique_\sigma$; see also Table 4) provide little insight into the agent's attack relations or argumentation framework model.

σ	$Cred_\sigma$	$Scept_\sigma$	Ver_σ	$Exists_\sigma$	$Exists_\sigma^{\neg\emptyset}$	$Unique_\sigma$	$ActualCount_\sigma$
cf	✓		✓				
ad			✓				
gr	✓				✓		
co			✓				
st			✓				
pr			✓				

Table 4: Questions used in the interviewing part according to the algorithm for σ. A ✓stands for used.

Due to the fact that only information on the extensions could be requested, some of the findings of Dunne et. al [23] on the signatures were suitable for the development of the algorithms. Through the iterative procedure or by guessing possibly existing extensions, properties such as incomparability, tightness and conflict-sensitivity can be used for the most part to check whether the extension to be checked can be contained at all. It should be noted, however, that this cannot be used to catch all non-existent extensions before they are verified by the agent, nor is it possible to check for all properties. For example, the com-closed property is not sufficiently suitable, so that it was not used in the algorithm for complete semantics.

Furthermore, Baroni and Giacomin [4] gave a classification of semantics based on so-called principles. However, these principles were not suitable for the applications in the algorithms, as they often needed a deeper insight into the AF of the agent (e.g. for reinstatement or directionality). Other principles from [2] such as allowing abstention could not be applied either.

In addition, the previously mentioned questions and computational problems are of interest. Only a few of them are in a complexity class considered tractable for an agent that has not enumerated the extensions in advance. This is particularly striking since only $\{cf, gr\} \subseteq \sigma$ use $Cred_\sigma$ as a question in these versions of the algorithms. The absence of this question in the other semantics means that the search space (the power set of given arguments) cannot be reduced. Since the question $ActualCount_\sigma$ could also not be meaningfully included (due to its complexity in #P), no further restrictions of the search space are possible. However, we suspect that the use of these questions, which have not been utilised so far, may well give a performance advantage if the solver has good strategies for answering those questions. Therefore, in Appendix B we also use another solver in addition to the one from the experiments in Section 5 and briefly examine whether these modified algorithms that also use questions from NP, coNP and other classes perform better with this other solver that takes a different implementation strategy.

As for ending the search, without $ActualCount_\sigma$ only the fact that a theoretical upper limit (the diversity function) exists for many semantics makes it possible to end the search early.

This also points to the runtime complexity. In the absence of stop conditions apart from the diversity function, the runtime is strongly dependent on the number of arguments the agent's AF possesses. The iteration over the search space (the power set) is thus in principle for all semantics at $O(n^{|Arg|})$ with $|Arg|$ as the number of arguments. This already suggests long runtimes, as will be verified in the experiments in Section 5 on evaluation.

The only exception is the algorithm for grounded semantics. It only grows with the size of the set of arguments and is therefore in $O(|Arg|)$.

All in all, the restriction to ask only semantic decision questions and not allowing an agent to enumerate in advance shows that, on the one hand, a lot of information is not available and, on the other hand, that the computation of the questions by the agent is often not tractable. As a result, only a few questions can be used meaningfully under these restrictions.

4.4. Discussion on the Reconstruction

In this subsection, we discuss the reconstruction. Above all, it will be shown what limits there are for a tractable construction of an AF from the information collected from the interview part. This provides a rationale why more computational complex or statistics-based approaches to generating frameworks have their raison d'être.

First, we clarify under which circumstances an extension-set is realisable under a given classical semantics at all.

On the one hand, a naïve approach can be taken, in which all attack relationships are simply tried until a suitable AF is found. This algorithm is depicted in Algorithm 10 but is obviously not very efficient, since in the worst case all combinations have to be tried.

Algorithm 10 Naïve Approach for Generating Part

```
Require: $Arg$ as a set of arguments, $\sigma$ as the semantics, an extension set $\mathbb{S}$
Ensure: An argumentation framework $F_E$
  $R \leftarrow (Arg \times Arg)$
  for all $r \in 2^R$ do
      $G \leftarrow (Arg, r)$
      if $\sigma(G) = \mathbb{S}$ then
          break
  $F_E \leftarrow G$
```

Dunne et al. [23] demonstrated that for σ, any extension-set $\mathbb{S}$ that is contained in the signature of σ, i.e. $\mathbb{S} \in \Sigma_\sigma$, can be realised with a canonical AF or an adapted version of it (see Definition 23 for canonical AFs; Definition 24 and Definition 26 for adapted canonical AFs). However, in the case of the adapted canonical AFs, a large number of artificial arguments are added that are needed for the construction of the canonical AF, but which do not occur in the agent's AF.

However, in our case, the arguments of the agent's AF are known and it is also known that there is at least one AF realising $\mathbb{S}$ without those artificial arguments. Therefore, in the following, we want to examine the possibilities of reconstructing an argumentation framework from the given information and discuss the limitations given the condition that there are no artificial arguments and all arguments of F_A are also in the constructed or generated AF F_E. We call such a reconstruction argument-congruent.

Definition 47. *Let Arg be a set of arguments of an abstract argumentation framework $F = (Arg, AR)$ and σ a semantics. An argument-congruent reconstruction with respect to σ is one in which a $F' = (Arg', AR')$ is generated such that $Arg = Arg'$ and $\sigma(F) = \sigma(F')$.*

We also say a reconstruction is immediate if the (re-)construction from the interviewing part is tractable, this means doable in polynomial time with regard to time complexity.

For exploring under which semantics immediate argument-congruent reconstructions are possible, we first show cases in which an at least σ-equivalent AF can be generated argument-congruently from the elicited extensions. We then examine cases in which this is not or not always possible, using among others the example of the preferred semantics and the running example from Section 2.2.

4.4.1. Immediate Argument-Congruent Reconstruction

Let us begin with the grounded semantics which produces a unique extension. As discussed earlier, a canonical AF is very suitable for generating a σ-equivalent AF under grounded semantics. Such an AF can be described by $F_E = (\{E\}, \emptyset)$ such that given the agent's AF F_A it holds that $gr(F_A) = \mathbb{S} = \{E\}$ because $E \subseteq Arg$.

However, this also means that not all the original arguments in Arg are necessarily found in this canonical AF. To get an argument-congruent AF and therefore syntactically closer such that all arguments of Arg are in the AF, the missing arguments $Arg \setminus E$ can be added to the AF if they attack themselves such that $AR = \{(a, a) \mid a \in (Arg \setminus E)\}$. The associated AF will look like $F_E = (Arg', AR')$ has $Arg' = Arg$ then.

Algorithm 11 Algorithm for an Argument-Congruent Reconstruction of the Grounded Semantics

```
Require: $Arg$ as a set of arguments, an extension set $\mathbb{S}$
Ensure: An argumentation framework $F_E$
  $E \leftarrow \mathbb{S}_1$                ▷ Get the first element via an implicit index set
  $N \leftarrow Arg \setminus E$
  $AR \leftarrow \{(a, a) \mid$ for each $a \in N\}$
  $F_E \leftarrow (Arg, AR)$
```

Algorithm 11 depicts how to generate such an AF. However, Example 23 shows why such a σ-equivalent AF is not guaranteed to be syntactically identical to the agent's hidden AF.

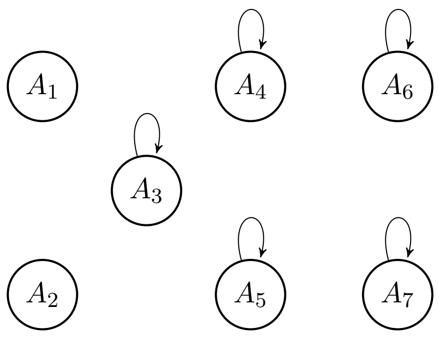

Figure 17: An example for a σ-equivalent reconstruction of framework F_2 under the grounded semantics.

Example 23. *For a reconstruction example, let us take again the already known AF F_2 from Figure 5, which already served as a running example before. For this AF, $Arg = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$ and the extension set from the first interview step is $\mathbb{S} = \{\{A_1, A_2\}\}$. We extract the extension from $\mathbb{S}$ such that $E = \{A_1, A_2\}$ and form the set difference $Arg \setminus E = \{A_3, A_4, A_5, A_6, A_7\}$. For each of these arguments in the set difference we* *generate a self-attack so that we get the AF F_E with $F_E = (\{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}, \{(A_3, A_3), (A_4, A_4), (A_5, A_5), (A_6, A_6), (A_7, A_7)\})$. The σ-equivalence is satisfied with $\sigma(F_2) = \mathbb{S}$ for $\sigma = gr$.*

Figure 17 visualises this AF. It is obvious that although the same arguments were used, the attack relations between AF F_2 and the one reconstructed here differ. Thus, only self-attacks are present in the reconstructed AF while F_2 does not have a single self-attack. In exchange, all other attack relations in F_2 are missing in the reconstructed AF.

Let us continue with another semantics for which such an approach is possible. That are conflict-free sets. This follows directly from Proposition 2, according to which a canonical AF $F^{cf}_{\mathbb{S}}$ can also be generated, which satisfies $cf(F^{cf}_{\mathbb{S}}) = \mathbb{S}$. Any arguments not in $Args_{\mathbb{S}}$ can again be added to an AF by self-attacks, so that AR is generated from the union of $(Args_{\mathbb{S}} \times Args_{\mathbb{S}}) \setminus Pairs_{\mathbb{S}}$ and $\{(a, a) \mid a \in Arg \setminus Args_{\mathbb{S}}\}$. Algorithm 12 formally describes how such an argument-congruent reconstruction is possible.

Algorithm 12 Algorithm for an Argument-Congruent Reconstruction of Conflict-Free Sets

```
Require: $Arg$ as a set of arguments, an extension set $\mathbb{S}$
Ensure: An argumentation framework $F_E$
  Generate $Pairs_{\mathbb{S}}$
  Generate $Args_{\mathbb{S}}$
  $AR_1 \leftarrow (Args_{\mathbb{S}} \times Args_{\mathbb{S}}) \setminus Pairs_{\mathbb{S}}$
  $AR_2 \leftarrow \{(a, a) \mid a \in Arg \setminus Args_{\mathbb{S}}\}$
  $F_E \leftarrow (Arg, AR_1 \cup AR_2)$
```

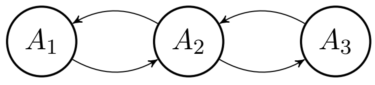

Figure 18: The reconstructed argumentation framework F_1 from Example 1.

Example 24. *In this example, we show the argument-congruent reconstruction of conflict-free sets. Our otherwise used running example F_2 from Figure 5, is not suitable for a short example in this case. We therefore fall back on AF F_1 from Figure 2. Recall $F_1 = (\{A_1, A_2, A_3\}, \{(A_1, A_2), (A_2, A_1), (A_2, A_3)\})$ and for $\sigma = cf$ is $\sigma(F_1) = \{\{\}, \{A_1\}, \{A_2\}, \{A_3\}, \{A_1, A_3\}\}$. We also assume that the extension set $\mathbb{S} = \sigma(F_1)$ for $\sigma = cf$ was found during the interview. Next, we present the procedure of Algorithm 12.*

First, we generate the argument pairs with $Pairs_{\mathbb{S}} = \{(A_1, A_1), (A_2, A_2, (A_3, A_3), (A_1, A_3), (A_3, A_1)\}$. Then we form $Args_{\mathbb{S}}$ and obtain for this $Args_{\mathbb{S}} = \{A_1, A_2, A_3\}$. Next, we construct the attack relation between all the arguments in $Args_{\mathbb{S}}$ without the $Pairs_{\mathbb{S}}$ and get $AR_1 = \{(A_1, A_2), (A_2, A_1), (A_2, A_3), (A_3, A_2)\}$. Moreover, we have to eliminate all arguments that are not in $Args_{\mathbb{S}}$ but are in Arg by self-attacks. In this case, there are no such arguments, so $AR_2 = \{\}$ holds.

We now (re-)construct F_E such that $F_E = (\{A_1, A_2, A_3\}, \{(A_1, A_2), (A_2, A_1), (A_2, A_3), (A_3, A_2)\}$. Figure 18 represents this. It is noticeable that this is very similar to AF F_1 and differs in the attack relation only by a symmetric attack between A_2 and A_3. Again, in the end $\sigma(F_1) = \mathbb{S} = \sigma(F_E)$ with $\sigma = cf$.

These findings and algorithms also correspond to the state-of-the-art research on the subclass of analytical AFs as in [8]. An AF is analytic if it has no implicit conflicts. Unlike attacks, which are a syntactic element, conflicts are with respect to a semantics and are characterised by the fact that two arguments do not occur together. The following Definition 48 and Definition 49 correspond to definitions 3 and 4 in Baumann et al.'s paper [8].

Definition 48. *Let an AF $F = (Arg, AR)$, a semantics σ and two arguments $a, b \in Arg$ be given. If $(a, b) \notin Pairs_{\sigma(F)}$, then a and b are in* conflict *in AF F for semantics σ. If there is an attack relation between the arguments, i.e. $a \hookrightarrow b \vee b \hookrightarrow a$, then the conflict is* explicit *and otherwise* implicit*.*

Definition 49. *Let σ be a semantics, then an AF F is called* analytic *for σ if all conflict in F for σ are explicit.*

For the other semantics, one consideration can be to put the arguments that do not occur together in extensions into an explicit conflict. However, because the Explicit Conflict Conjecture (ECC) was rejected for admissible, complete, preferred and stable semantics, they cannot be always translated into an AF with only explicit conflicts and therefore this consideration cannot be applied. The question underlying the ECC was whether an arbitrary AF can be transformed into an analytic AF without additional arguments under the stable semantics. The conjecture was disproved in [8] and, moreover, it was shown for the other not yet considered semantics that the subclass of finite analytic AFs is a strict subset of finite AFs. From this, it can be concluded that for these semantics the implicit conflict is part of their expressive power.

4.4.2. Negative Results in Immediate Argument-Congruent Reconstruction

From the previous results, we now want to show that immediate construction algorithms, as previously given with Algorithm 11 and Algorithm 12, are not sufficient for (immediate) argument-congruent reconstructions from the given information and therefore more computational complex approaches are needed for semantics other than conflict-free sets and the grounded semantics such as e.g. described in the synthesis of AFs [37] for these semantics.

For this, we show the limits on the basis of the preferred semantics. However, since the Explicit Conflict Conjecture has also been refuted for other semantics, the ideas can be applied to these as well.

Demonstration on Preferred Extensions It is known that the preferred semantics is based on admissibility. Thus, in order to realise an extension E under the given arguments Arg, there must be a conflict between the arguments in E and the AF's arguments that are not in E. This conflict must be designed in such a way that the AF's arguments are not in E do not appear in E while all other extensions can still be successfully produced. An extension under Arg with Arg from an AF F is thus producible if for each argument $a \in Arg \setminus E$ either i) it holds that it does not occur in any other extension, i.e. $\forall E' \in \mathbb{S} : a \notin E'$ or ii) if it occurs in another extension, then in all remaining extensions $E' \in \mathbb{S}$ with $E' \neq E$ there must be at least one argument $b \in E$ for which there is no pair, i.e. $(a, b) \notin Pairs_{\mathbb{S}}$.

A first approach based on the previously described algorithms would be to construct an analytical AF, i.e. one where all those conflicts are made explicit by attacks. Thus, for each $(a, b) \notin Pairs_{\mathbb{S}}$, an attack $a \hookrightarrow b$ is generated for an explicit conflict. This gives us an argumentation framework $F = (Arg, \{(a, b) \mid (a, b) \notin Pairs_{\mathbb{S}} \wedge a, b \in Arg\})$.

Example 25. *Let us take again the AF from Figure 5, i.e.* $F_A = (\{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}, \{(A_1, A_3), (A_2, A_3), (A_3, A_4), (A_3, A_5), (A_4, A_5), (A_5, A_4), (A_4, A_6), (A_5, A_7)\}$, *under the preferred semantics,* $\sigma = pr$. *After the interview has been conducted with the agent, the extension-set* $\mathbb{S} = \{\{A1, A2, A4, A7\}, \{A1, A2, A5, A6\}\}$ *was elicited and thus* $\sigma(F) = \mathbb{S}$ *holds. In order to produce a* σ*-equivalent AF* F_E *from* $\mathbb{S}$, $Pairs_{\mathbb{S}}$ *must first be calculated. The attack relation* AR *can be constructed from all pair of arguments that are not in* $Pairs_{\mathbb{S}}$, *such that* $AR = \{(A_1, A_3), (A_2, A_3), (A_3, A_1), (A_3, A_2), (A_3, A_3), (A_3, A_4), (A_3, A_5), (A_3, A_6), (A_3, A_7), (A_4, A_3), (A_4, A_5), (A_5, A_3), (A_5, A_4), (A_4, A_6), (A_6, A_3), (A_{6,A} 4), (A_4, A_6), (A_5, A_7), (A_7, A_5), (A_6, A_7), (A_7, A_6), (A_7, A_3)\}$.

Thus F_E *is constructed. Obviously, all conflicts are now explicit and symmetric in this case. It is worth noting, that* A_6 *and* A_7 *are in explicit conflict, while in* F_A *there is only an implicit conflict.*

The Example 25 can be used to illustrate such a construction. In this case, by means of the construction of an analytic AF, it could be provided that $\mathbb{S} = pr(F_E) = pr(F_A)$. Noticeably, the AF F_E has symmetric attacks with the exception of the self-attack $A_3 \hookrightarrow A_3$. This self-attack can be understood as a special case when an argument is not found in any extension at all, i.e. $\{a\} \nsubseteq \mathbb{S}$. In this case, that argument is in conflict with all arguments in all extensions. Such an argument can be deleted from all extensions by only attacking itself, i.e. $a \hookrightarrow a$, since a self-attacking argument cannot be conflict-free. All other conflicts with this argument then must not be stated explicitly. This reduces the number of attacks, since according to the algorithm described earlier, a large number of conflicting pairs would be found that are meaning less, as a pair (a, a) would be found and besides all the symmetric attacks, this argument a would be also be self-attacking to make the conflict explicit.

In Example 26, for instance, such an AF F_E is constructed. From this, it can be concluded that not only analytical AF exist for the extension set under the preferred semantics. This circumstance is called quasi-analytic in [8]. See also Definition 50 which is in accordance with [8].

Definition 50. *An AF F is called* quasi-analytic *for σ, if there is an AF F' such that $Arg_F = Arg_{F'}$, $\sigma(F) = \sigma(F')$ and F' is analytic for σ. Otherwise an AF is* non-analytic *for σ if it is not quasi-analytic for σ.*

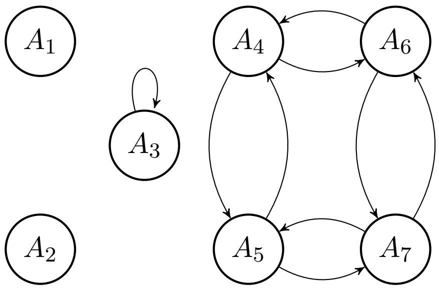

Figure 19: An example for a constructed σ-equivalent AF F_E in Example 26.

Example 26. *Let us take again the AF from Figure 5, i.e. $F_A = (\{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}, \{(A_1, A_3), (A_2, A_3), (A_3, A_4), (A_3, A_5), (A_4, A_5), (A_5, A_4), (A_4, A_6), (A_5, A_7)\}$, under the preferred semantics, $\sigma = pr$. After the interview has been conducted with the agent, the extension-set $\mathbb{S} = \{\{A_1, A_2, A_4, A_7\}, \{A_1, A_2, A_5, A_6\}\}$ was elicited and thus $\sigma(F) = \mathbb{S}$ holds. In order to produce a σ-equivalent AF F_E from $\mathbb{S}$, $Pairs_{\mathbb{S}}$ must first be calculated. Thereupon it can be established that A_3 does not occur in any extension and therefore A_3 cannot be conflict-free. So we add to AR in F_E the relation (A_3, A_3) and in the following A_3 must not be considered further. Now those relations can be calculated for arguments that cannot occur together in an extension. This, together with the self-attack for A_3, gives $AR = \{(A_4, A_5), (A_5, A_4), (A_4, A_6), (A_{6,A}4), (A_4, A_6), (A_5, A_7), (A_7, A_5), (A_6, A_7), (A_7, A_6)\}$. Thus F_E is constructed, see also Figure 19 for an illustration. Obviously, A_3 in F_E attacks itself, while in F_A the argument is attacked by A_1 and A_2. This attack relation in turn no longer exists and A_1 and A_2 are isolated. Between A_4 and A_6 as well as A_5 and A_7 there are now symmetrical attacks instead of unidirectional ones. Moreover, A_6 and A_7 are in explicit conflict, which only occurs implicitly in F_A.*

Next, we show that forming attacks solely from pairs of arguments that do not occur together does not always lead to the desired result. In Example 27, we demonstrate such a case by using a modification of the AF from Example 5.

Example 27. *Let us take the modified AF from Figure 20 such that $F = (\{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}, \{(A_1, A_2), (A_2, A_1), (A_1, A_3), (A_3, A_1), (A_1, A_4), (A_3, A_4), (A_4, A_3), (A_3, A_5), (A_4, A_6)\})$ under the preferred semantics, i.e. $\sigma = pr$. Several modifications are made to the attack relation, such that e.g. A_7 is not attacked any more as well as that there is, for instance, a symmetric attack between A_1 and A_2 amongst further changes. After conducting the interview part, an extension-set $\mathbb{S} = \{\{A_2, A_3, A_6, A_7\}, \{A_2, A_4, A_5, A_7\}, \{A_1, A_5, A_6, A_7\}\}$ is gathered.*

We now show that making all conflicts explicit does not create an AF that realises this extension. Therefore the attack relation AR has to be created from $\{(a, b) \mid (a, b) \notin Pairs_{\mathbb{S}} \wedge$

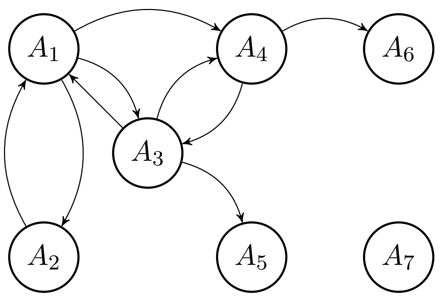

Figure 20: An example for an AF F in Example 27, where a σ-equivalent AF can not be argument-congruently constructed immediately from the given information.

$a, b \in Arg\}$) with Arg being the arguments of F. This results in the AF $F_E = (Arg,$ $\{(A_1, A_2), (A_3, A_4), (A1, A_3), (A_3, A_5), (A_1, A_4), (A_4, A_1), (A_6, A_4), (A_2, A_1), (A_4, A_3),$ $(A_3, A_1), (A_5, A_3)\})$. This AF F_E is depicted in Figure 21 and has as the extension-set $\sigma(F_E) = \{\{A_2, A_4, A_5\}, \{A_2, A_3, A_6, A_7\}, \{A_1, A_5, A_6, A_7\}, \{A_2, A_5, A_6, A_7\}\}$. This violates the postcondition that $\mathbb{S} = \sigma(F) = \sigma(F_E)$, since $pr(F) \neq pr(F_E)$.

One way to repair this AF F_E is to remove the attack $A_5 \hookrightarrow A_3$. However, it is not straightforward to find out which conflicts actually need to become explicitly attack relationships.

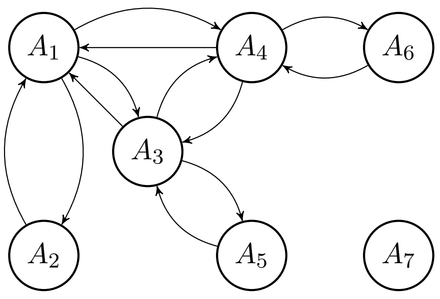

Figure 21: The AF F_E in Example 27.

Finally, in [8], the conclusion is drawn that there are also non-analytic AFs for $\sigma = pr$ (cf. collary 26 in [8]). This is based on the theorem that there exist non-analytic AFs for the stable semantics. The authors prove this in theorem 24 of their paper. In the example used for the proof, the stable extensions coincide with, among others, the preferred extensions. From this, they conclude that there are also non-analytical AFs for the preferred semantics.

This, together with the realisation that attacks cannot be formed directly from the pairs of arguments that do not occur together, since the symmetry in $\{(a, b) \mid (a, b) \notin Pairs_{\mathbb{S}} \wedge a, b \in Arg\})$ when $a \neq b$ does not always give the expected result, leads to the fact that the approach is not suitable for immediate argument-congruent reconstruction.

Overall conclusion In summary, without syntactic questions setting out the attack relations, no immediate argument-congruent reconstruction is possible for all existing finite AFs with respect to the remaining semantics $\sigma = \{ad, co, pr, st\}$. A reconstruction based only on the given information (that is, the semantics, the extension set and the occurring arguments in the AF) only works partially for specific classes of AF. Therefore non-immediate approaches as e.g. an iterative approach like the AF synthesis are of interest if one wants to reconstruct the AF argument-congruent. Baumann et al. [8] also showed that the decision whether an analytic AF for $\sigma \in \{ad, co, pr, st\}$ exists lies in the problem class NP and is complete for NP. Therefore, even the cases where reconstruction would be immediately possible are not of much relevance, since NP-completeness is seen intractable and approaches like AF synthesis are mostly NP-complete[16], too, and in the case of a zero-cost solution always directly lead to an AF which is σ-equivalent. This means that instead of checking whether it is a case for immediate reconstruction, one can also directly perform a reconstruction that is not immediate and obtain a σ-equivalent AF.

5. Evaluation

To assess the practicability of the algorithms introduced in Section 4, some experiments are now following. We have developed three scenarios for this purpose. In the first experiment, we assume that the agent has not already enumerated the extensions and is not able to remember this information in the question-answer exchange. At first, the naïve baseline will be compared with the developed algorithms. Then, we will experimentally test which runtime differences occur when the agent was previously able to enumerate the extensions. Finally, as a third scenario, modifications are made to the algorithms so that for agents who were previously able to enumerate all extensions, the questions that were previously intractable are also applied. The runtimes are also measured in this case.

With regard to the experiments, it should be mentioned that the focus is on the interview part. Therefore as approaches for reconstruction already exist, a comparison between naïve baseline and existing approaches in the reconstruction of a σ-equivalent AF is not considered experimentally. The underlying reason is that complexity analyses and experiments already exist in the literature, e.g. for abstract argumentation framework synthesis in [37].

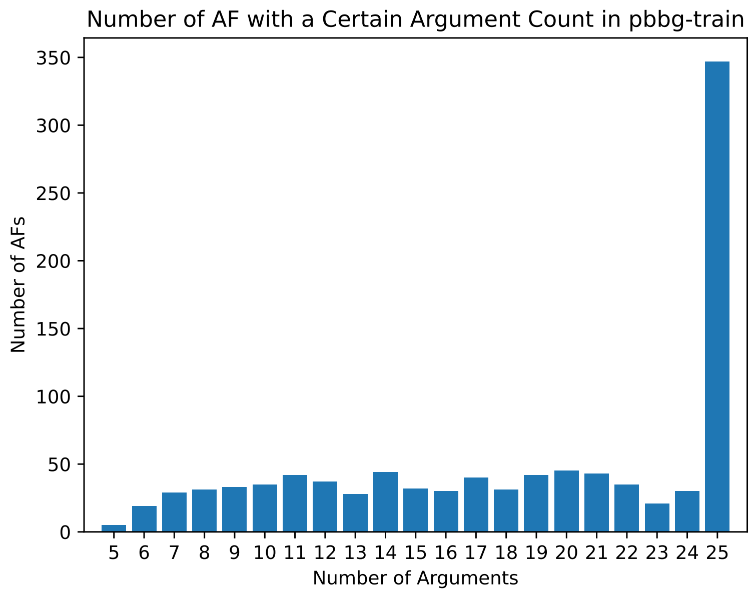

Figure 22: Number of AF in pbbg-train with a certain argument count.

5.1. Experimental Setup

The different algorithms for the interviewing part were implemented using Java and the TweetyProject library, an open source project providing an general interface for, amongst others, computational argumentation [46, 47]. For this purpose, we developed solvers for the individual computational tasks of the questions directly in Java. However, in the Appendix B we briefly present what effect the use of a different solver has on the performance. This first stage of the elicitation process is also the main focus of this Master's thesis.

The frameworks of the third International Competition on Computational Models of Argumentation (ICCMA'19) [11] on the one hand and the test data set (pbbg-test) and training data set (pbbg-train) from the work of Craandijk and Bex [18] on the other hand were used as the data basis. The ICCMA'19 datasets consist of 326 AFs without the new benchmarks. In contrast, the latter datasets from [18] consist of 1000 AFs with $|Arg| = 25$ arguments for pbbg-test and one hundred thousand AFs with $5 \leq |Arg| \leq 25$ for pbbg-train. 1000 datasets were randomly selected from the pbbg-train for the experiments. The distribution of the randomly chosen AFs can be seen in Figure 22. It is noteworthy that about one third of the randomly selected AFs have an argument count of $|Arg| = 25$.

[16]Except for the grounded semantics which is in P and the preferred semantics which is NP-hard and in Σ_2^P in the unrestricted case.

The different datasets allow us to compare AFs of different difficulties. While the pbbg-train and pbbg-test are small AFs, the ICCMA'19 AFs vary in size from small AFs with the size of pbbg-train and pbbg-test to AFs with several hundred arguments. In addition, due to its fixed number of arguments, pbbg-test serves to check to what extent the attack relationships influence the running times of the algorithms.

The following experiments were carried out with the implementations mentioned on a machine with an Intel Core i5-8350U with 1.70 GHz and 4 cores as well as 8 GB RAM. Debian 11 bullseye was used as the operating system and Eclipse Temurin for JDK 17 was used as the Java platform.

5.2. Evaluation of the Developed Algorithms

The first experiment compares the naïve baseline with the developed algorithms for an agent of the type $Agent_{\neg\mathbb{E}}^{dec,sem}$. For this purpose, the runtime and the number of arguments in each instance are the metrics to be measured. The aim is to answer the following questions:

1. *How well do the developed algorithms perform in terms of runtime compared to the naïve algorithm?*
   To benchmark the performance, the developed algorithms will be compared with the naïve baseline in terms of CPU runtime. The aim is to examine whether the developed algorithms bring about an improvement.

   As part of the analysis, a hypothesis is formulated and then tested using the Wilcoxon signed-rank test. This statistical hypothesis test is a non-parametric test for matched-pair data. This test assigns a sign to each observation depending on whether the observation is below or above a value. The paired difference test is suitable for two matched samples and the one-sample variant for testing the location of a population. A comprehensive explanation can be found in the article by Whitley and Ball [51].

2. *Do the number of arguments and the running time correlate?*
   The aim of this question is not only to check whether there is a correlation but also whether they are linearly or non-linearly related.

   For this purpose, a regression analysis is carried out as a standard statistics instrument. As a measure besides the coefficient of determination, the mean absolute percentage error (MAPE) is used as a measure of accuracy and is defined as $MAPE = \frac{100\%}{n} \sum_{t=1}^{n} |\frac{A_t - F_t}{A_t}|$ with A_t the true value and F_t the forecast value. It is a popular measure of prediction accuracy and is therefore suitable for checking how far a prediction for a function deviates from the actual runtime data. This measure is explained clearly for the interested reader in the paper [32], among others.

To investigate these questions, the data sets pbbg-train and pbbg-test are used. An upper time limit (a timeout) was set at five minutes. All semantics $\sigma = \{cf, ad, co, gr, pr, st\}$ were used for the comparison of the algorithms.

First, we describe the results in the following subsections per semantics. Therefore a scatter plot illustrates the comparison of the running times between the algorithms. Furthermore, a box plot for pbbg-train per semantics is given to investigate the influence of the set of arguments on the execution time.

Subsequently, the questions posed earlier are answered collectively for the all of the semantics from the data obtained.

5.2.1. Conflict-Free Sets

Let us start with the evaluation of the algorithms for eliciting conflict-free sets. Figure 23 is a scatter plot with the X-axis for the runtime of Algorithm 4 in milliseconds and the Y-axis for the naïve algorithm. Both datasets pbbg-test (with red triangles) and pbbg-train (with cyan X-markers) are shown. It is noticeable in this graph that there is a high number of timeouts for Algorithm 4. Apart from a few outliers, it can be seen that the naïve algorithm was faster.

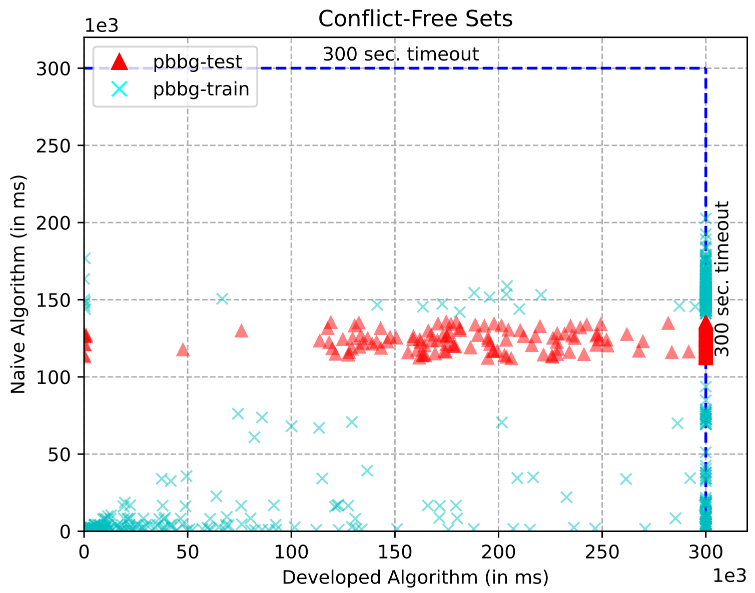

Figure 23: Results for the CPU runtime of the interviewing algorithms for eliciting conflict-free sets.

This is also confirmed by Table 5, which shows the number of timeouts and the average runtime in nanoseconds. Since the diversity function was not used as a stop condition, there is no entry in the table for this condition. It is clear from the table that there is a high number of timeouts for the newly developed algorithm. This can be seen both in the dataset pbbg-train with the very small instances with $5 \leq |Arg| \leq 25$ and in the instances from pbbg-test with $|Arg| = 25$. In pbbg-train, timeouts occurred from $|Arg| \geq 18$ arguments. Although only with 24 arguments did more than two-thirds of all instances time out.

dataset	**algorithm**	**# timeouts**	**avg time (ns)**	$\#\Delta_{cf} = \lvert\mathbb{S}\rvert$
pbbg-train	naïve	0	$5.78 \cdot 10^{10}$	-
	developed	436	$1.44188 \cdot 10^{11}$	-
pbbg-test	naïve	0	$1.23347 \cdot 10^{11}$	-
	developed	881	$2.84769 \cdot 10^{11}$	-

Table 5: Performance of the naïve and developed algorithms for the conflict-free sets on datasets pbbg-train and pbbg-test.

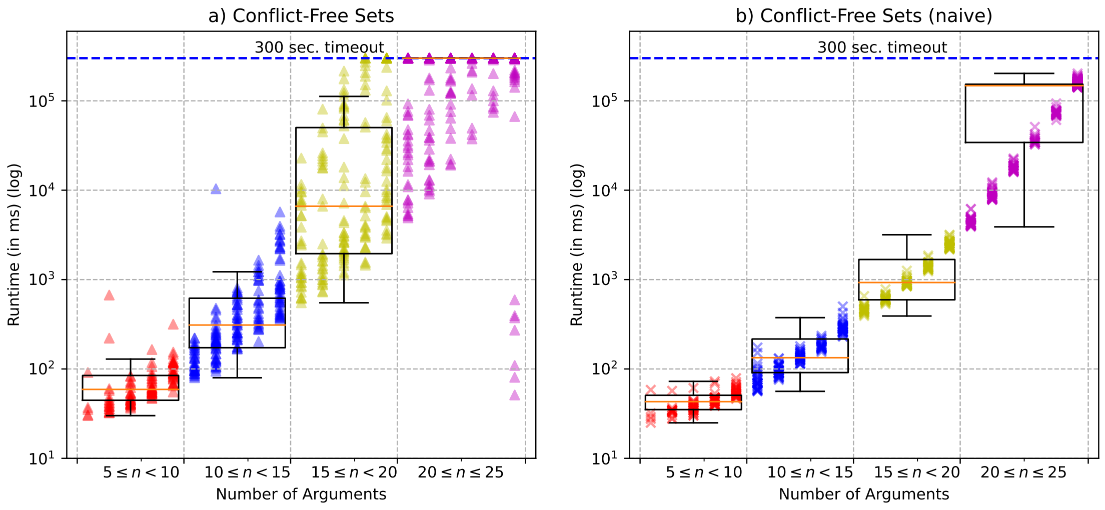

Figure 24: Plot of the runtimes per number of arguments of the instances with four data bins. On the left side (a) is the Algorithm 4 for conflict-free sets developed in the Master's thesis and on the right side (b) the naïve approach. Note that in the left plot the inter-quartile range for the fourth data bin $20 \leq n \leq 25$ is close to the upper limit due to the many timeouts and therefore difficult to see.

Next, the dataset pbbg-train should be used to check whether there is a relationship between the number of arguments of an instance and the CPU runtime. First, the linear relationship should be checked. The coefficient R^2 for a linear test is 0.702. The left plot in Figure 24 represents the data for the Algorithm 4 developed. From it, we can speculate that the relationship is non-linear. With a mean absolute percentage error of 18.07% for a 19th degree polynomial function, an acceptable prediction can be made. However, the R^2 value here is also only 0.786, which is nevertheless sufficient for a strong correlation. So a non-linear correlation can be confirmed, but just the more arguments are added, the greater is the dispersion of the runtime.

For the naïve variant we also test both a linear and a non-linear correlation. For the linear correlation we find the value 0.641 for the coefficient of determination R^2. The right plot in Figure 24 suggests that there is a non-linear correlation. For a polynomial function of the 18th degree, an R^2 value of 0.994 can be determined and a MAPE of only 0.08%. So that a strong correlation exists in this case.

From the preceding data, it can be assumed that the naïve algorithm performs significantly better. The number of timeouts is particularly high in pbbg-test for the algorithm developed in this Master's thesis. The hypothesis that the naïve algorithm has a better performance is therefore tested with the Wilcoxon signed-rank test. The one-sided test with $\tilde{x}_N$ for the CPU runtime in nanoseconds of the naïve algorithm and $\tilde{x}_D$ for the CPU runtime in nanoseconds of the previously developed algorithm has the hypothesis $H_0 : \tilde{x}_D \geq \tilde{x}_N$. For the two data sets, this gives:

- pbbg-train: the p-value for this data set is $p < 0.05$ with $p \approx 1.79 \cdot 10^{-156}$. The sum of the ranks of the difference T^+ is 493475. There is a clear strong effect size with $r = 0.986$.
- pbbg-test: Again, the p-value is $p < 0.05$ with $p \approx 4.529 \cdot 10^{-164}$ and for $T^+ = 499397$ and a strong effect size of $r = 0.998$.

The hypothesis can therefore be accepted and the better performance of the naive algorithm can be considered statistically significant.

5.2.2. Admissible Sets

Let us continue with the admissible sets. Here, too, Figure 25 represents a scatter plot in which the runtime of Algorithm 5 can be read on the X-axis and that of the naïve algorithm on the Y-axis. Again, both datasets pbbg-test (with red triangles) and pbbg-train (with cyan X-markers) are depicted.

In contrast to the conflict-free sets earlier, it can be seen from the plot that far fewer instances timed out for the pbbg-train dataset. The runtimes also seem to be closer together in general. For pbbg-test, on the other hand, a high number of timeouts can be seen, but also some instances in which the naïve algorithm was even slower. But here, too, the overall picture is that the naïve algorithm performed better altogether.

This is supported by the data from Table 6. The data shows only 63 timeouts for pbbg-train. For pbbg-test, on the other hand, over 70% ran into the five-minute timeout again. However, the average runtimes are closer overall than for the conflict-free sets (cf. Table 5).

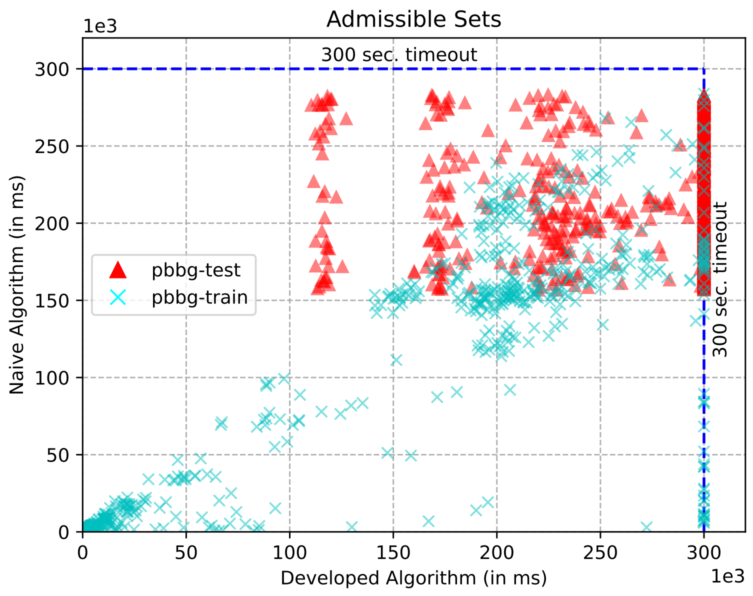

Figure 25: Results for the CPU runtime of the interviewing algorithms for eliciting admissible sets.

We continue by checking the relationship between the number of arguments of an instance and the CPU runtime using the dataset pbbg-train. For this purpose, a linear relationship must first be checked for the developed Algorithm 5. With a coefficient of determination R^2 of 0.613, the linear relationship is rather unlikely. This is supported by $MAPE = 159.12\%$, which is an unacceptable error. The left plot in Figure 26 shows the algorithm. A non-linear relationship can be assumed from the figure. With a polynomial function of the 23rd degree, a mean absolute percentage error of only 3.507% can be obtained and the value 0.82 can be determined for R^2, indicating a non-linear correlation.

dataset	**algorithm**	**# timeouts**	**avg time (ns)**	**#$\Delta_{ad} = \|\mathbb{S}\|$**
pbbg-train	naïve	0	$6.56 \cdot 10^{10}$	-
	developed	63	$9.26 \cdot 10^{10}$	-
pbbg-test	naïve	0	$2.18602 \cdot 10^{11}$	-
	developed	713	$2.72833 \cdot 10^{11}$	-

Table 6: Performance of the naïve and developed algorithms for the admissible sets on datasets pbbg-train and pbbg-test.

Let us now also check the naïve algorithm with regard to the number of arguments and the relationship to the runtime. Again, the regression analysis for a linear correlation only yields an R^2 value of 0.603, which suggests a non-linear relationship. If we look at the right plot in Figure 26, this assumption can be confirmed visually. Using a polynomial function of the 22nd degree, $R^2 = 0.942$ can be determined and a MAPE of 0.203%. A non-linear correlation also exists here.

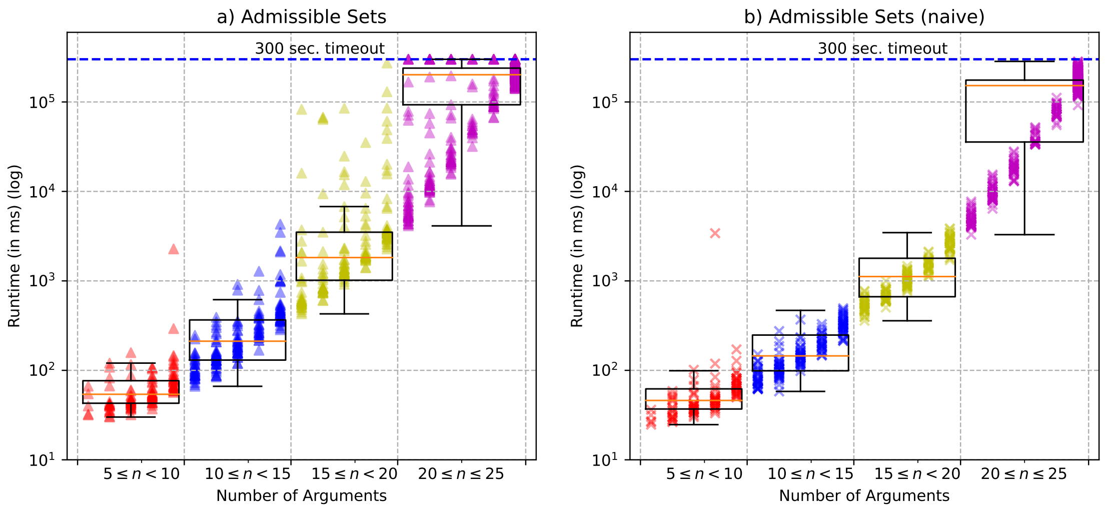

Figure 26: Plot of the runtimes per number of arguments of the instances with four data bins. On the left side (a) is the Algorithm 5 for admissible sets developed in the Master's thesis and on the right side (b) the naïve approach.

Again, we will statistically test whether the assumption that the naïve algorithm is faster holds. For this purpose, a one-sided Wilcoxon signed-rank test is carried out once again. Likewise, $\tilde{x}_N$ and $\tilde{x}_D$ stand for the CPU runtimes of the naïve algorithm and the algorithm developed in the Master's thesis, respectively, in nanoseconds. The hypothesis is again $H_0 : \tilde{x}_D \geq \tilde{x}_N$, i.e. the runtimes of the naïve algorithm are shorter. The results for the two datasets are as follows:

- pbbg-train: The test yields a p-value with $p \approx 3.22 \cdot 10^{-108}$, making $p < 0.05$. The sum of the ranks with positive difference T^+ is 451852. A strong effect size results from $r = 0.90$.
- pbbg-test: Again, $p < 0.05$ is clear with $p \approx 3.808 \cdot 10^{-99}$. The calculation for T^+ results in 443029 and a strong $r = 0.885$.

Thus, even for admissible sets, it can be statistically significantly demonstrated that the naïve algorithm has a better runtime.

5.2.3. Complete Semantics

Next, we look at the benchmark data for complete semantics. For this purpose, the scatter plot in Figure 27 shows the runtime of Algorithm 6 on the X-axis and that of the naïve algorithm on the Y-axis for the instances of the two datasets pbbg-test (with red triangles) and pbbg-train (with cyan X-markers).

It is immediately noticeable that none of the instances ran into a timeout. Also, as a first visual observation, it can be taken that the instances seem to be very close to each other in their runtime. From the distribution of the triangles and X-markers it can be seen that sometimes the naïve algorithm and sometimes the algorithm developed in this thesis were faster.

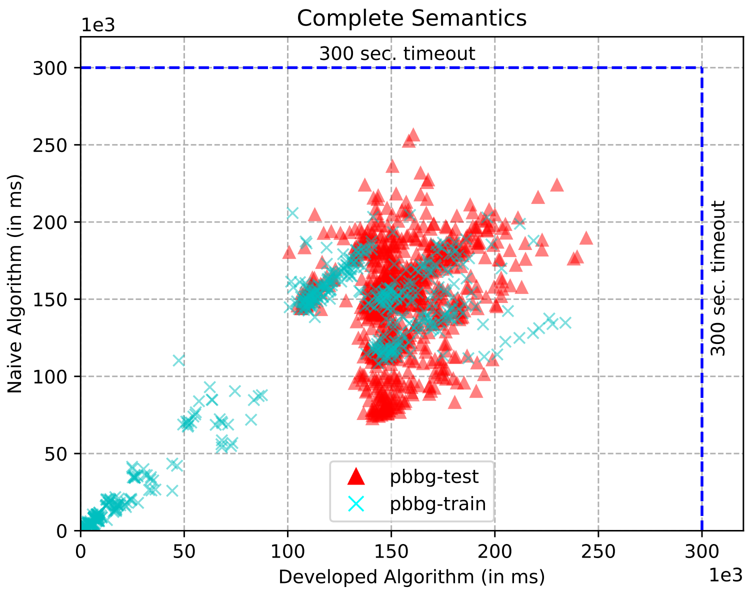

Figure 27: Results for the CPU runtime of the interviewing algorithms for eliciting under the complete semantics.

If we look at the data from Table 7, this first impression is supported. For pbbg-train, the dataset with the mixed-sized instances, the runtime of the developed algorithm is slightly faster on average, while for pbbg-test the naive algorithm was faster. A closer look at the benchmark results show that for pbbg-train in 520 instances the developed algorithm was faster while in pbbg-test the naive one with 511 instances was faster. No apparent property of the elicited extensions can be determined that leads to such an outcome.

First, the relationship between the number of arguments of an instance and the CPU runtime for the dataset pbbg-train is to be checked again. Therefore, a linear relationship between the number of arguments and the runtime is tested. For this, the coefficient of determination R^2 is only 0.606 and $MAPE$ can also be calculated with 131.09% for Algorithm 6. If we look at the left plot in Figure 28, we can again assume a non-linear relationship. For a polynomial function of the 18th degree, $MAPE = 0.176\%$ can be calculated and a coefficient of $R^2 = 0.945$. As expected, a non-linear correlation is present here, which was to be expected with the algorithmic design.

dataset	algorithm	# timeouts	avg time (ns)	$\#\Delta_{co} = \|\mathbb{S}\|$
pbbg-train	naïve	0	$5.67 \cdot 10^{10}$	-
	developed	0	$5.45 \cdot 10^{10}$	0
pbbg-test	naïve	0	$1.4841 \cdot 10^{11}$	-
	developed	0	$1.53844 \cdot 10^{11}$	0

Table 7: Performance of the naïve and developed algorithms for the complete semantics on datasets pbbg-train and pbbg-test.

The same can be determined for the naïve algorithm. Again, based on the right plot in Figure 28, it can be assumed that there is no linear correlation. An R^2 value of 0.625 confirms this. For the non-linear regression, $R^2 = 0.97$ can be found with $MAPE = 0.21\%$ for a polynomial function of the 23rd degree. So, again, a non-linear correlation is present. This is to be anticipated on the basis of the structure of the algorithm.

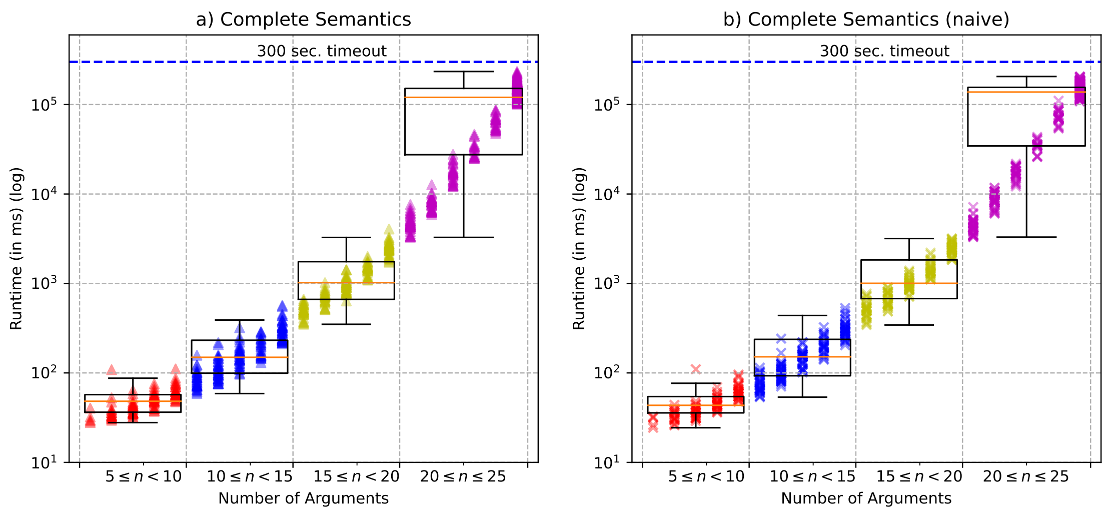

Figure 28: Plot of the runtimes per number of arguments of the instances with four data bins. On the left side (a) is the Algorithm 6 for complete semantics developed in the Master's thesis and on the right side (b) the naïve approach.

Finally, we test the hypothesis that the developed algorithm is faster or as fast as the naïve algorithm. For this purpose, we again use a one-sided Wilcoxon signed-rank test with $\tilde{x}_D$ for the CPU runtime of the algorithm developed in the Master's thesis in nanoseconds and $\tilde{x}_N$ analogously for the naïve algorithm. The hypothesis posed H_0 is $\tilde{x}_D \geq \tilde{x}_N$. For our two pbbg datasets, we obtain the following results:

- pbbg-train: the p-value corresponds to $p \approx 0.0013$, which is $p < 0.05$ below the necessary limit. The hypothesis is therefore to be accepted. For the sum of the differences above zero, $T^+ = 222737$ is and therefore an acceptable effect size of $r = 0.445$ is obtained.
- pbbg-test: Again, with $p \approx 0.000478$ the limit of $p < 0.05$ is undercut and the hypothesis is to be accepted. T^+ is calculated here to 280422 with a strong effect size of $r = 0.56$.

Thus, for $\sigma = co$, it can be said that the developed algorithm does not perform worse but tends to perform better than the naïve algorithm.

5.2.4. Grounded Semantics

We continue with grounded semantics. Due to the design of Algorithm 7, we expect a significantly better runtime of the developed algorithm compared to the naïve approach. Figure 29 shows a scatter plot with the runtimes of Algorithm 7 on the X-axis and the naïve algorithm on the Y-axis for the two datasets pbbg-test (with red triangles) and pbbg-train (with cyan X-markers).

This plot is special compared to the previous ones. For the developed algorithm we have relatively short runtimes, while for the naïve algorithm we have a high timeout share. This observed information is also strengthened by Table 8. For the dataset pbbg-train with the instances between 5 and 25 arguments, 360 timeouts have already occurred for the naïve algorithm. The runtime is also several orders of magnitude higher on average. In fact, the timeouts for the naïve approach occurred with instances of 24 argument size and larger. This is supported by the data from pbbg-test, where 74.3% of all instances, rounded up, ran into a timeout. It should be noted that the diversity function Δ_{gr} was not used and therefore no entries are available.

Next, we check the correlation between the number of arguments of an instance and the runtime using the dataset pbbg-train. From the two plots in Figure 30, we can assume for the left plot, which represents the developed algorithm, that there is a linear correlation or if it is non-linear, then the polynomial has a low degree. For the right plot, on the other hand, which represents the naïve algorithm, we assume that we are again finding a higher degree polynomial function - justified by the exhaustive search.

We therefore first begin to test a linear relationship for the developed algorithm. The R^2 value calculates to 0.0014, however, also with a low $MAPE$ of only 1.005%.

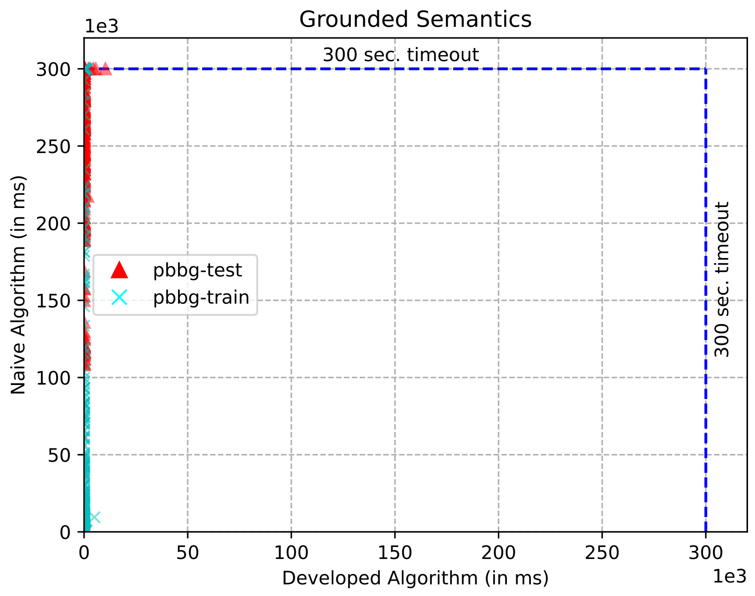

Figure 29: Results for the CPU runtime of the interviewing algorithms for eliciting under the grounded semantics.

If we all take a polynomial function of the second degree, the mean absolute percentage error is slightly higher at 1.006% with also $R^2 \approx 0.0018$. Polynomials of a higher degree do not bring any improvement, so we conclude that there is a linear relationship since on the one hand the structure of Algorithm 7 suggests this and on the other hand the forecast values and thus also the $MAPE$ value are satisfactory. It can therefore be assumed that the instances are too small and the run times too fast so that no correlation is indicated.

dataset	**algorithm**	**# timeouts**	**avg time (ns)**	$\#\Delta_{gr} = \lvert\mathbb{S}\rvert$
pbbg-train	naïve	360	$1.20632 \cdot 10^{11}$	-
	developed	0	$5.68 \cdot 10^{7}$	-
pbbg-test	naïve	743	$2.89101 \cdot 10^{11}$	-
	developed	0	$7.53 \cdot 10^{7}$	-

Table 8: Performance of the naïve and developed algorithms for the grounded semantics on datasets pbbg-train and pbbg-test.

For the naïve algorithm, despite the assumption that it is a non-linear correlation, we still first check R^2 for a linear function. We calculate this as $R^2 \approx 0.697$ which is not sufficient for a strong correlation. With a polynomial function of the 25th degree, on the other hand, we obtain a R^2 value of approximately 0.992 and $MAPE = 0.27\%$, which makes it clear that the number of arguments correlates clearly non-linearly with the running time, as it was to be expected.

We conclude our analysis for the grounded semantics in our first experiment with the testing of the hypothesis: The runtime of the developed algorithm is faster or at least as fast as the naïve approach.

For this, we again use a one-sided Wilcoxon signed-rank test with $\tilde{x}_D$ for the CPU runtime of the developed algorithm and $\tilde{x}_N$ for the naïve algorithm. We therefore formulate $H_0 : \tilde{x}_N \geq \tilde{x}_D$. We obtain the following results for the two datasets considered:

- pbbg-train: For this dataset the p-value is $p \approx 1.778 \cdot 10^{-163}$ and hence the developed algorithm is faster. The sum of the differences over zero calculates to only $T^+ = 498393$ and gives an effect size of $r = 0.997$.
- pbbg-test: Also for this second dataset we calculate $p \approx 1.663 \cdot 10^{-165}$ and $T^+ = 500500$ and therefore $r = 1$. We can therefore accept the hypothesis.

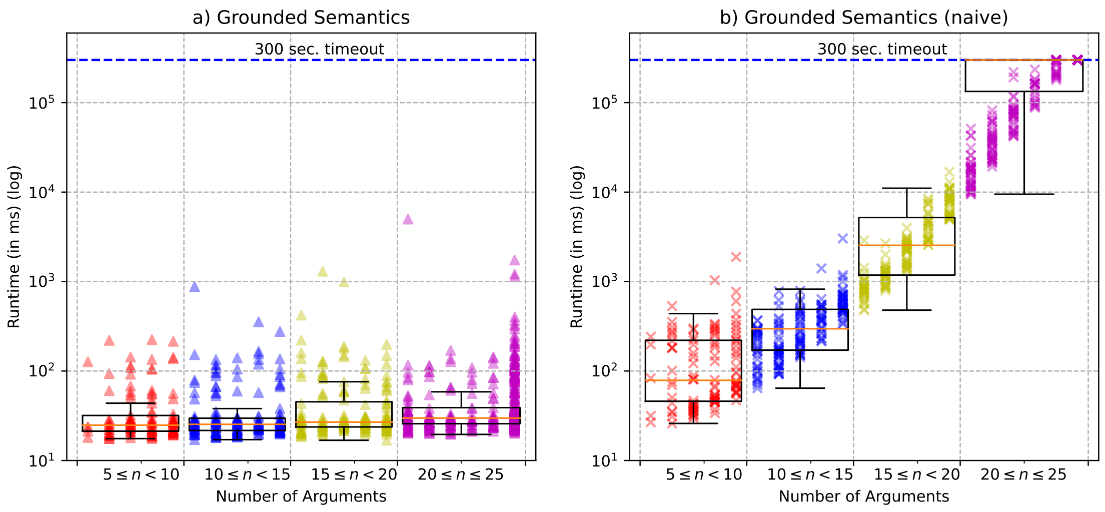

Figure 30: Plot of the runtimes per number of arguments of the instances with four data bins. On the left side (a) is the Algorithm 7 for grounded semantics developed in the Master's thesis and on the right side (b) the naïve approach.

In summary, the Algorithm found here is significantly faster than the naïve approach.

5.2.5. Preferred Semantics

As the penultimate semantics, we check the preferred semantics. Performance expectations are not too high for the developed algorithm as well as for the naïve algorithm. This is because Ver_{pr} is in the complexity class coNP.

The result of the benchmark is shown in Figure 31. Here the runtime of Algorithm 8 is shown on the X-axis and that of the naïve algorithm on the Y-axis. The instances of the two datasets are shown as usual with red triangles for pbbg-test and cyan X-markers for pbbg-train.

It is immediately noticeable that a high number of timeouts can be seen. However, some instances of pbbg-test are also faster for the developed algorithm than for the naïve one. Nevertheless, the majority of instances that did not time out seem to be faster in the naïve algorithm. This observation is supported by the data from Table 9. The naïve algorithm does slightly better for the dataset pbbg-train with 320 timeouts than the developed one with 373 and the runtime is also slightly faster. A similar picture emerges for pbbg-test, although the number of timeouts for the naïve algorithm is significantly better here with 513 to 624. The same applies to the runtime.

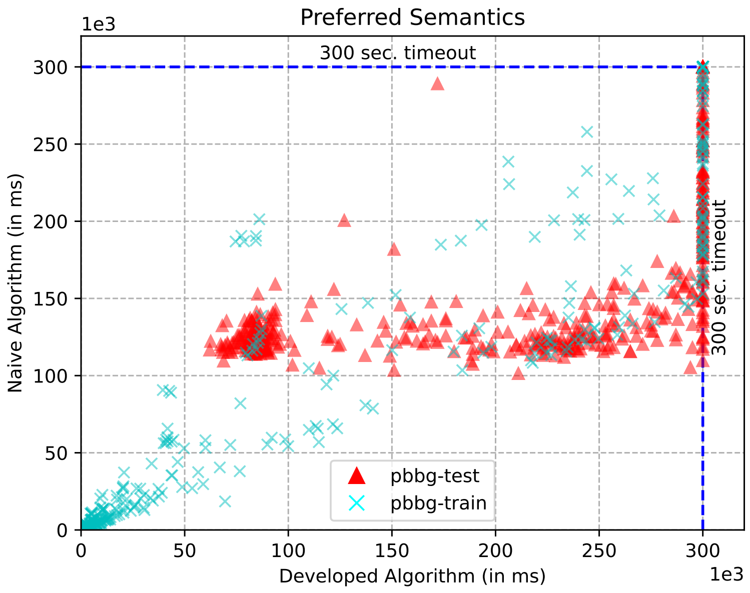

Figure 31: Results for the CPU runtime of the interviewing algorithms for eliciting under the preferred semantics.

dataset	algorithm	# timeouts	avg time (ns)	#$\Delta_{pr} = \|\mathbb{S}\|$
pbbg-train	naïve	320	$1.25581 \cdot 10^{11}$	-
	developed	373	$1.35035 \cdot 10^{11}$	0
pbbg-test	naïve	513	$2.2642 \cdot 10^{11}$	-
	developed	624	$2.57558 \cdot 10^{11}$	0

Table 9: Performance of the naïve and developed algorithms for the preferred semantics on datasets pbbg-train and pbbg-test.

Next, we check the relationship between the number of arguments and the runtime using the dataset pbbg-train. Figure 32 serves as a visualisation of the results. On the left side, we see the plot for the developed algorithm. Based on the box plot for the databins, we can assume a non-linear relationship. On the right side of the figure is the box plot of the naïve algorithm. Here too, as for the previous semantics, we assume a non-linear relationship.

We start with the developed algorithm and nevertheless check a linear relationship first. As expected, the R^2 value is 0.613, which is too low to confirm a linear relationship. The $MAPE$ value is also 82.84%, which is why we exclude a linear relationship. With a polynomial function of the 26th degree, however, we find a coefficient R^2 of 0.70 and a $MAPE$ of 6.64%, so that we can assume that there is a non-linear relationship, which even worsens with each argument more. The R^2 value is below a threshold of 0.75 for which we previously assumed a strong correlation. We suspect that this is due to the high number of timeouts.

We check the same again for the naïve algorithm. Here, too, no linear correlation can be seen with an R^2 coefficient of 0.585. Also $MAPE$ is 147.26%. Again, for a 26th degree polynomial function, $MAPE$ is 7.52%, which we consider sufficient in this case. The R^2 value is even only 0.672 in this case. Again, we suspect that no strong correlation emerges because of the high number of timeouts in the data.

Finally, we check whether the hypothesis that the naïve algorithm is better, based on the read data, really holds. Again, we use the Wilcoxon signed-rank test with $\tilde{x}_D$ for the CPU runtime of the developed algorithm and $\tilde{x}_N$ for that of the naïve algorithm. Our hypothesis H_0 is that $\tilde{x}_D \geq \tilde{x}_N$. We check this for both datasets and get the following results:

- pbbg-train: With a p-value of approximately $5.277 \cdot 10^{-34}$ and the sum of the differences above zero being $T^+ = 360125$ resulting in a strong effect size of $r = 0.738$, the hypothesis can be confirmed for this dataset.
- pbbg-test: Also for this dataset we obtain $p < 0.05$ with $p \approx 1.306 \cdot 10^{-39}$. T^+ is calculated to be 370086 and $r = 0.739$. Therefore, the hypothesis is confirmed here as well.

Overall, as expected, the performance of the naïve algorithm is comparatively better, but nonetheless characterised by a high number of timeouts. For the developed

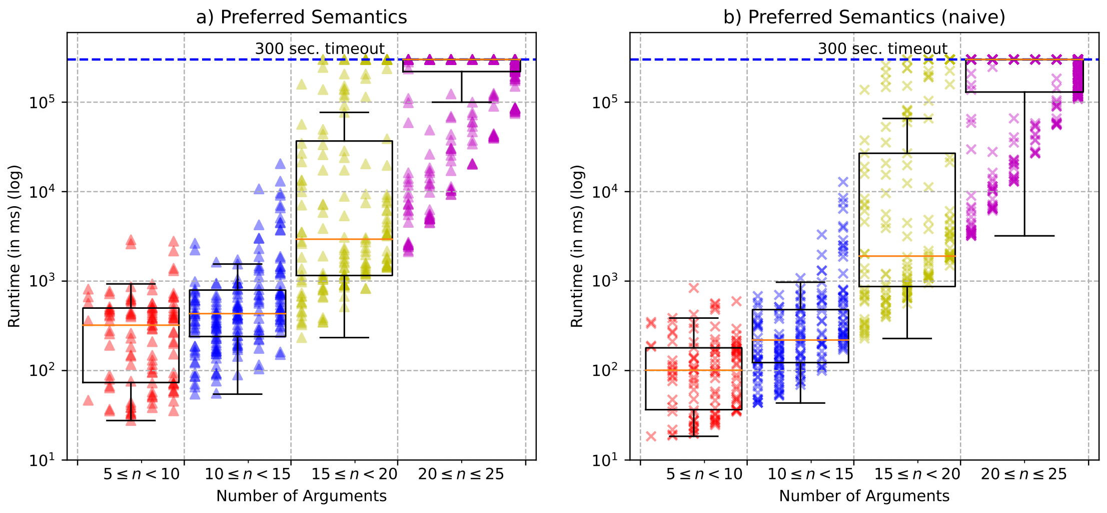

Figure 32: Plot of the runtimes per number of arguments of the instances with four data bins. On the left side (a) is the Algorithm 8 for preferred semantics developed in the Master's thesis and on the right side (b) the naïve approach.

algorithm, the check for incomparability and conflict sensitivity are presumably too computationally intensive, so that the naïve approach has an advantage.

5.2.6. Stable Semantics

Let us conclude our observations on this experiment with the stable semantics. In Figure 33, on the one hand, it can be seen for pbbg-test (red triangles) that the developed Algorithm 9 was either equally fast as the naïve algorithm or significantly slower. The same can be said about the data set pbbg-train with, among other things, smaller AFs. Here, too, it can be seen from the cyan x markers that the naïve algorithm was mostly faster and otherwise equally fast.

dataset	algorithm	# timeouts	avg time (ns)	$\#\Delta_{st} = \lvert\mathbb{S}\rvert$
pbbg-train	naïve	0	$4.45 \cdot 10^{10}$	-
	developed	0	$6.94 \cdot 10^{10}$	0
pbbg-test	naïve	0	$1.26793 \cdot 10^{11}$	-
	developed	3	$1.90432 \cdot 10^{11}$	0

Table 10: Performance of the naïve and developed algorithms for the stable semantics on datasets pbbg-train and pbbg-test.

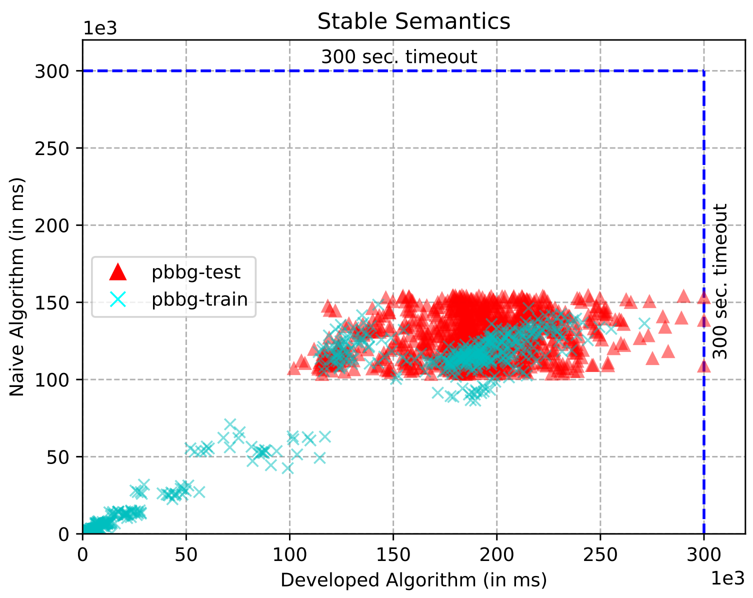

Figure 33: Results for the CPU runtime of the interviewing algorithms for eliciting under the stable semantics.

This observation is also supported by Table 10. This shows that in three cases pbbg-test even ran into the five-minute timeout. Furthermore, it can be seen for both pbbg-test and pbbg-train that the naïve algorithm was faster on average.

Using the dataset pbbg-train, which contains instances of different numbers of arguments, it is checked whether a correlation between the runtimes is recognisable. For the developed stable semantics algorithm, the coefficient of determination R^2 for a simple linear regression of the number of arguments to time is only 0.61. The left plot in Figure 34 suggests a polynomial relationship. This can be confirmed, as for a polynomial of the 11th degree the R^2 value is 0.953 with a mean absolute percentage error of 0.726%. Thus, there is a polynomial correlation between the number of arguments and the running time.

For the naïve algorithm, shown in the right plot in Figure 34, a simple linear regression results in $R^2 = 0.633$. Here, too, a polynomial relationship between the number of arguments and time can be hypothesised. For a polynomial function of the 12th degree, an R^2 value of 0.986 can be obtained with a mean absolute percentage error of 0.437%. Hence, there is a polynomial relationship between the number of arguments of the AF and the runtime.

Derived from these results, we test the hypothesis that the naïve algorithm has a faster runtime than the algorithm developed in this Master's thesis. For the one-sided Wilcoxon signed-rank test with $\tilde{x}_N$ for the CPU runtime in nanoseconds of the naive algorithm and $\tilde{x}_D$ for the CPU runtime in nanoseconds of the previously developed algorithm. The hypothesis H_0 is $\tilde{x}_D \geq \tilde{x}_N$.

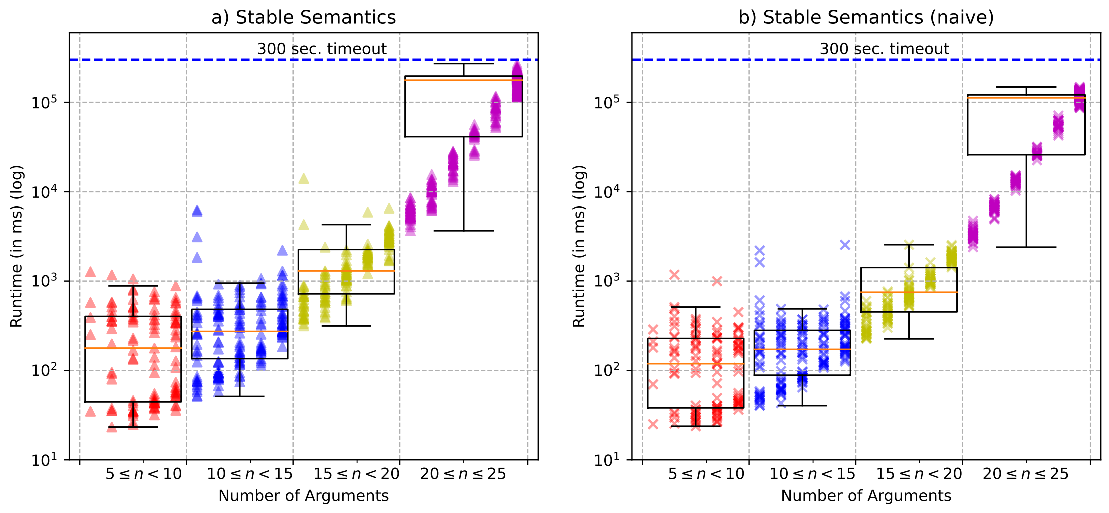

Figure 34: Plot of the runtimes per number of arguments of the instances with four data bins. On the left side (a) is the Algorithm 9 for stable semantics developed in the Master's thesis and on the right side (b) the naïve approach.

For the two datasets, this yields:

- pbbg-train: The sum of the difference over zero calculated in this case is $T^+ = 477743$ and the p-value of $p \approx 3.54 \cdot 10^{-138}$ which is $p < 0.05$ and hence confirms that the naïve algorithm performed better with a strong effect size of 0.955.
- pbbg-test: Also for this dataset the p-value is $p \approx 1.94 \cdot 10^{-164}$ and therefore confirms with $p < 0.05$ that the naïve algorithm performed better. For T^+, the calculated value is 499679 and therefore $r = 0.998$.

Thus, in this case, the naïve algorithm is statistically significantly faster.

5.2.7. Summary and Answering the Questions

Let us summarise the results. Based on the benchmarks, it was shown that for $\sigma = \{cf, ad, st, pr\}$ the naïve algorithm performed better. In the case of complete semantics, a slightly better picture emerged for the developed algorithm, which was faster in some instances and frequently at least as fast overall. Unsurprisingly, for the grounded semantics, the developed algorithm was significantly faster. Rather than being based on an exhaustive search of all verified extensions, it is based on the acceptance of arguments.

σ	**pbbg-train**	**pbbg-test**
cf	naïve	naïve
ad	naïve	naïve
co	developed	developed
gr	developed	developed
pr	naïve	naïve
st	naïve	naïve

Table 11: Determine for the datasets whether the naïve algorithm or the developed algorithm performed statistically better according to the Wilcoxon signed-rank test.

Since the naïve algorithm also had to deal with many timeouts in the grounded semantics and both algorithms in case of the preferred semantics, two findings from Oikarinen and Woltran's work on strong equivalence [38] can be included. Namely, if $ad(F_A) = ad(F_E)$ holds, then $pr(F_A) = pr(F_E)$ also holds. Furthermore, it is also true that if $co(F_A) = co(F_E)$, then $pr(F_A) = pr(F_E)$ as well as $gr(F_A) = gr(F_E)$. Therefore, the algorithms for $\sigma \in \{ad, co\}$ can be used to elicit an extension-set for $\sigma = pr$ respectively in the second case $\sigma \in \{pr, gr\}$ as well as to generate an AF F_E that has the same extension set as F_A under the preferred or grounded semantics. To get the desired extension set, the corresponding extensions only have to be selected from the admissible respectively the complete extension set (cf. Subsection 2.2 on extension-based semantics).

Finally, we answer the two questions. The first question was: *How well do the developed algorithms perform in terms of runtime compared to the naïve algorithm?*

This has already been described before and is illustrated once again in Table 11. All in all, only the developed algorithms for complete and grounded semantics are faster than the naïve approach in the case of $Agent^{dec,sem}_{\neg\mathbb{E}}$.

The second question was: *Do the number of arguments and running time correlate?* This question can be answered positively. However, with the exception of the developed algorithm for grounded semantics, all algorithms, whether developed or naïve, show that the correlation is non-linear. This was to be expected, however, since the algorithms implement an exhaustive search and therefore have an exponential running time in the worst case if no stop condition applies. This is never the case with the naïve algorithm and with the others it never occurred in the test data sets that the diversity function Δ_σ with the theoretical maximum was equal to the cardinality of the extension set.

Overall, it can be concluded from the first experiment that of the algorithms developed, only those of the complete semantics (cf. Algorithm 6) and the grounded semantics (cf. Algorithm 7) are feasible for smaller AFs, although the runtime was already in the range of minutes and thus they are not suitable for larger instances.

5.3. Analyses on an Agent that Enumerated the Extensions Beforehand

In this second experiment, the aim is to compare the different configurations of an agent that can only answer semantic decision questions. For this purpose, we compare the type of the stateless agent $Agent_{\neg\mathbb{E}}^{dec,sem}$, which did not enumerate the extensions before and cannot remember this information, with one that has a state and already enumerated the extensions, thus an agent of the type $Agent_{\mathbb{E}}^{dec,sem}$.

The $Agent_{\mathbb{E}}^{dec,sem}$ is implemented in such a way that all extensions are enumerated first and the interviewer can ask his questions afterwards. Since the aim is not to measure the enumeration performance of a solver, the runtime measurement only starts from the point at which the interviewer asks his first question.

We again use the 1000 randomly chosen instances from pbbg-train with 5 to 25 arguments. As there is a significant number of instances with 25 arguments in pbbg-train, we will not run experiments with pbbg-test. For the composition of the instances of pbbg-train see Subsection 5.2.

Next, we formulate the questions we want to answer with this experiment:

- *Which of the algorithms are on average faster given an $Agent_{\mathbb{E}}^{dec,sem}$?*
  To answer this, we compare the arithmetic mean of the running times of both the developed and naïve algorithms for each data bin of arguments in the dataset. For this purpose, we form four data bins $5 \leq |Arg| < 10$, $10 \leq |Arg| < 15$, $15 \leq |Arg| < 20$ and $20 \leq |Arg| \leq 25$. We also examine changes in terms of the number of timeouts for these data bins.

- *In this scenario, are the developed algorithms faster than the naïve algorithm?*
  To answer this question, we again compare the runtimes and test the runtime hypotheses using the Wilcoxon signed-rank test.

What follows is a brief elaboration of the results of the second experiment for the semantics $\sigma = \{cf, ad, co, gr, pr, st\}$. Thereafter, we answer the questions posed collectively for all semantics.

5.3.1. Conflict-Free Sets

We start with the conflict-free sets. Two comparisons of the runtimes of Algorithm 4 and the naïve approach are illustrated in Figure 35. The left scatter plot is for the developed algorithm and the right is for the naïve approach. In the left one, it can be seen that the runtimes seem to be relatively the same between the agent types. No improvement can be assumed here. From the right plot, it can be seen that even with the $Agent_{\mathbb{E}}^{dec,sem}$ there was a deterioration, as there a high number of timeouts now entered there.

To check these observations, we first look at the collected data from Table 12. These support the observation because basically the same orders of magnitude are recorded for both the developed and the naïve approach for each agent type. The number of timeouts for Algorithm 4 is hardly different for both types, while for the naïve approach, they are to the disadvantage of the variant with the agent $Agent_{\mathbb{E}}^{dec,sem}$ with 128 additional timeouts.

Based on these findings, we hypothesise that even in the case of $Agent_{\mathbb{E}}^{dec,sem}$ the naïve approach has a lower runtime, even if it became worse. To do this, we formulate $H_0 : \tilde{x}_D \geq \tilde{x}_N$ for the $Agent_{\mathbb{E}}^{dec,sem}$ with $\tilde{x}_D$ the runtime of Algorithm 4 and $\tilde{x}_N$ that for the naïve algorithm. We check these using the one-sided Wilcoxon signed-rank test. Again, our boundary for acceptance is a p-value of $p < 0.05$. This is passed with $p \approx 3.968 \cdot 10^{-150}$ and a sum of ranks of differences above zero with $T^+ = 488416$. Also giving an effect size of $r = 0.976$.

In summary, we see no strong improvement and rather even a worsening. This is because, on the one hand, the credulous acceptance problem is solved via simple attack verification as well as the verification problem is solved via attacks of the arguments within the extension to be verified at $Agent_{\neg\mathbb{E}}^{dec,sem}$. Depending on the extension and the structure of the AF, this can be faster than searching the pre-calculated extensions.

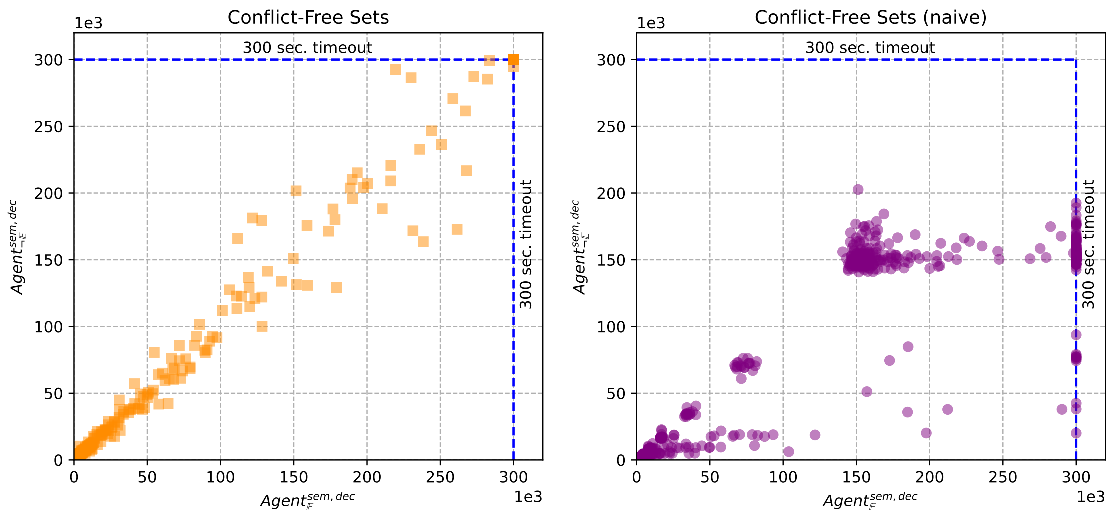

Figure 35: Plot of the runtimes in milliseconds of the two agent types. On the left side is the scatter plot of Algorithm 4 for conflict-free sets and on the right side the naïve approach.

5.3.2. Admissible Sets

We continue with the admissible sets. Here, too, we first analyse the effects of the agent type. Figure 36 shows two scatter plots. The left one shows the relationship between the two agent types for the developed Algorithm 5. Here it is noticeable that there is a high number of timeouts for $Agent_{\mathbb{E}}^{dec,sem}$ in relation to the other agent. The right scatter plot shows this for the naïve approach. Here there seem to be a few timeouts, but also cases where the $Agent_{\mathbb{E}}^{dec,sem}$ was faster than the agent type $Agent_{\neg\mathbb{E}}^{dec,sem}$.

Algorithm 4 for $\sigma = cf$	$\mathbf{5 \leq \|n\| < 10}$	$\mathbf{10 \leq \|n\| < 15}$	$\mathbf{15 \leq \|n\| < 20}$	$\mathbf{20 \leq \|n\| \leq 25}$
$\overline{time}$ for $Agent_{\mathbb{E}}^{dec,sem}$	$4.02 \cdot 10^7$	$5.99 \cdot 10^8$	$5.36 \cdot 10^{10}$	$2.58 \cdot 10^{11}$
$\overline{time}$ for $Agent_{\neg\mathbb{E}}^{dec,sem}$	$7.52 \cdot 10^7$	$6.27 \cdot 10^8$	$5.39 \cdot 10^{10}$	$2.58 \cdot 10^{11}$
$\sum$ T/O $Agent_{\mathbb{E}}^{dec,sem}$	0	0	37	400
$\sum$ T/O $Agent_{\neg\mathbb{E}}^{dec,sem}$	0	0	17	419
$\Delta_{timeouts}$	0	0	+20	−19

Naïve Algorithm for $\sigma = cf$	$\mathbf{5 \leq \|n\| < 10}$	$\mathbf{10 \leq \|n\| < 15}$	$\mathbf{15 \leq \|n\| < 20}$	$\mathbf{20 \leq \|n\| \leq 25}$
$\overline{time}$ for $Agent_{\mathbb{E}}^{dec,sem}$	$2.38 \cdot 10^7$	$1.60 \cdot 10^8$	$1.96 \cdot 10^9$	$1.58 \cdot 10^{11}$
$\overline{time}$ for $Agent_{\neg\mathbb{E}}^{dec,sem}$	$4.44 \cdot 10^7$	$1.61 \cdot 10^8$	$1.25 \cdot 10^9$	$1.11 \cdot 10^{11}$
$\sum$ T/O $Agent_{\mathbb{E}}^{dec,sem}$	0	0	0	128
$\sum$ T/O $Agent_{\neg\mathbb{E}}^{dec,sem}$	0	0	0	0
$\Delta_{timeouts}$	0	0	0	+128

Table 12: Comparison of runtimes in nanoseconds and timeouts between the two agent types for conflict-free sets. The top table shows the results for Algorithm 4 and the bottom for the naïve algorithm.

These observations are supported by Table 13. For the Algorithm 5, significantly more timeouts are recorded, especially in the data bin with instances of 20 arguments and above. However, it can be seen from the available data that timeouts already occur from 15 arguments. In the naïve algorithm, the $Agent_{\mathbb{E}}^{dec,sem}$ is also worse in terms of the number of timeouts (7 versus 0). However, the average runtimes are slightly better.

From these findings, we hypothesise the following for the comparison of the developed and the naïve algorithm for $Agent_{\mathbb{E}}^{dec,sem}$: The naïve algorithm has a lower runtime than the developed algorithm 5. We formulate this to $H_0 : \tilde{x}_D \geq \tilde{x}_N$ with $\tilde{x}_D$ the runtime of the Algorithm 5 and $\tilde{x}_N$ for the naïve approach under an $Agent_{\mathbb{E}}^{dec,sem}$. The hypothesis is confirmed by $p \approx 2.82 \cdot 10^{-165}$ and $T^+ = 500324$ giving $r \approx 1$, since $p < 0.05$.

While the naïve algorithm also performs better in terms of runtime in the case of $Agent_{\mathbb{E}}^{dec,sem}$, the results show that an agent $Agent_{\neg\mathbb{E}}^{dec,sem}$ otherwise performs better. Again, the verification via the attacks is apparently faster than the search in the enumerated extensions.

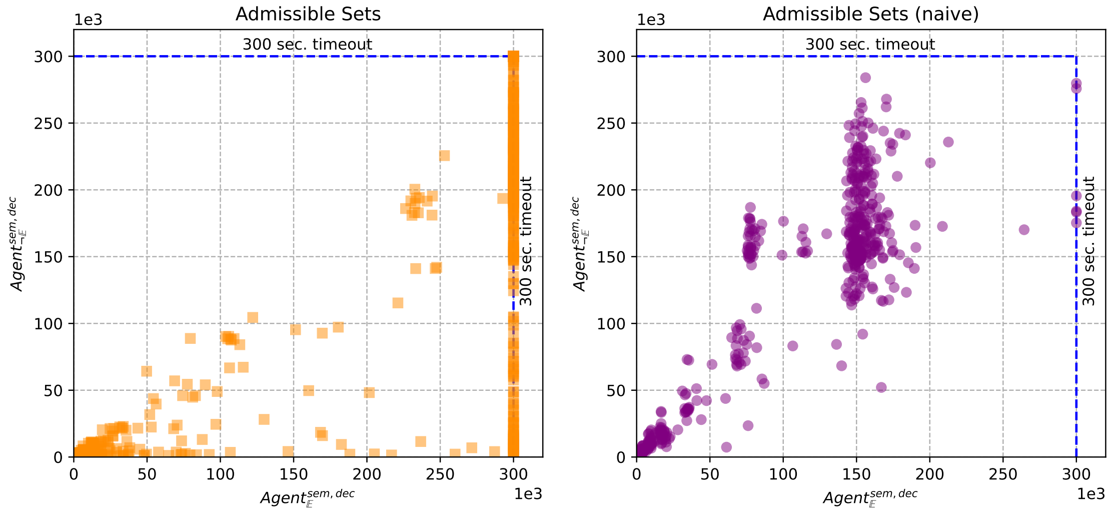

Figure 36: Plot of the runtimes in milliseconds of the two agent types. On the left side is the scatter plot of Algorithm 5 for admissible sets and on the right side the naïve approach.

Algorithm 5 for $\sigma = ad$	$\mathbf{5 \leq \lvert n \rvert < 10}$	$\mathbf{10 \leq \lvert n \rvert < 15}$	$\mathbf{15 \leq \lvert n \rvert < 20}$	$\mathbf{20 \leq \lvert n \rvert \leq 25}$
$\overline{time}$ for $Agent_{\mathbb{E}}^{dec,sem}$	$8.58 \cdot 10^{7}$	$1.78 \cdot 10^{9}$	$8.17 \cdot 10^{10}$	$2.54 \cdot 10^{11}$
$\overline{time}$ for $Agent_{\neg\mathbb{E}}^{dec,sem}$	$8.31 \cdot 10^{7}$	$3.33 \cdot 10^{8}$	$8.61 \cdot 10^{9}$	$1.75 \cdot 10^{11}$
$\sum$ T/O $Agent_{\mathbb{E}}^{dec,sem}$	0	0	37	409
$\sum$ T/O $Agent_{\neg\mathbb{E}}^{dec,sem}$	0	0	0	63
$\Delta_{timeouts}$	0	0	+37	+346

Naïve Algorithm for $\sigma = ad$	$\mathbf{5 \leq \lvert n \rvert < 10}$	$\mathbf{10 \leq \lvert n \rvert < 15}$	$\mathbf{15 \leq \lvert n \rvert < 20}$	$\mathbf{20 \leq \lvert n \rvert \leq 25}$
$\overline{time}$ for $Agent_{\mathbb{E}}^{dec,sem}$	$2.26 \cdot 10^{7}$	$1.30 \cdot 10^{8}$	$1.23 \cdot 10^{9}$	$1.08 \cdot 10^{11}$
$\overline{time}$ for $Agent_{\neg\mathbb{E}}^{dec,sem}$	$8.03 \cdot 10^{7}$	$1.81 \cdot 10^{8}$	$1.37 \cdot 10^{9}$	$1.25 \cdot 10^{11}$
$\sum$ T/O $Agent_{\mathbb{E}}^{dec,sem}$	0	0	0	6
$\sum$ T/O $Agent_{\neg\mathbb{E}}^{dec,sem}$	0	0	0	0
$\Delta_{timeouts}$	0	0	0	+6

Table 13: Comparison of runtimes in nanoseconds and timeouts between the two agent types for admissible sets. The top table shows the results for Algorithm 5 and the bottom for the naïve algorithm.

5.3.3. Complete Semantics

Next, we look at the runtime differences for the complete semantics. In Figure 37 can be seen for the agents $Agent_{\mathbb{E}}^{sec,dem}$ and $Agent_{\neg\mathbb{E}}^{sec,dem}$. The left scatter plot shows Algorithm 6 and the right one for the naïve algorithm. Comparing the two plots, it can be seen that the pattern is relatively similar. It can also be seen that for some instances the algorithms with an $Agent_{\neg\mathbb{E}}^{sec,dem}$ are faster.

Next, we look at whether the data collected supports this hypothesis derived from the plots. For this purpose, the formed data bins with their results are shown in Table 14.

Somewhat unexpected is the fact that the Algorithm 6 as well as the naïve algorithm for an $Agent_{\neg\mathbb{E}}^{dec,sem}$ was for smaller AF faster on average than the same algorithm with $Agent_{\mathbb{E}}^{dec,sem}$. Since the algorithm was not changed between the agent types, it is reasonable to assume that the verification runs faster with few arguments in one instance. Additionally, for both types of agent per algorithm, it can be seen from the table that there were no timeouts for the instances between 5 and 25 arguments.

Let us now compare the developed algorithm with the naïve one for an $Agent_{\mathbb{E}}^{dec,sem}$. Based on the slightly better average running times for smaller instances (specifically the $5 \leq |n| < 10$ data bin with $|n| = |Arg|$), that as with the $Agent_{\neg\mathbb{E}}^{dec,sem}$ the naïve algorithm is faster or equally fast. Therefore, we hypothesise that the running time is less or at least equal for the naïve algorithm than for the developed algorithm. We again formulate this as $H_0 : \tilde{x}_D \geq \tilde{x}_N$ for the $Agent_{\mathbb{E}}^{dec,sem}$ with $\tilde{x}_D$ being the running time of Algorithm 6 and $\tilde{x}_N$ that for the naïve algorithm. We check this with a one-sided Wilcoxon signed-rank test.

For the hypothesis H_0 we get $p \approx 0.001$, which is below the threshold of 0.05 and $T^+ = 274862$ with an effect size of $r = 0.549$, which lets us accept the hypothesis.

In summary, we attribute this result to the fact that many verification questions still had to be asked. In 836 cases the number of questions was the same and in only 67 the difference of questions asked compared to the naïve approach was larger than 10% out of which even only 28 showed a difference larger than one-third. This shows that often enough the developed algorithm could not save significant questions to the agent due to the extension set.

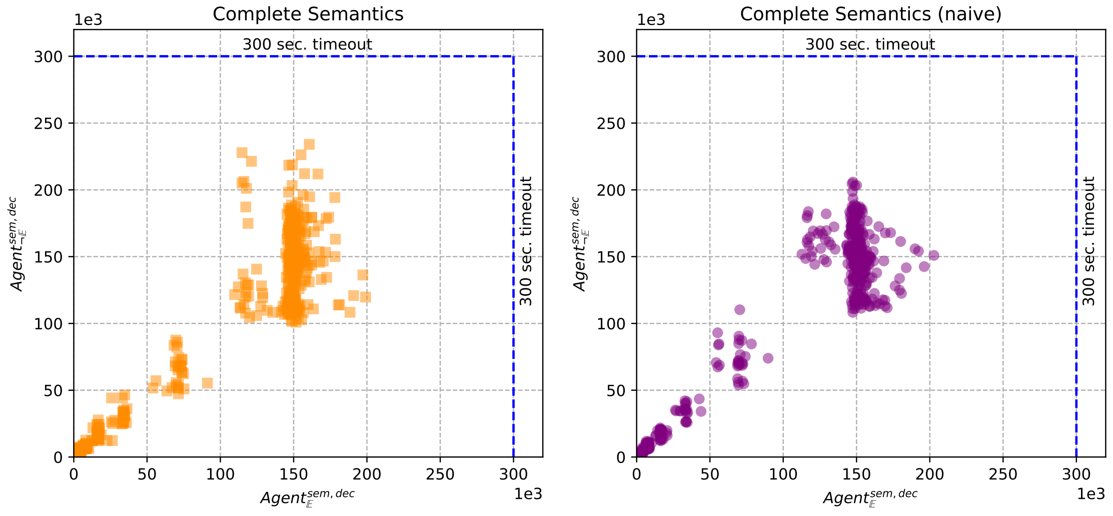

Figure 37: Plot of the runtimes in milliseconds of the two agent types. On the left side is the scatter plot of Algorithm 6 for the complete semantics and on the right side the naïve approach.

Algorithm 6 for $\sigma = co$	$\mathbf{5 \leq \|n\| < 10}$	$\mathbf{10 \leq \|n\| < 15}$	$\mathbf{15 \leq \|n\| < 20}$	$\mathbf{20 \leq \|n\| \leq 25}$
$\overline{time}$ for $Agent_{\mathbb{E}}^{dec,sem}$	$3.33 \cdot 10^9$	$2.73 \cdot 10^{10}$	$1.14 \cdot 10^9$	$1.10 \cdot 10^{11}$
$\overline{time}$ for $Agent_{\neg\mathbb{E}}^{dec,sem}$	$4.96 \cdot 10^7$	$1.73 \cdot 10^9$	$1.29 \cdot 10^9$	$1.04 \cdot 10^{11}$
$\sum$ T/O $Agent_{\mathbb{E}}^{dec,sem}$	0	0	0	0
$\sum$ T/O $Agent_{\neg\mathbb{E}}^{dec,sem}$	0	0	0	0
$\Delta_{timeouts}$	0	0	0	0

Naïve Algorithm for $\sigma = co$	$\mathbf{5 \leq \|n\| < 10}$	$\mathbf{10 \leq \|n\| < 15}$	$\mathbf{15 \leq \|n\| < 20}$	$\mathbf{20 \leq \|n\| \leq 25}$
$\overline{time}$ for $Agent_{\mathbb{E}}^{dec,sem}$	$2.87 \cdot 10^9$	$2.62 \cdot 10^{10}$	$1.11 \cdot 10^9$	$1.11 \cdot 10^{11}$
$\overline{time}$ for $Agent_{\neg\mathbb{E}}^{dec,sem}$	$4.65 \cdot 10^7$	$1.72 \cdot 10^8$	$1.29 \cdot 10^9$	$1.08 \cdot 10^{11}$
$\sum$ T/O $Agent_{\mathbb{E}}^{dec,sem}$	0	0	0	0
$\sum$ T/O $Agent_{\neg\mathbb{E}}^{dec,sem}$	0	0	0	0
$\Delta_{timeouts}$	0	0	0	0

Table 14: Comparison of runtimes in nanoseconds and timeouts between the two agent types for the complete semantics. The top table shows the results for Algorithm 6 and the bottom for the naïve algorithm.

5.3.4. Grounded Semantics

Next, we look at grounded semantics. For these, we had already found a very low runtime for an $Agent_{\neg\mathbb{E}}^{dec,sem}$ in the Algorithm 7. The two scatter plots in Figure 38 visually compare the agents for the developed algorithm (left) and the naïve algorithm (right). Here, it is noteworthy that the runtimes for both algorithms for $Agent_{\mathbb{E}}^{dec,sem}$ are significantly smaller.

The data from Table 15 also suggest such a state of affairs. The runtimes for algorithm 7 are an order of magnitude lower at 10^6 instead of 10^7. For the naïve algorithm, the difference is less clear from the runtime, but the number of timeouts is decisively larger for the data bin $n \leq |n| \leq 25$ with 1 for $Agent_{\mathbb{E}}^{dec,sem}$ versus 360 for $Agent_{\neg\mathbb{E}}^{dec,sem}$.

From the available data and the previous considerations, we hypothesise that also for an $Agent_{\mathbb{E}}^{dec,sem}$ the developed algorithm is faster. We therefore formulate $H_0 : \tilde{x}_N \leq \tilde{x}_D$ for the $Agent_{\mathbb{E}}^{dec,sem}$ with $\tilde{x}_D$ being the running time of Algorithm 7 and $\tilde{x}_N$ that for the naïve algorithm. We check this with a one-sided Wilcoxon signed-rank test. For this we obtain a p-value of $p \approx 1.663 \cdot 10^{-165}$ for the hypothesis and $T^+ = 500500$ such that $r = 1$ with which we can accept the hypothesis.

So again we see significant improvements in this algorithm for an $Agent_{\mathbb{E}}^{dec,sem}$, as testing the acceptance of an argument in a set relies on a simple search of the occurrence and $|gr(F)| = 1$ this can be done in constant time[17].

[17]See for example Java's HashSet implementation.

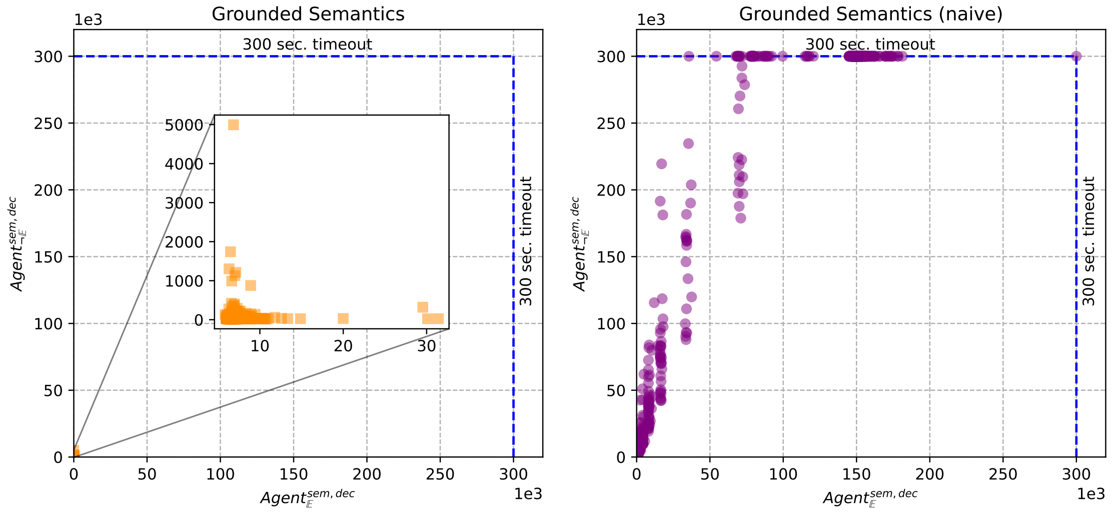

Figure 38: Plot of the runtimes in milliseconds of the two agent types. On the left side is the scatter plot of Algorithm 7 for the grounded semantics and on the right side the naïve approach. Note for the zoomed area in the left graph that the scale is not to be multiplied by 10^3.

Algorithm 7 for $\sigma = gr$	$\mathbf{5 \leq \lvert n \rvert < 10}$	$\mathbf{10 \leq \lvert n \rvert < 15}$	$\mathbf{15 \leq \lvert n \rvert < 20}$	$\mathbf{20 \leq \lvert n \rvert \leq 25}$
$\overline{time}$ for $Agent_{\mathbb{E}}^{dec,sem}$	$7.06 \cdot 10^6$	$7.30 \cdot 10^6$	$7.01 \cdot 10^6$	$7.20 \cdot 10^6$
$\overline{time}$ for $Agent_{\neg\mathbb{E}}^{dec,sem}$	$4.06 \cdot 10^7$	$4.43 \cdot 10^7$	$1.29 \cdot 10^7$	$5.85 \cdot 10^7$
$\sum$ T/O $Agent_{\mathbb{E}}^{dec,sem}$	0	0	0	0
$\sum$ T/O $Agent_{\neg\mathbb{E}}^{dec,sem}$	0	0	0	0
$\Delta_{timeouts}$	0	0	0	0

Naïve Algorithm for $\sigma = gr$	$\mathbf{5 \leq \lvert n \rvert < 10}$	$\mathbf{10 \leq \lvert n \rvert < 15}$	$\mathbf{15 \leq \lvert n \rvert < 20}$	$\mathbf{20 \leq \lvert n \rvert \leq 25}$
$\overline{time}$ for $Agent_{\mathbb{E}}^{dec,sem}$	$2.42 \cdot 10^8$	$1.36 \cdot 10^8$	$1.07 \cdot 10^9$	$1.07 \cdot 10^{11}$
$\overline{time}$ for $Agent_{\neg\mathbb{E}}^{dec,sem}$	$1.63 \cdot 10^8$	$3.64 \cdot 10^8$	$3.59 \cdot 10^9$	$2.30 \cdot 10^{11}$
$\sum$ T/O $Agent_{\mathbb{E}}^{dec,sem}$	0	0	0	1
$\sum$ T/O $Agent_{\neg\mathbb{E}}^{dec,sem}$	0	0	0	360
$\Delta_{timeouts}$	0	0	0	-359

Table 15: Comparison of runtimes in nanoseconds and timeouts between the two agent types for the grounded semantics. The top table shows the results for Algorithm 7 and the bottom for the naïve algorithm.

5.3.5. Preferred Semantics

As the second to last semantics, we examine the preferred semantics in terms of runtime for the agent types. Again, two scatter plots were created, which are shown in Figure 39. The left plot again shows the developed algorithm while the right plot shows the naïve approach. Especially for the left plot it can be seen that presumably there is a lot difference in the runtime, as there seem to be no timeouts for the $Agent_{\mathbb{E}}^{dec,sem}$. In the right plot we also see a strong improvement in favour of the agent that enumerated the extensions before. Here, too, there are no more evident timeouts and the runtime seems to be generally shorter.

Again, the collected data support this visually extracted speculation from the figure. In Table 16 it can be seen that the average runtimes are clearly lower for the developed Algorithm 8 with a big advantage for $Agent_{\mathbb{E}}^{dec,sem}$, where the timeouts are also no longer occur. This is similar for the naïve approach, where no more timeouts occurred for $Agent_{\mathbb{E}}^{dec,sem}$.

We deduce from this data that the two algorithms are faster for $Agent_{\mathbb{E}}^{dec,sem}$. Our hypothesis is therefore that the developed Algorithm is faster or equally fast and thus $H_0 : \tilde{x}_D \leq \tilde{x}_N$ for the $Agent_{\mathbb{E}}^{dec,sem}$ with $\tilde{x}_D$ the runtime of the Algorithm 8 and $\tilde{x}_N$ for the naïve one. For the one-sided test for hypothesis H_0, we obtain $p \approx 8.089 \cdot 10^{-63}$ with $T^+ = 97801$, therefore it can be accepted since $p < 0.05$. However, we point out here the comparatively low effect size of $r = 0.195$.

So for the preferred semantics, we find that the developed as well as naïve algorithms are significantly faster at $Agent_{\mathbb{E}}^{dec,sem}$.

Algorithm 8 for $\sigma = pr$	$\mathbf{5 \leq \lvert n \rvert < 10}$	$\mathbf{10 \leq \lvert n \rvert < 15}$	$\mathbf{15 \leq \lvert n \rvert < 20}$	$\mathbf{20 \leq \lvert n \rvert \leq 25}$
$\overline{time}$ for $Agent_{\mathbb{E}}^{dec,sem}$	$4.56 \cdot 10^7$	$9.37 \cdot 10^7$	$6.23 \cdot 10^8$	$6.29 \cdot 10^{10}$
$\overline{time}$ for $Agent_{\neg\mathbb{E}}^{dec,sem}$	$4.03 \cdot 10^8$	$1.08 \cdot 10^9$	$6.05 \cdot 10^{10}$	$2.38 \cdot 10^{11}$
$\sum$ T/O $Agent_{\mathbb{E}}^{dec,sem}$	0	0	0	0
$\sum$ T/O $Agent_{\neg\mathbb{E}}^{dec,sem}$	0	0	25	348
$\Delta_{timeouts}$	0	0	-25	-348

Naïve Algorithm for $\sigma = pr$	$\mathbf{5 \leq \lvert n \rvert < 10}$	$\mathbf{10 \leq \lvert n \rvert < 15}$	$\mathbf{15 \leq \lvert n \rvert < 20}$	$\mathbf{20 \leq \lvert n \rvert \leq 25}$
$\overline{time}$ for $Agent_{\mathbb{E}}^{dec,sem}$	$1.59 \cdot 10^8$	$1.45 \cdot 10^8$	$7.55 \cdot 10^8$	$7.48 \cdot 10^{10}$
$\overline{time}$ for $Agent_{\neg\mathbb{E}}^{dec,sem}$	$1.41 \cdot 10^8$	$6.05 \cdot 10^8$	$5.33 \cdot 10^{10}$	$2.23 \cdot 10^{11}$
$\sum$ T/O $Agent_{\mathbb{E}}^{dec,sem}$	0	0	0	0
$\sum$ T/O $Agent_{\neg\mathbb{E}}^{dec,sem}$	0	0	18	302
$\Delta_{timeouts}$	0	0	-18	-302

Table 16: Comparison of runtimes in nanoseconds and timeouts between the two agent types for the preferred semantics. The top table shows the results for Algorithm 8 and the bottom for the naïve algorithm.

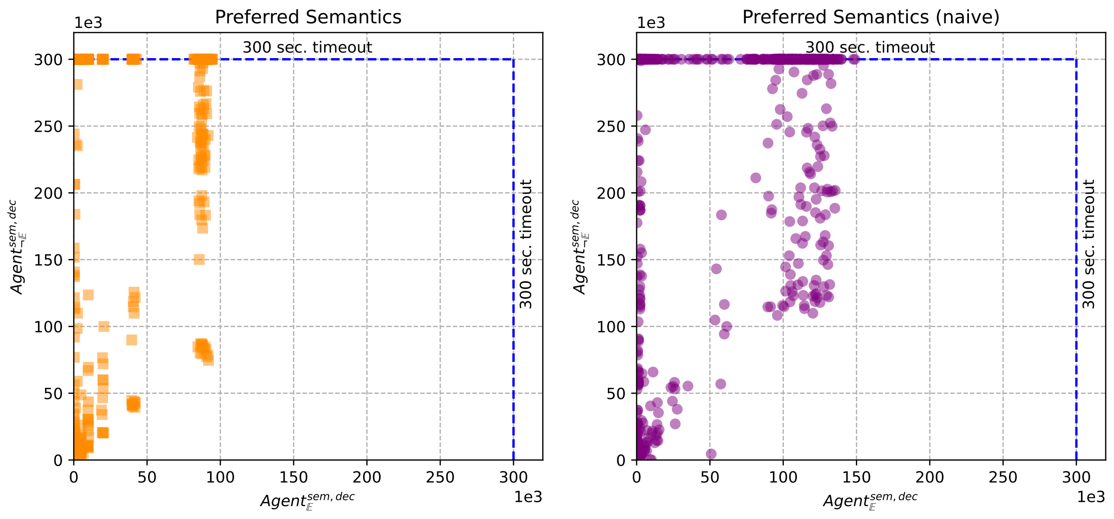

Figure 39: Plot of the runtimes in milliseconds of the two agent types. On the left side is the scatter plot of Algorithm 8 for the preferred semantics and on the right side the naïve approach.

5.3.6. Stable Semantics

We close the runtime analyses on the agents in experiment 2 with the stable semantics. First, we have a look at the Figure 40 with its two scatter plots. The left one represents the developed Algorithm 9. For this one, we see a slight improvement in runtimes among instances in favour of $Agent_{\mathbb{E}}^{dec,sem}$. The right scatter plot shows the naïve algorithm. For this one, the runtime seems to be the same, even with a slight advantage in favour of $Agent_{\neg\mathbb{E}}^{dec,sem}$.

This visual analysis corresponds to the data found in Table 17. For the developed algorithm, the runtime is marginally better in favour of $Agent_{\mathbb{E}}^{dec,sem}$. The situation is turned around for the naïve approach. While the two data bins with the smaller instances still show a small advantage for $Agent_{\mathbb{E}}^{dec,sem}$, the opposite is true for the two larger ones.

From the collected data, we therefore assume that in the case of the stable semantics, the naïve approach is marginally faster and otherwise equally fast. We test this again with a one-sided Wilcoxon signed-rank test with the hypothesis $H_0 : \tilde{x}_D \geq \tilde{x}_N$ with $\tilde{x}_D$ for the CPU runtime of the developed algorithm and $\tilde{x}_N$ for that of the naïve algorithm. We compute $p \approx 6.643 \cdot 10^{-114}$ and $T^+ = 457190$ where $p < 0.05$ and we therefore accept the hypothesis with an effect size of $r = 0.913$.

Regarding the stable semantics, we note that for the instances in the dataset pbbg-train, the naïve approach is again slightly faster. This is mainly due to the fact that the incomparability and tightness conditions rarely work sufficiently, as too few extensions are found. Only in 5 instances were the questions asked one third fewer compared to 903 instances where the number of questions asked was the same. This, together with the computation time for the conditions, makes the developed algorithm slightly inferior in the case of the present instances.

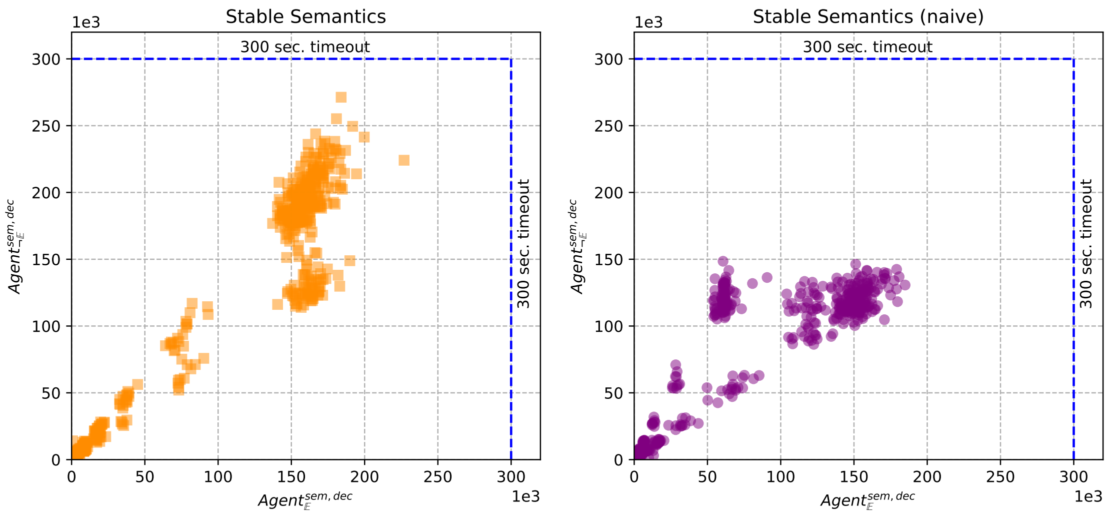

Figure 40: Plot of the runtimes in milliseconds of the two agent types. On the left side is the scatter plot of Algorithm 9 for the stable semantics and on the right side the naïve approach.

Algorithm 9 for $\sigma = st$	$\mathbf{5 \leq \lvert n \rvert < 10}$	$\mathbf{10 \leq \lvert n \rvert < 15}$	$\mathbf{15 \leq \lvert n \rvert < 20}$	$\mathbf{20 \leq \lvert n \rvert \leq 25}$
$\overline{time}$ for $Agent_{\mathbb{E}}^{dec,sem}$	$1.51 \cdot 10^8$	$2.20 \cdot 10^8$	$1.27 \cdot 10^9$	$1.15 \cdot 10^{11}$
$\overline{time}$ for $Agent_{\neg\mathbb{E}}^{dec,sem}$	$2.67 \cdot 10^8$	$4.33 \cdot 10^8$	$1.72 \cdot 10^9$	$1.32 \cdot 10^{11}$
$\sum$ T/O $Agent_{\mathbb{E}}^{dec,sem}$	0	0	0	0
$\sum$ T/O $Agent_{\neg\mathbb{E}}^{dec,sem}$	0	0	0	0
$\Delta_{timeouts}$	0	0	0	0

Naïve Algorithm for $\sigma = st$	$\mathbf{5 \leq \lvert n \rvert < 10}$	$\mathbf{10 \leq \lvert n \rvert < 15}$	$\mathbf{15 \leq \lvert n \rvert < 20}$	$\mathbf{20 \leq \lvert n \rvert \leq 25}$
$\overline{time}$ for $Agent_{\mathbb{E}}^{dec,sem}$	$9.55 \cdot 10^7$	$1.42 \cdot 10^8$	$1.03 \cdot 10^9$	$8.88 \cdot 10^{10}$
$\overline{time}$ for $Agent_{\neg\mathbb{E}}^{dec,sem}$	$1.59 \cdot 10^8$	$2.26 \cdot 10^8$	$9.51 \cdot 10^8$	$8.49 \cdot 10^{10}$
$\sum$ T/O $Agent_{\mathbb{E}}^{dec,sem}$	0	0	0	0
$\sum$ T/O $Agent_{\neg\mathbb{E}}^{dec,sem}$	0	0	0	0
$\Delta_{timeouts}$	0	0	0	0

Table 17: Comparison of runtimes in nanoseconds and timeouts between the two agent types for the stable semantics. The top table shows the results for Algorithm 9 and the bottom for the naïve algorithm.

5.3.7. Summary and Answering the Questions

We conclude this subsection by answering the two questions posed. We start with the first question: *Which of the algorithms are on average faster given an $agent_{\mathbb{E}}^{dec,sem}$?*

For the semantics $\sigma \subseteq \{cf, ad\}$ we recorded a degradation for the naïve algorithm and in the case of $\sigma = ad$ even for the developed algorithm. In the case of $\sigma = co$, there was no clear improvement for the algorithms but even a slight deterioration for instances with few arguments. In the case of $\sigma = gr$, there is a slight improvement for the developed algorithm and a clear improvement for the naïve algorithm by eliminating timeouts. Very significant improvements occurred for $\sigma = pr$ in both algorithms. We attribute this to the fact that Ver_{pr} is in the complexity class Π_2^P for an $Agent_{\neg\mathbb{E}}^{dec,sem}$ and a pure search in a precomputed extension set has a significantly lower computational complexity. Finally, for $\sigma = st$ a slight runtime reduction can be seen for the developed algorithm, while for the naïve algorithm there was no improvement but rather a minimal deterioration.

The second question to be answered was: *In this scenario, are the developed algorithms faster than the naïve algorithm?*

From the statistical tests, it can be seen that only for $\{gr, pr\} \subseteq \sigma$ was there any improvement, although it is worth noting that for the complete and stable semantics, the developed algorithms are only slightly slower, as can be seen from the data.

5.4. Experiment on Modifications of the Algorithms

We conclude our experiments with a third in which we modified the developed algorithms once again to take advantage of all possible questions now available through an $Agent_{\mathbb{E}}^{dec,sem}$ due to the runtime properties of searches in sets. These feature the inclusion of $ActualCount_{\sigma}^{n}$ as well as $Exists_{\sigma}^{\neg\emptyset}$ for all semantics, $Cred_{\sigma}$ for $\{ad, co, st, pr\} \subseteq \sigma$ and $Exists_{st}$ for the stable semantics. The modified algorithms, except for the grounded semantics, since we did not make any adjustments to it, can be found with an explanation in Appendix A.

In this experiment, we briefly compare the modified algorithms with the previously developed ones using the dataset pbbg-train. Secondly, using runtime improvements we investigate with the ICCMA'19 dataset up to which number of arguments in an instance the modified algorithms work for the semantics $\{co, gr, st, pr\} \subseteq \sigma$. We left $\{cf, ad\} \subseteq \sigma$ out because the modified algorithms are also not fast enough for large instances.

The questions to be answered are:

- *Are the modified algorithms faster on average?*
  For this we again form four data bins $5 \leq |Arg| < 10$, $10 \leq |Arg| < 15$, $15 \leq |Arg| < 20$ and $20 \leq |Arg| \leq 25$. We compare the algorithms using the dataset pbbg-train based on the arithmetic mean of the runtimes for the data

bins. If the average runtimes and also the number of timeouts are lower, we accept the assumption even without a statistical test.

- *At what number of arguments in an instance are the modified algorithms no longer practical?*
The ICCMA'19 instances contain AFs with up to several hundred arguments. We examine which instances did not time out and check how many arguments they have. We present the results.

We will only briefly describe the results of this experiment, as it only re-examines improvements in runtime and includes an additional dataset. Therefore, we start directly with answering the first question regarding the improvement of the average runtime. The Figure 41 shows scatter plots comparing the originally developed algorithms and the modified ones for all considered semantics. Only for the grounded semantics, no improvements through a modification were possible.

Benchmark Results As for the results, it is notable that there was a considerable improvement for the admissible sets. Conversely, there is no improvement, but rather a minimal worsening, in the case of conflict-free sets. In contrast to the four other semantics, however, there are still timeouts to be observed for both conflict-free and admissible sets. For all other semantics apart from the grounded semantics, the improvement is once again clear. This can explicitly be attributed to the use of the further computational problems respectively questions, which only became possible by using the $Agent_{\mathbb{E}}^{dec,sem}$. The restriction of the possible arguments in extensions by $Cred_\sigma$ and the search for the extension set cardinality by $ActualCount_\sigma^n$ limit the search space decisively. Also, with the acquired cardinality, a good stopping condition is available. The average runtimes per data bin and semantics are listed in Table 19.

Limit of Practicability Let us now turn to the second question. From the available data, it is evident that with the modified algorithms, it is not the number of arguments in an AF that is decisive, but above all the cardinality of the extension set and, furthermore, the set of credulous accepted arguments. Thus, even instances with several hundreds of arguments could be solved in cases where there were few extensions and accepted arguments. The results are shown in Table 18 for the largest solvable instances and Table 20 for the smallest unsolvable instances. From these tables it can be seen which instances were solved with how many arguments and extensions.

Summary We therefore conclude the third experiment with the findings that the modified algorithms are particularly suitable for $\sigma \subseteq \{co, pr, st\}$ are tractable for larger instances with respect to the number of arguments as long as the number of extensions is small, that is between 3 and 4 extensions. Only by using all the question categories found are runtimes achievable that also make larger instances solvable. For grounded semantics, on the other hand, it can be noted that it can also process very large instances although no improvements through a modification were possible. The good performance is because the algorithm is primarily based on the question $Cred_{gr}$ and only grows with the number of arguments of an instance.

σ	**Instance**	$\|$**Arg**$\|$	$\|\mathbb{S}\|$	$\|\|\mathbb{S}\|\|$	**T/O**
co	C-3-afinput_exp_acyclic_depvary_step7_batch_yyy08	1009	1	19	no
	T-4-afinput_exp_acyclic_depvary_step7_batch_yyy08	1009	1	19	no
	A-4-afinput_exp_acyclic_indvary1_step2_batch_yyy07	1028	1	9	no
	C-2-afinput_exp_acyclic_indvary1_step2_batch_yyy07	1028	1	9	no
	T-4-afinput_exp_acyclic_indvary1_step2_batch_yyy07	1028	1	9	no
gr	A-2-admbuster_10000	10000	1	5000	no
	C-1-admbuster_10000	10000	1	5000	no
	T-2-admbuster_10000	10000	1	5000	no
	A-3-grd_8034_1_2	8034	1	442	no
	C-1-grd_8034_1_2	8034	1	442	no
pr	C-3-afinput_exp_acyclic_depvary_step7_batch_yyy08	1009	1	19	no
	T-4-afinput_exp_acyclic_depvary_step7_batch_yyy08	1009	1	19	no
	A-4-afinput_exp_acyclic_indvary1_step2_batch_yyy07	1028	1	9	no
	C-2-afinput_exp_acyclic_indvary1_step2_batch_yyy07	1028	1	9	no
	T-4-afinput_exp_acyclic_indvary1_step2_batch_yyy07	1028	1	9	no
st	C-3-afinput_exp_acyclic_depvary_step7_batch_yyy08	1009	1	19	no
	T-4-afinput_exp_acyclic_depvary_step7_batch_yyy08	1009	1	19	no
	A-4-afinput_exp_acyclic_indvary1_step2_batch_yyy07	1028	1	9	no
	C-2-afinput_exp_acyclic_indvary1_step2_batch_yyy07	1028	1	9	no
	T-4-afinput_exp_acyclic_indvary1_step2_batch_yyy07	1028	1	9	no

Table 18: The largest instances regarding the number of arguments that could be solved by the modified algorithms for the complete, grounded, preferred and stable semantics.

	$\overline{time}$ (arithmetic means in nanoseconds)				
	$5 \leq \|Arg\| < 10$	$10 \leq \|Arg\| < 15$	$15 \leq \|Arg\| < 20$	$20 \leq \|Arg\| \leq 25$	$\sum$ timeouts
cf	$3.66 \cdot 10^7$	$5.11 \cdot 10^8$	$4.57 \cdot 10^{10}$	$2.52 \cdot 10^{11}$	417
ad	$1.52 \cdot 10^7$	$9.99 \cdot 10^7$	$2.87 \cdot 10^9$	$3.30 \cdot 10^{10}$	43
co	$1.27 \cdot 10^7$	$2.01 \cdot 10^7$	$7.26 \cdot 10^8$	$2.78 \cdot 10^8$	0
gr	$3.76 \cdot 10^6$	$3.17 \cdot 10^6$	$3.28 \cdot 10^6$	$3.62 \cdot 10^6$	0
pr	$1.38 \cdot 10^7$	$2.21 \cdot 10^7$	$7.89 \cdot 10^7$	$3.90 \cdot 10^8$	0
st	$1.39 \cdot 10^7$	$2.58 \cdot 10^7$	$1.10 \cdot 10^8$	$5.36 \cdot 10^8$	0

Table 19: Average runtimes for four data bins and the sum of the timeouts for the modified algorithms per semantics.

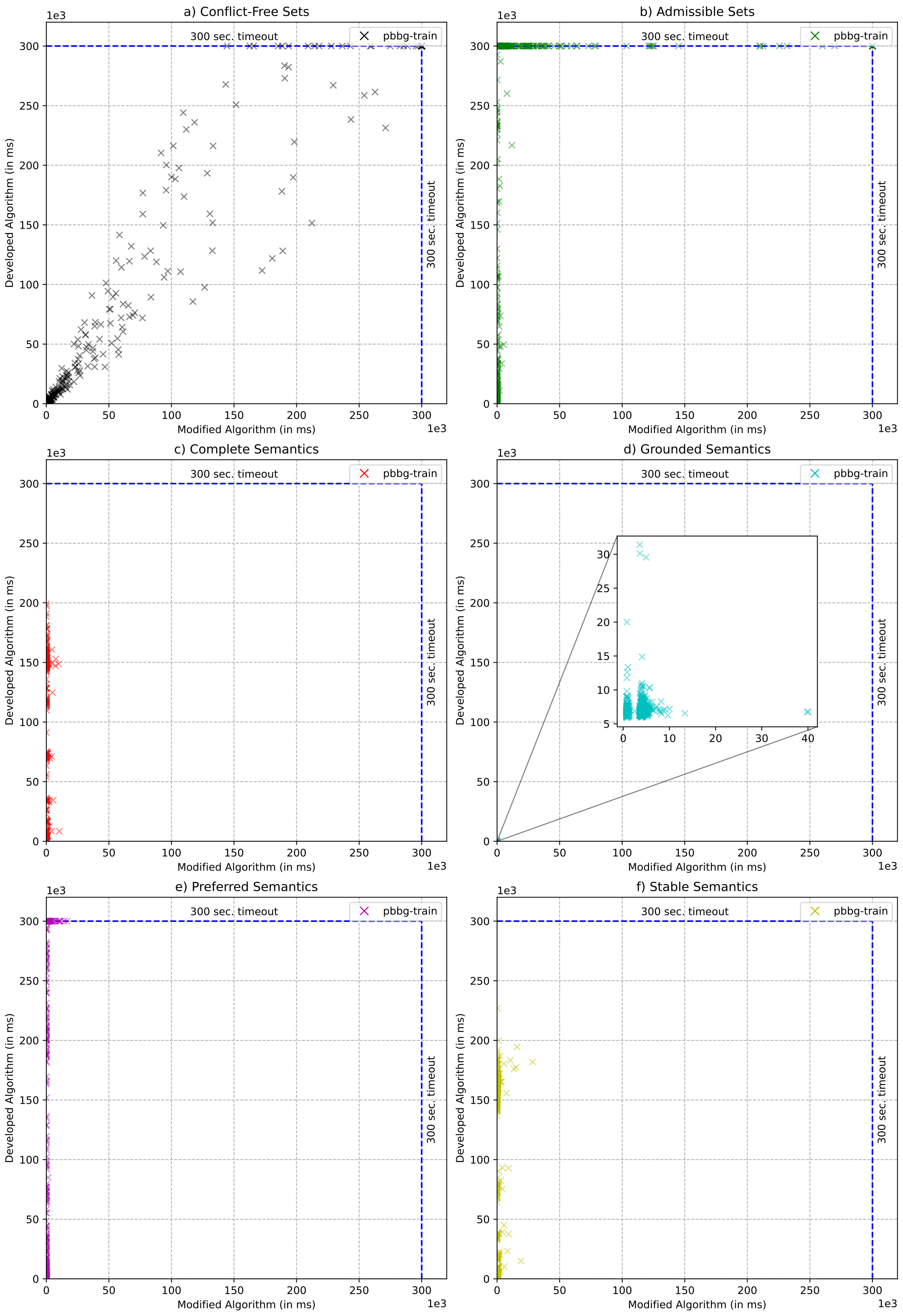

Figure 41: Plot of the runtimes in milliseconds comparing the modified algorithms and the originally developed ones for an $Agent_{\mathbb{E}}^{dec,sem}$.

σ	**Instance**	$\|$**Arg**$\|$	$\|\mathbb{S}\|$	$\|\|\mathbb{S}\|\|$	**T/O**
co	Small-result-b44	27	4	27	yes
	Small-result-b12	30	3	30	yes
	Small-result-b15	30	3	30	yes
	Small-result-b18	30	3	30	yes
	C-1-calaveras-transit_20151216_1707.gml.50	32	13404	28	yes
gr	-	-	-	-	-
pr	Small-result-b44	27	3	27	yes
	Small-result-b12	30	2	30	yes
	Small-result-b15	30	2	30	yes
	Small-result-b18	30	2	30	yes
	C-1-calaveras-transit_20151216_1707.gml.50	32	375	28	yes
st	Small-result-b44	27	3	27	yes
	Small-result-b12	30	2	30	yes
	Small-result-b15	30	2	30	yes
	Small-result-b18	30	2	30	yes
	Small-result-b60	32	4	32	yes

Table 20: The small instances regarding the number of arguments that could not be solved for the complete, grounded, preferred and stable semantics.

5.5. Discussion of Results

The results of the experiments will now be discussed briefly. To do this, we will look at all three experiments individually, seek explanations and possibilities for improvement and draw an overall conclusion at the end.

The First Experiment The analyses of these experiments showed unexpected results. Only in two cases were the developed algorithms faster for the two examined datasets pbbg-train and pbbg-test. Given that the test conditions were introduced to reflect the properties of the signatures, it was to be expected that the algorithms would perform better. In fact, however, this was only the case for the complete semantics and the grounded semantics. Although it is no surprise for the second semantics, as it does not implement an exhaustive search.

In general, the runtimes were high and the number of timeouts for some semantics like $\{cf, ad, pr\} \subseteq \sigma$ were considerably large. This leads to the argument that an algorithm that works according to the brute force approach is hardly sustainable. This is not surprising due to the exponential growth, which became visible when examining the non-linear correlation, among other things.

The Second Experiment This experiment showed that simply changing the agent type does not necessarily lead to better runtimes. Only for the grounded semantics could significant runtime improvements be achieved. If one directly compares the developed algorithms with the naïve algorithm per semantics, it is noticeable that the naïve algorithm became equally better and was even able to undercut the complete semantics again. The algorithms were even slightly slower for $\{cf, ad\} \subseteq \sigma$.

Thus, this experiment showed that under certain circumstances the Ver_σ-question can be computed quickly and that a search in a previously computed extension set did not bring such a large improvement in some developed algorithms in principle that the runtimes for the test conditions could be compensated. However, this is also due to the fact that properties such as tightness or conflict-sensitivity could often not be used profitably because the extension sets were designed in such a way that an exhaustive search meant that first extensions were only found late and thus the conditions could hardly be used.

The Third Experiment An $Agent_{\mathbb{E}}^{dec,sem}$ can answer questions that were previously not tractable by having the extensions enumerated in advance. We modified the algorithms so that these questions were included. These were among others $Unique_\sigma$, $Cred_\sigma$ or $ActualCount_\sigma^n$. In this mixture, a clear acceleration was recorded for $\{co, pr, st\} \subseteq \sigma$. For the grounded semantics, no modification was possible, but also not necessary, as the performance was already good compared to the other semantics.

The experiment showed the influence of the arguments and extensions on the runtime. For this purpose, the datasets pbbg-train were used for comparison with the results from experiment 2 and the ICCMA'19 dataset for instances with $|Arg| > 25$. What was observed in the analysis is that the number of arguments in an instance can be high, but the number of extensions ($|\mathbb{S}|$) and the number of arguments in it ($||\mathbb{S}||$) are crucial. The present results show that by modifying the algorithms by the uniqueness question, i.e. $|\mathbb{S}| = 1$, the performance for eliciting unique extensions improves to the level of the algorithm for grounded semantics, since only those arguments have to be found that are in the unique extension by means of $Cred_\sigma$.

For $|\mathbb{S}| > 1$ we recorded that with more than four extensions the number of timeouts of 5 minutes increased strongly, especially if there were more than 28 to 30 arguments in the extensions, depending on the semantics. We can use the exhaustive search as an explanation, since the modified algorithms only reduced the search space and were able to determine an actual upper limit, but in the inner loop body the power set of all accepted arguments continued to be traversed, which allows results from experiment 2 to be carried over. The instances there had $|Arg| \leq 25$ and the algorithm performed an exhaustive search over $|Arg|$. Since the search space of such an exhaustive search grows exponentially with the number of accepted arguments and runtimes well over one minute on average were already observed in experiment 2, the results from experiment 3 are not unexpected.

Summary All in all, the overall conclusion is that for an $Agent_{\neg\mathbb{E}}^{dec,sem}$, elicitation is generally impractical apart from, on the one hand, the grounded semantics using the developed algorithm and, on the other hand, conflict-free and admissible sets using the naïve algorithm. This is partly due to the lack of a good stopping condition and an exponential runtime of $O(n^{|Arg|})$ with $|Arg|$ as the number of arguments in the elicitation. That being said, the naïve algorithm is faster for this class of agent in most cases. This is partly due to the fact that the checks within the algorithms have, among other things, linear runtimes. Since more than one of these checks is usually used, the realistic runtime of the developed algorithms is worse than that of the naïve algorithm. These checks do not outweigh the reduction of questions to the agent in terms of runtime. Much more, at the beginning of the exhaustive search, little information about the final extension set is known, so that, for example, in the stable semantics algorithm (see Algorithm 9) $Pairs_{\mathbb{S}}$ cannot be sufficiently filled yet and so many verification questions are asked of presumed extensions that would never have been submitted to the agent for verification with the knowledge of the entire extension set $\mathbb{S}$.

For $Agent_{\mathbb{E}}^{dec,sem}$, on the other hand, the picture looks somewhat different. Although here too the unmodified developed algorithms are slower compared to the naïve approach for $\{cf, ad, co, st\}$, the use of the modified algorithms is faster especially as the number of arguments increases. This is partly because $Cred_{\sigma}$ can be used to greatly restrict the search space. A use of $ActualCount_{\sigma}^{n}$ was also aimed at, such that the actual number of extensions could be retrieved with it, which must be lower or equal to the diversity function. This provided a good stopping condition and decreased the runtime.

Furthermore, for elicitation with an $Agent_{\mathbb{E}}^{dec,sem}$, it can be seen that it is much more feasible from a runtime point of view if the algorithms are extended to include the additional question options. This is because many questions are reduced in complexity to a search in a set. In this case, however, the total runtime consists not only of the elicitation by the questions but also of the enumeration of the extensions by the agent beforehand, so that the runtime consists of an enumeration part T_1 and the interview T_2, so that $T = T_1 + T_2$ is the total runtime T. It should also be mentioned that all questions in the end only restrict the search space and, except in the case of a unique extension, an exhaustive search must still be performed over all accepted arguments. This means that the runtime T_2 grows exponentially in the worst case for this implementation.

In conclusion, with the restrictions with the selected questions under both agent types, not enough information could be gathered to find algorithms that did not have an exponential runtime in the worst case. But even for the average case, it showed that the lack of a good stopping condition (except in the modified algorithms) also kept the runtimes high. The theoretical upper bounds used in the developed algorithms did not work for those instances[18]. However, since conditions exist to include or exclude certain extensions after collecting enough information, it makes sense for a future investigation to look again at optimisation potentials regarding the search space.

[18]Deeper analysis showed that they were significantly higher than the actual extension set sizes.

6. Summary and Outlook

In this section, we first summarise the most important findings of this Master's thesis. Then we show what further research can be conducted in the field of elicitation based on the considerations from this thesis.

6.1. Conclusion

The research topic of elicitation pursues the goal of reconstructing an argumentation framework from partially revealed information and an interactive question-answer setting. For this, the reconstructed argumentation framework must meet certain equivalence criteria. In this Master's thesis, we therefore developed and analysed algorithms for questioning an agent to elicit a σ-equivalent argumentation framework, taking into account classical semantics, with the restriction that the agent honestly answers semantic decision questions.

For this purpose, we first developed a question pool consisting of typical computational problems for argumentation frameworks. These can be solved by an agent just with the knowledge about the arguments and their attack relation and, most notably, without knowing about the extension set under that semantics. With these questions and known properties about the extension sets of the semantics, algorithms were developed. Only questions that are also considered tractable in terms of their computational complexity were included. The performance of the algorithms was then evaluated in an experiment. We also extended these to agents that already know the extension set and finally modified the algorithms to include questions that are tractable with such an agent type. Furthermore, we discussed the possibilities of reconstructing a σ-equivalent argumentation framework from the interview information. For this purpose, we focused on the reconstruction of an argumentation framework that contains only those arguments that are also present in the agent's argumentation framework.

Based on the results of the experiments from Section 5, it becomes apparent that the developed algorithms are not suitable for large argumentation frameworks. This is due to, among other things, only a very weak stop condition and the fact that the algorithms are based on an exhaustive search. Also, the performance of the self-implemented question solver is not optimal. Other implementation strategies, such as MaxSAT instead of procedural Java implementations, could make the modified algorithms faster than the basic versions developed in Subsection 4.3. This assumption is supported by brief supplementary experiments in Appendix B. The findings there suggest that if computationally more complex questions can be answered relatively quickly, then this can positively affect the runtime, as the search space can be reduced in at least some cases. This is in line with the results presented here in the main body. Our own implementation is not able to use the modified algorithms in a meaningful way for $Agent^{dec,sem}_{\neg\mathbb{E}}$ and is only to some degree tractable for the modified algorithms with $Agent_{\mathbb{E}}^{dec,sem}$, while some other solvers could at least in some cases be able to answer these more complex questions faster for $Agent_{\neg\mathbb{E}}^{dec,sem}$ and therefore drop out later under this agent. Nevertheless, the choice of a solver only shifts the point at which elicitation becomes impractical.

As for a subject of further investigations, we identified, among other things, that the opening or modification of the restrictions is of interest, for example, allowing syntactic questions. With regard to the reconstruction part of the elicitation, this is of interest, since more information about conflicts can be collected. This means, that argument-congruent reconstructions can become possible in cases that were not possible in the scenario of this Master's thesis. We also suspect that this will make it easier to formulate stop conditions and sort out sets, thereby improving runtimes and hence tractability.

6.2. Future Work

As the very last element of this Master's thesis, we give a rather brief insight into further research topics. This includes the expansion of the created question pool, the use of additional or even multiple semantics, the possibilities to translate between them, and what can be explored to improve argument-congruent reconstruction.

6.2.1. Expansion of the Permitted Questions

Within the scope of the research question, we have made some restrictions, among other things, on the allowed questions that can be asked of an agent or, conversely, that the agent can answer. In Subsection 4.2 we eventually selected the concrete questions and limited ourselves to semantic decision questions. These have a direct relation to the extensions.

Therefore, further research concerning the questions should be conducted. On the one hand, from the developed algorithms and the work on properties of signatures, we see a potential to formulate these properties as questions. Thus, decision questions such as "Exists $(a, b) \in Pairs_{\mathbb{S}}$?" or "Is $(\mathbb{T} \subseteq \mathbb{S}) \cup S$ conflict-sensitive?" may be asked and answered by the agent. Studies to this extent are therefore of interest.

Further opportunities arise from the enlargement of the decision question pool by syntactic questions, on the one hand with reference to the extensions like, "Does extension E attack an argument a?" or, on the other hand, questions without this reference, like, for example, "Does a attack the argument b, i.e. $a \hookrightarrow b$?". Here, in addition to the improvement of the interview algorithms, we also assume one for the argument-congruent reconstruction, since information on the internal structure is revealed.

Finally, it should also be examined whether queries can be introduced. For this purpose, scenarios are to be developed under which circumstances such questions are to be asked in a meaningful way. In general, but also in the question of queries, the investigation of the applicability to machine learning can be examined. In this

Master's thesis, it was assumed that the agent possesses and hides an argumentation framework. An agent that accepts arguments based on methods from the field of machine learning is therefore of interest as a research object, since it is conceivable that the elicitation approach can be applied there to find an explanatory AF for the model.

6.2.2. Examine Further and Allow Multiple Semantics

Next, we take a look at possible further research in terms of semantics in elicitation. First, with regard to extension-based semantics, there are considerably more than the classical semantics, such as the eager semantics [13] or ideal semantics [21], etc., to name only a few. For these, algorithms for elicitation can be developed as the next step.

In addition, the use of multiple semantics is of interest. This requires the agent to be able to answer questions about more than one semantics.

In this scenario, insights from Kuhlmann et al. [35] regarding the distinguishability of semantics can be used here. A semantics σ distinguishes a second semantics σ' if it holds that argumentation frameworks that have the same extensions under σ have the same extensions for σ' as well.

The results of their work show that such distinguishability is quasi only given under specific graph classes. Under the restrictions of the scenario of this Master's thesis, they can therefore not be utilised. However, if the pool of questions to the agent is expanded, as previously suggested in Subsection 6.2.1, so that syntactic questions may be asked that allow the detection of a graph class, then the results may be of interest and should be examined accordingly.

In addition, one semantics could be used to retrieve certain information that is useful for analysing the other semantics. For example, $Cred_{cf}$ could be used to find out which arguments are accepted in conflict-free sets. From this, it can be deduced which ones attack themselves and these then need not be considered further in e.g. $\sigma = co$. This reduces the search space. This kind of information extraction using multiple semantics offers a further research option as well.

Finally, instead of extension-based semantics, it makes sense to explore semantics based on labels. Labels bring additional status information about arguments, as shown in Subsection 2.3, so that not only could the question-answer process change, but also there might be an influence on the argument-congruent reconstruction abilities.

6.2.3. Investigating Applicability of Intertranslation

The experiments from Section 5 show that the runtimes of the algorithms can differ substantially. Therefore, it may be of interest to transfer one semantics into another in order to have a better overall performance under the same output.

The idea also corresponds to that of translation, among others. Dvořák and Woltran first introduced the topic in [29] as an adoption of the idea of reductions from computational complexity into abstract argumentation.

Informally speaking, a argumentation framework is transformed in such a way that the original semantics is implemented by a target semantics, i.e. a function $Tr : \sigma \to \sigma'$ is given that transforms σ into another σ'. Spanring [45] as well as later Dvořák and Spanring [28] gave a lot of insights into possible translations. A translation is exact if $\sigma(F) = \sigma'(Tr(F))$ is given. Also, there are less strict variants as a faithful translation where $|\sigma(F)| = |\sigma'(Tr(F))|$ and $\sigma(F) = \{E \cap A \mid E \in \sigma'(Tr(F))\}$ prove to be more practicable.

However, it turns out that, as Spanring's Master's thesis [45] shows, no practical transformation of a computationally more complex semantics into a simpler one is possible for the considered computational problems respectively decision questions and classical semantics shown. Yet, with the background of the experiments on the developed algorithms, a few translations such as ad to co can be found that might be interesting.

The choice from possible translations and the examination of their application in the case of elicitation can therefore be a further subject of research.

6.2.4. Identifying Necessary Requirements on Elicitation for Argument-Congruent Reconstructions

During the discussion in Subsection 4.4 on reconstruction using only the known arguments Arg from the agent's argumentation framework, it was found, that due to the Explicit Conflict Conjecture, a reconstruction with the collected information for the complete, grounded and stable semantics as well as the admissible sets is not possible immediately.

Therefore, requirements need to be formulated that have to be fulfilled in the interview part of the elicitation in order to enable an immediate argument-congruent reconstruction. These requirements could be set for the questioning process and the information to be gathered. A starting point for this could be existing studies on the enforcing problem [6, 50] and the minimal change problem [5, 7]. These questions deal with how the introduction of new information creates an argumentation framework that has the desired extensions (or subsets thereof).

References

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009.

[2] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *The knowledge engineering review*, 26(4):365–410, 2011.

[3] Pietro Baroni, Paul E Dunne, and Massimiliano Giacomin. On extension counting problems in argumentation frameworks. In *Computational Models of Argument*, pages 63–74. IOS Press, 2010.

[4] Pietro Baroni and Massimiliano Giacomin. On principle-based evaluation of extension-based argumentation semantics. *Artificial Intelligence*, 171(10-15):675–700, 2007.

[5] Ringo Baumann. What does it take to enforce an argument? minimal change in abstract argumentation. In *ECAI*, volume 12, pages 127–132, 2012.

[6] Ringo Baumann and Gerhard Brewka. Expanding argumentation frameworks: Enforcing and monotonicity results. *COMMA*, 10:75–86, 2010.

[7] Ringo Baumann and Gerhard Brewka. Spectra in abstract argumentation: An analysis of minimal change. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 174–186. Springer, 2013.

[8] Ringo Baumann, Wolfgang Dvořák, Thomas Linsbichler, Christof Spanring, Hannes Strass, and Stefan Woltran. On rejected arguments and implicit conflicts: The hidden power of argumentation semantics. *Artificial Intelligence*, 241:244–284, 2016.

[9] Ringo Baumann and Hannes Strass. On the maximal and average numbers of stable extensions. In *International Workshop on Theorie and Applications of Formal Argumentation*, pages 111–126. Springer, 2013.

[10] Ringo Baumann and Hannes Strass. Open problems in abstract argumentation. In *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation*, pages 325–339. Springer, 2015.

[11] Stefano Bistarelli, Lars Kotthoff, Francesco Santini, and Carlo Taticchi. A first overview of iccma'19. In *AI^3@ AI* IA*, pages 90–102, 2020.

[12] Gerhard Brewka, Sylwia Polberg, and Stefan Woltran. Generalizations of dung frameworks and their role in formal argumentation. *IEEE Intelligent Systems*, 29(1):30–38, 2013.

[13] Martin Caminada. 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, 2007.

[14] Martin WA Caminada and Dov M Gabbay. A logical account of formal argumentation. *Studia Logica*, 93(2):109–145, 2009.

[15] Chalaguine, Lisa A. and Hunter, Anthony. A persuasive chatbot using a crowd-sourced argument graph and concerns. *Computational Models of Argument: Proceedings of COMMA 2020*, 326:9, 2020.

[16] Günther Charwat, Wolfgang Dvořák, Sarah A Gaggl, Johannes P Wallner, and Stefan Woltran. Methods for solving reasoning problems in abstract argumentation–a survey. *Artificial intelligence*, 220:28–63, 2015.

[17] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Symmetric argumentation frameworks. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 317–328. Springer, 2005.

[18] Dennis Craandijk and Floris Bex. Deep learning for abstract argumentation semantics. *Proceedings of the Twenty- Ninth International Joint Conference on Artificial Intelligence*, pages 1667–1673, 2020.

[19] Yannis Dimopoulos and Alberto Torres. Graph theoretical structures in logic programs and default theories. *Theoretical Computer Science*, 170(1-2):209–244, 1996.

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

[21] Phan Minh Dung, Paolo Mancarella, and Francesca Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 171(10-15):642–674, 2007.

[22] Paul E Dunne and Trevor JM Bench-Capon. Coherence in finite argument systems. *Artificial Intelligence*, 141(1-2):187–203, 2002.

[23] Paul E Dunne, Wolfgang Dvořák, Thomas Linsbichler, and Stefan Woltran. Characteristics of multiple viewpoints in abstract argumentation. *Artificial Intelligence*, 228:153–178, 2015.

[24] Paul E Dunne, Anthony Hunter, Peter McBurney, Simon Parsons, and Michael Wooldridge. Weighted argument systems: Basic definitions, algorithms, and complexity results. *Artificial Intelligence*, 175(2):457–486, 2011.

[25] Wolfgang Dvorák. On the complexity of the uniqueness problem in abstract argumentation. 2017.

[26] Wolfgang Dvorák and Paul E Dunne. Computational problems in formal argumentation and their complexity. *Handbook of formal argumentation*, 4, 2018.

[27] Wolfgang Dvořák, Sebastian Ordyniak, and Stefan Szeider. Augmenting tractable fragments of abstract argumentation. *Artificial Intelligence*, 186:157–173, 2012.

[28] Wolfgang Dvořák and Christof Spanring. Comparing the expressiveness of argumentation semantics. *Journal of Logic and Computation*, 27(5):1489–1521, 2017.

[29] Wolfgang Dvořák and Stefan Woltran. On the intertranslatability of argumentation semantics. *Journal of Artificial Intelligence Research*, 41:445–475, 2011.

[30] Arthur C Graesser and Natalie K Person. Question asking during tutoring. *American educational research journal*, 31(1):104–137, 1994.

[31] Hadassa Jakobovits and Dirk Vermeir. Robust semantics for argumentation frameworks. *Journal of logic and computation*, 9(2):215–261, 1999.

[32] Ummul Khair, Hasanul Fahmi, Sarudin Al Hakim, and Robbi Rahim. Forecasting error calculation with mean absolute deviation and mean absolute percentage error. In *Journal of Physics: Conference Series*, volume 930, page 012002. IOP Publishing, 2017.

[33] Hiroyuki Kido and Beishui Liao. A bayesian approach to direct and inverse abstract argumentation problems. *arXiv e-prints*, pages arXiv–1909, 2019.

[34] Isabelle Kuhlmann. Towards Eliciting Attacks in Abstract Argumentation Frameworks. *Online Handbook of Argumentation for AI*, 2, 2021.

[35] Isabelle Kuhlmann, Tjitze Rienstra, Lars Bengel, Kenneth Skiba, and Matthias Thimm. Distinguishability in Abstract Argumentation. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 18, pages 686–690, 2021.

[36] Wendy Grace Lehnert. *The process of question answering.* Yale University, 1977.

[37] Andreas Niskanen, Johannes Wallner, and Matti Järvisalo. Synthesizing argumentation frameworks from examples. *Journal of Artificial Intelligence Research*, 66:503–554, 2019.

[38] Emilia Oikarinen and Stefan Woltran. Characterizing strong equivalence for argumentation frameworks. *Artificial intelligence*, 175(14-15):1985–2009, 2011.

[39] C.H. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244–259, 1984.

[40] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[41] John L. Pollock. *Cognitive Carpentry: A Blueprint for How to Build a Person*. The MIT Press, May 1995.

[42] Antonio Rago, Oana Cocarascu, Christos Bechlivanidis, and Francesca Toni. Argumentation as a framework for interactive explanations for recommendations. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 17, pages 805–815, 2020.

[43] Régis Riveret and Guido Governatori. On learning attacks in probabilistic abstract argumentation. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 653–661, 2016.

[44] Murray Singer. Answering yes-no questions about causes: Question acts and question categories. *Memory & cognition*, 14(1):55–63, 1986.

[45] Christof Spanring. *Intertranslatability results for abstract argumentation semantics*. Master's Thesis, 2012.

[46] Matthias Thimm. Tweety - A Comprehensive Collection of Java Libraries for Logical Aspects of Artificial Intelligence and Knowledge Representation. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14)*, July 2014.

[47] Matthias Thimm. The Tweety Library Collection for Logical Aspects of Artificial Intelligence and Knowledge Representation. *Künstliche Intelligenz*, 31(1):93–97, March 2017.

[48] Markus Ulbricht. On the maximal number of complete extensions in abstract argumentation frameworks. In *KR*, pages 707–711, 2021.

[49] Leon van der Torre and Srdjan Vesic. The principle-based approach to abstract argumentation semantics. *IfCoLog Journal of Logics and Their Applications*, 2017.

[50] Johannes P Wallner, Andreas Niskanen, and Matti Järvisalo. Complexity results and algorithms for extension enforcement in abstract argumentation. *Journal of Artificial Intelligence Research*, 60:1–40, 2017.

[51] Elise Whitley and Jonathan Ball. Statistics review 6: Nonparametric methods. *Critical care*, 6(6):1–5, 2002.

A. Modified Algorithms

What follows is a description of the modified algorithms for an $Agent_{\mathbb{E}}^{dec,sem}$. Depending on the algorithm, the adaptations are slightly different, so we will present them one by one. In fact, the core of the extension search algorithm has remained the same, so we do not describe it in detail. However, some modifications have happened around this iterative search to narrow the search space. We now list these modifications and name the semantics to which it applies:

- *Checking for the presence of an extension*: We introduced this check using $Exists_\sigma$ only for the stable semantics. It is the only one among the classical semantics for which there is no guarantee that an extension exists at all.
- *Check for at least one non-empty extension*: This is realised via $Exists_\sigma^{\neg\emptyset}$ and was introduced for all semantics. In the case where only one empty extension exists, this makes the solution trivial and speeds up the algorithm.
- *Check for more than one extension*: This was implemented for all semantics except $\sigma = gr$, since the cardinality is always $|\sigma(F)| = 1$ for any AF F under $\sigma = gr$. For all others, this was taken as a speedup using $Unique_\sigma$. In the case of semantics where the empty set $\{\}$ always occurs as an extension, this is taken. Otherwise, a search is made for the (credulous) accepted arguments and their set is taken as extension.
- *Elicit the number of extensions*: The diversity function Δ_σ gives an upper limit. However, this does not give the actual amount of extensions. By means of $ActualCount_\sigma^n$, a fetching of the actual number of extensions is implemented in $\sigma \in \{cf, ad, co, pr, st\}$ iteratively up to the upper bound.
- *Search space reduction by accepted arguments*: If more than one extension is present, the agent is asked beforehand which arguments are accepted credulously. This can reduce the search space in favourable cases, as the power set is smaller. This is applied to all semantics apart from $\sigma = gr$.

In the following subsections, we briefly describe the modified algorithm for each semantics apart from the grounded semantics as we did no modification to it.

A.1. Conflict-Free Sets

The adjustments in Algorithm 13 for conflict-free sets are somewhat smaller than in the following algorithms, because $Cred_{cf}$ is already present as a question in the Algorithm 4. We therefore added a check as a bracket to the algorithm whether a non-empty extension exists ($Exists_{cf}^{\neg\emptyset}$).

Moreover, in the case that there is more than one empty extension, there follows another new loop after step I, counting up from 2 to Δ_{cf} and asking the agent whether the counter i corresponds to the cardinality of the extension set. Since the agent answers truthfully to $ActualCount^i_{cf}$ and Δ_{cf} by definition contains the theoretical upper bound, the loop is always terminated with $n = |\sigma(F_A)|$ with F_A the agent's AF. Subsequently, in the loop body for the questioning, $|\mathbb{S}| = n$ is checked. Since n contains the cardinality, this stop condition takes effect when all relevant extensions have been found.

No other changes were made.

Algorithm 13 Modified Algorithm for Conflict-Free Sets for the Interviewing Part

```
Require: $Arg$ as a set of arguments, $\sigma = cf$
Ensure: An extension set $\mathbb{S}$
  $Args_{\mathbb{S}} \leftarrow \emptyset$
  $\mathbb{S} \leftarrow \emptyset$
  if $Exists^{\neg\emptyset}_{cf}$ then
      for all $a \in Arg$ do                                    ▷ Step I
          if $Cred_{cf}(a) = \top$ then
              $Args_{\mathbb{S}} \leftarrow Args_{\mathbb{S}} \cup \{a\}$
      for $i = 2..\Delta_{cf}(|Args_{\mathbb{S}}|)$ do
          if $ActualCount^i_{cf} = \top$ then
              $n \leftarrow i$
              break
      $\mathbb{S} \leftarrow \{\{\}\}$
      $\mathcal{S} \leftarrow 2^{Args_{\mathbb{S}}} \setminus \mathbb{S}$
      for all $s \in \mathcal{S}$ do                            ▷ Step II
          if $|\mathbb{S}| = n$ then                            ▷ Stop Condition
              break
          if $\exists s' \in \mathbb{S} : s \subseteq s'$ then
              $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$
          else
              $r \leftarrow$ Ask agent $Ver_{cf}(s)$
              if $r = \top$ then
                  $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$
  else
      $\mathbb{S} \leftarrow \{\{\}\}$
```

A.2. Admissible Sets

Algorithm 14 Modified Algorithm for Admissible Sets for the Interviewing Part

```
Require: $Arg$ as a set of arguments, $\sigma$ as the semantics
Ensure: An extension set $\mathbb{S}$
  $\mathbb{S} \leftarrow \emptyset$
  if $Exists_{ad}^{\neg\emptyset}$ then
      $Args_{\mathbb{S}} \leftarrow \emptyset$
      for all $a \in Arg$ do
          if $Cred_{ad}(a) = \top$ then
              $Args_{\mathbb{S}} \leftarrow Args_{\mathbb{S}} \cup \{a\}$
      for $i = 2..\Delta_{ad}(|Args_{\mathbb{S}}|)$ do
          if $ActualCount_{ad}^{i} = \top$ then
              $n \leftarrow i$
              break
      $\mathbb{S} \leftarrow \{\{\}\}$                                    ▷ Step I
      $Pairs_{\mathbb{S}} \leftarrow \{\}$
      $\mathcal{S} \leftarrow 2^{Args_{\mathbb{S}}} \setminus \mathbb{S}$
      for all $s \in \mathcal{S}$ do
          if $|\mathbb{S}| = n$ then                         ▷ Stop Condition
              break
                                                              ▷ Step II
          if $s \in \mathbb{S}$ then
              continue
                                                              ▷ Step III
          $r \leftarrow$ Ask agent $Ver_{\sigma}(s)$
          if $r = \top$ then
              $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$          ▷ Add $s$ to the extension set $\mathbb{S}$
              for all $p \in s \times s$ do
                  $Pairs_{\mathbb{S}} \leftarrow Pairs_{\mathbb{S}} \cup \{p\}$     ▷ Add pairs of arguments appearing

              for all $S \in \mathbb{S}$ do                            ▷ Step IV
                  if $S \cup s \notin \mathbb{S}$ then
                      $c \leftarrow \top$
                      if $\forall a, b \in S \cup s : (a, b) \in Pairs_{\mathbb{S}}$ then
                          $c \leftarrow \bot$
                      if $c \neq \top$ then        ▷ No conflict means potential for existing
                          if $Ver_{ad}(S \cup s)$ then
                              $\mathbb{S} \leftarrow \mathbb{S} \cup \{S \cup s\}$
  else
      $\mathbb{S} \leftarrow \{\{\}\}$
```

A.3. Complete Semantics

For the complete semantics, we have made further changes that go beyond the previous ones. Algorithm 15 represents these. Again, we use $Exists_{co}^{\neg\emptyset}$ and $ActualCount_{co}^{i}$ as well as $Cred_{co}$. A new addition is to be found in $Unique_{co}$. We put this after the question about credulously accepted arguments, because in the case that $|\mathbb{S}| = 1$ holds, these form the only extension. Everything else is as known.

A.4. Preferred Semantics

The previously mentioned changes for the complete semantics in algorithm 15 can also be applied to the preferred semantics. Again, $Exists_{pr}^{\neg\emptyset}$ and $ActualCount_{pr}^{i}$ as well as $Cred_{pr}$ are used, in the same way as before.

A.5. Stable Semantics

Lastly, there remains the stable semantics. For this, too, we have added the same parts in Algorithm 17 as, for example, in Algorithm 15 to the complete semantics.

Algorithm 15 Modified Algorithm for the Complete Semantics for the Interviewing Part

```
Require: $Arg$ as a set of arguments, $\sigma = co$
Ensure: An extension set $\mathbb{S}$
  $Args_{\mathbb{S}} \leftarrow \emptyset$
  $\mathbb{S} \leftarrow \{\}$
  if $Exists_{co}^{\neg\emptyset}$ then
      for all $a \in Arg$ do                                              ▷ Step I
          if $Cred_{co}(a) = \top$ then
              $Args_{\mathbb{S}} \leftarrow Args_{\mathbb{S}} \cup \{a\}$
      if $Unique_{co} = \bot$ then
          for $i = 2..\Delta_{co}(|Args_{\mathbb{S}}|)$ do
              if $ActualCount_{co}^{i} = \top$ then
                  $n \leftarrow i$
                  break

          $z \leftarrow$ Ask agent $Ver_{co}(\{\})$                       ▷ Step I
          if $z = \top$ then
              $\mathbb{S} \leftarrow \mathbb{S} \cup \{\{\}\}$
          $e \leftarrow \emptyset$
          $\mathcal{S} \leftarrow 2^{Args_{\mathbb{S}}} \setminus \{\{\}\}$
          for all $s \in \mathcal{S}$ do
              if $|\mathbb{S}| = n$ then                                  ▷ Step II
                  break
              if $z = \bot \wedge |\mathbb{S}| \geq 2 \wedge |(s \cap e)| \neq 0 \wedge (s \cap e) \nsubseteq e$ then   ▷ Step III
                  continue                  ▷ Intersection of all arguments not given
              $r \leftarrow$ Ask agent $Ver_{co}(s)$                     ▷ Step IV
              if $r = \top$ then
                  $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$
                  if $z = \bot \wedge |\mathbb{S}| \geq 2$ then          ▷ Step V
                      $e \leftarrow \bigcap_{S' \in \mathbb{S}} S'$       ▷ Find the intersecting arguments
      else
          $\mathbb{S} \leftarrow \{Args_{\mathbb{S}}\}$
  else
      $\mathbb{S} \leftarrow \{\{\}\}$
```

Algorithm 16 Modified Algorithm for the Preferred Semantics for the Interviewing Part

```
Require: $Arg$ as a set of arguments, $\sigma$ as the semantics
Ensure: An extension set $\mathbb{S}$
  $Args_{\mathbb{S}} \leftarrow \emptyset$
  $\mathbb{U} \leftarrow \{\}$
  $\mathbb{S} \leftarrow \{\}$
  if $Exists_{pr}^{\neg\emptyset}$ then
      for all $a \in Arg$ do                                                ▷ Step I
          if $Cred_{pr}(a) = \top$ then
              $Args_{\mathbb{S}} \leftarrow Args_{\mathbb{S}} \cup \{a\}$
      if $Unique_{pr} = \bot$ then
          for $i = 2..\Delta_{pr}(|Args_{\mathbb{S}}|)$ do
              if $ActualCount_{pr}^{i} = \top$ then
                  $n \leftarrow i$
                  break

          $Pairs_{\mathbb{S}} \leftarrow \{\}$
          $\mathcal{S} \leftarrow 2^{Args_{\mathbb{S}}}$
          for all $s \in \mathcal{S}$ do
              if $|\mathbb{S}| = n$ then                                      ▷ Step I
                  break
              if $s \in \mathbb{U}$ then                                      ▷ Step II
                  continue
              if $\exists s' \in \mathbb{S}$ s.t. $s \supset s' \vee s \subset s'$ then          ▷ Step III
                  continue
              if $\exists S' \in \{S \cup s | S \in \mathbb{S}\} \forall a, b \in S' : (a, b) \in Pairs_{\mathbb{S}}$ then   ▷ Step IV
                  continue
              $r \leftarrow$ Ask agent $Ver_{\sigma}(s)$                            ▷ Step V
              if $r = \top$ then
                  $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$
                  for all $p \in s \times s$ do
                      $Pairs_{\mathbb{S}} \leftarrow Pairs_{\mathbb{S}} \cup \{p\}$
                  if $|\mathbb{S} \geq 2|$ then                                 ▷ Step VI
                      for all $S, S' \in \mathbb{S}$ do
                          if $S \neq S'$ then
                              $U \leftarrow S \cup S'$
                              $\mathbb{U} \leftarrow \mathbb{U} \cup \{U\}$
      else
          $\mathbb{S} \leftarrow \{Args_{\mathbb{S}}\}$
  else
      $\mathbb{S} \leftarrow \{\{\}\}$
```

Algorithm 17 Modified Algorithm for the Stable Semantics for the Interviewing Part

```
Require: $Arg$ as a set of arguments, $\sigma = st$ as the semantics
Ensure: An extension set $\mathbb{S}$
  $Args_{\mathbb{S}} \leftarrow Arg$
  $\mathbb{S} \leftarrow \{\}$
  $Pairs_{\mathbb{S}} \leftarrow \{\}$
  $\mathcal{S} \leftarrow 2^{Args_{\mathbb{S}}}$
  if $Exists_{st} = \top$ then
    if $Exists_{st}^{\neg\emptyset} = \top$ then
      for all $a \in Arg$ do                                   ▷ Step I
        if $Cred_{pr}(a) = \top$ then
          $Args_{\mathbb{S}} \leftarrow Args_{\mathbb{S}} \cup \{a\}$
      if $Unique_{st} = \bot$ then
        for $i = 2..\Delta_{st}(|Args_{\mathbb{S}}|)$ do
          if $ActualCount_{st}^{i} = \top$ then
            $n \leftarrow i$
            break

        for all $s \in \mathcal{S}$ do
          if $|\mathbb{S}| = n$ then                              ▷ Step I
            break
          if $\exists s' \in \mathbb{S} : s \supset s' \vee s \subset s'$ then       ▷ Step II
            continue                      ▷ Incomparability is required
          $\mathcal{D} \leftarrow Args_{\mathcal{S}} \setminus s$
          $t \leftarrow \forall d \in \mathcal{D}\ \exists a \in s : (d, a) \notin Pairs_{\mathbb{S}}$      ▷ Step III
          if $t = \bot$ then
            continue                            ▷ $\mathbb{S} \cup \{s\}$ is not tight
          $r \leftarrow$ Ask agent $Ver_{\sigma}(s)$
          if $r = \top$ then                                    ▷ Step IV
            for all $p \in s \times s$ do
              Add $p$ to $Pairs_{\mathbb{S}}$
            $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$              ▷ Add $s$ to the extension set $\mathbb{S}$
      else
        $\mathbb{S} \leftarrow \{Args_{\mathbb{S}}\}$
    else
      $\mathbb{S} \leftarrow \{\{\}\}$
```

B. Comparing Two Solver Implementations on Performance

In this section of the appendix, we briefly compare two implementation types. In the main part of this Master's thesis, we used a Java-based implementation developed by ourselves for solving the computational tasks behind the questions. Since we also used the dataset of the ICCMA'19 [11] competition, we look at the solvers involved there. The solver mu-toksia was consistently leading in this competition. So we select it to demonstrate that the choice of solver in the elicitation has a relevance on the overall performance. Furthermore, we want to show that nonetheless, for extension sets with large cardinality, these also become impractical, i.e. the runtimes increase strongly. Since the competition did not consider conflict-free sets and admissible sets and the solver therefore did not implement them, we restrict ourselves to $\sigma \in \{co, gr, pr, st\}$ for the comparison.

To carry out the evaluation, we first compare the developed algorithms directly between our own Java implementation and one that uses the solver mu-toksia. Then we show that for the dataset pbbg-train even the modified algorithms in Appendix A are applicable, despite computational tasks in NP. Finally, we briefly show how this affects the ICCMA'19 dataset. All this is done using an $Agent_{\neg\mathbb{E}}^{dec,sem}$. We do not use graphical illustrations and stick purely to the results in tabular form.

B.1. Performance Comparison of the Developed Algorithms

At the beginning, we will briefly explain how the solver is accessed. For this purpose, the probo interface is used and the usual computational problems are applied. For $Cred_\sigma$ and $Scept_\sigma$ there are no surprises. The other questions, however, benefit from the fact that Ver_σ and other iterative checks are not implemented in the solvers. Thus $Enum_\sigma$ is performed once and thereafter each question is answered by a search in these extensions. This means that these solvers cannot be compared one-to-one with the Java implementation. However, the speed advantage of a complete enumeration should be fairly limited, which can be seen, among other things, in the results below, which show that the chosen solver is not several orders of magnitude faster than the simple, procedural Java implementation.

To evaluate the performance, we selected 50 out of the 1000 randomly selected instances of pbbg-train from Subsection 5.1.

The results can be seen in Table 21 and can be compared with the results from Subsection 5.2. Very briefly, it can be said that the performance of the developed algorithms with a faster solver is generally better than the procedural Java implementation from the main part. However, the results are not many orders of magnitude apart.

σ	**Data Bin**	$\overline{time}$	$\sum$ **T/O**	H_0 **(Wilcoxon)**	**Accepted?**
co	$5 \leq \|Arg\| < 10$	$1.06 \cdot 10^9$	0	$\tilde{x}_J \geq \tilde{x}_M$	yes
	$10 \leq \|Arg\| < 15$	$1.11 \cdot 10^9$	0		
	$15 \leq \|Arg\| < 20$	$2.69 \cdot 10^9$	0		
	$20 \leq \|Arg\| \leq 25$	$2.29 \cdot 10^9$	0		
gr	$5 \leq \|Arg\| < 10$	$6.67 \cdot 10^9$	0	$\tilde{x}_J \geq \tilde{x}_M$	no
	$10 \leq \|Arg\| < 15$	$8.82 \cdot 10^9$	0		
	$15 \leq \|Arg\| < 20$	$1.17 \cdot 10^{10}$	0		
	$20 \leq \|Arg\| \leq 25$	$2.00 \cdot 10^{10}$	0		
pr	$5 \leq \|Arg\| < 10$	$1.08 \cdot 10^9$	0	$\tilde{x}_J \geq \tilde{x}_M$	yes
	$10 \leq \|Arg\| < 15$	$1.15 \cdot 10^9$	0		
	$15 \leq \|Arg\| < 20$	$2.19 \cdot 10^9$	0		
	$20 \leq \|Arg\| \leq 25$	$9.84 \cdot 10^{10}$	0		
st	$5 \leq \|Arg\| < 10$	$1.18 \cdot 10^9$	0	$\tilde{x}_J \geq \tilde{x}_M$	yes
	$10 \leq \|Arg\| < 15$	$1.30 \cdot 10^9$	0		
	$15 \leq \|Arg\| < 20$	$2.85 \cdot 10^9$	0		
	$20 \leq \|Arg\| \leq 25$	$1.62 \cdot 10^{11}$	0		

Table 21: Performance representation per semantics. The hypothesis H_0 is tested by means of a one-sided Wilcoxon sign-rank test. $\tilde{x}_J$ is the CPU runtime of our own implementation and $\tilde{x}_M$ from mu-toksia.

B.2. Modified Algorithms Using a Faster Solver

In the main part of this Master's thesis we did not check the use of the modified algorithms with an $Agent_{\neg\mathbb{E}}^{dec,sem}$. The background is the poor performance in general. Since it was previously shown that with mu-toksia the performance is fundamentally better with the developed algorithms, we want to check their adaptation with questions of the complexity class NP, coNP or higher in the polynomial hierarchy. For this purpose, we used 50 out of the 1000 randomly selected instances from pbbg-train (cf. Subsection 5.1).

The results of this experiment are summarised in Table 22. A comparison between the two solvers (our Java implementation and mu-toksia) shows that mu-toksia was generally faster for the developed algorithms. This was to be expected, however, a comparison with the originally developed algorithms from Subsection 4.3 and the modified ones shows that only in a few cases, those were some larger instances, were the modified algorithms with mu-toksia faster. A direct comparison between the two algorithm variants for our Java implementation shows an even stronger picture (compare results from Table 22 with those from Subsection 5.2). It clearly shows that the modified ones are slower and that it was therefore a good choice to exclude questions from classes like NP.

Therefore, using only computational tasks in L or P turned out to be the right choice for the own solver implementation in general, but we point out that for other solvers a different picture may emerge in that there may be instances where the modified algorithms are faster. If $Cred_\sigma$ can be answered in a relatively short time and the search space of arguments is thereby significantly reduced, this may mean that instances with many arguments in the AF, but few different arguments in the extension sets, are also solvable. Large instances, on the other hand, are virtually impracticable with the both versions of the algorithms, even with fast solvers. In the next subsection, we will briefly examine this assumption.

σ	**Data Bin**	**Solver**	$\overline{time}$	$\sum$ **T/O**	H_0 **(Wilcoxon)**	**Accepted?**
co	$5 \leq \lvert Arg \rvert < 10$	own	$1.18 \cdot 10^8$	0	$\tilde{x}_J \geq \tilde{x}_M$	yes (barely)
		mu-toksia	$9.24 \cdot 10^9$	0		
	$10 \leq \lvert Arg \rvert < 15$	own	$5.76 \cdot 10^9$	0		
		mu-toksia	$1.23 \cdot 10^{10}$	0		
	$15 \leq \lvert Arg \rvert < 20$	own	$4.86 \cdot 10^9$	0		
		mu-toksia	$1.59 \cdot 10^{10}$	0		
	$20 \leq \lvert Arg \rvert \leq 25$	own	$6.02 \cdot 10^{11}$	0		
		mu-toksia	$2.42 \cdot 10^{10}$	0		
gr	$5 \leq \lvert Arg \rvert < 10$	own	$4.06 \cdot 10^7$	0	$\tilde{x}_J \geq \tilde{x}_M$	no
		mu-toksia	$6.96 \cdot 10^9$	0		
	$10 \leq \lvert Arg \rvert < 15$	own	$4.33 \cdot 10^7$	0		
		mu-toksia	$9.29 \cdot 10^9$	0		
	$15 \leq \lvert Arg \rvert < 20$	own	$5.85 \cdot 10^7$	0		
		mu-toksia	$1.25 \cdot 10^{10}$	0		
	$20 \leq \lvert Arg \rvert \leq 25$	own	$6.42 \cdot 10^7$	0		
		mu-toksia	$2.10 \cdot 10^{10}$	0		
pr	$5 \leq \lvert Arg \rvert < 10$	own	$3.00 \cdot 10^{11}$	6	$\tilde{x}_J \geq \tilde{x}_M$	yes
		mu-toksia	$8.06 \cdot 10^9$	0		
	$10 \leq \lvert Arg \rvert < 15$	own	$3.00 \cdot 10^{11}$	12		
		mu-toksia	$9.09 \cdot 10^9$	0		
	$15 \leq \lvert Arg \rvert < 20$	own	$3.00 \cdot 10^{11}$	8		
		mu-toksia	$9.77 \cdot 10^9$	0		
	$20 \leq \lvert Arg \rvert \leq 25$	own	$3.00 \cdot 10^{11}$	24		
		mu-toksia	$1.81 \cdot 10^{10}$	0		
st	$5 \leq \lvert Arg \rvert < 10$	own	$1.09 \cdot 10^8$	0	$\tilde{x}_J \geq \tilde{x}_M$	yes
		mu-toksia	$1.19 \cdot 10^{10}$	0		
	$10 \leq \lvert Arg \rvert < 15$	own	$5.32 \cdot 10^9$	0		
		mu-toksia	$1.16 \cdot 10^{10}$	0		
	$15 \leq \lvert Arg \rvert < 20$	own	$3.10 \cdot 10^9$	0		
		mu-toksia	$1.12 \cdot 10^{10}$	0		
	$20 \leq \lvert Arg \rvert \leq 25$	own	$6.42 \cdot 10^{11}$	14		
		mu-toksia	$2.44 \cdot 10^{10}$	0		

Table 22: Comparison of the performance of the own implementation (own) and another solver (mu-toksia) for the modified algorithms per semantics. The hypothesis H_0 is tested by means of a one-sided Wilcoxon sign-rank test. $\tilde{x}_J$ is the CPU runtime of our own implementation and $\tilde{x}_M$ from mu-toksia.

B.3. Large Instances with a Different Solver

Finally, we briefly summarise the results for larger instances. With an $Agent_{\neg\mathbb{E}}^{dec,sem}$ and the faster solver mu-toksia, significantly larger instances than in the previous experiment could not be solved. This is due to the fact that many of the questions such as $Cred_\sigma$ are dependent in runtime to the number of arguments. Since answering these questions with upper bounds in NP, coNP or higher seems to take a lot of time overall, larger instances become equally intractable. The limit here seems to be between 27 and 32 arguments in the AF of the instance.

σ	**Instance**	$\|$**Arg**$\|$	$\|\mathbb{S}\|$	$\|\|\mathbb{S}\|\|$	**T/O**
co	Small-result-b64	32	6	26	yes
gr	-	-	-	-	-
pr	Small-result-b60	32	4	32	yes
st	Small-result-b44	27	3	27	yes

Table 23: The small instances regarding the number of arguments that could not be solved with mu-toksia for the complete, grounded, preferred and stable semantics.

B.4. Summary of the Solver Evaluation

From the previous results it can be said that the choice of a solver does have an influence. It was surprising to see that questions initially excluded in the main part could be answered relatively quickly, at least compared to the procedural Java implementation. As a result, the modified algorithms could even solve larger instances better in cases with, however, small extensions, i.e. few arguments and low cardinality of the extension set. Nevertheless, it turned out that the modified algorithms did not always perform better with mu-toksia as the chosen solver and, above all, that many instances could still not be solved for the ICCMA datasets. The fundamental statements of the main part of this Master's thesis therefore remain unaffected by the choice of a solver. Only the point at which elicitation becomes impractical depends on the particular chosen solver.