FernUniversität in Hagen
Fakultät für Mathematik und Informatik

AIG
Artificial Intelligence Group

Algorithmen für initiale Mengen in abstrakter Argumentation

Bachelorarbeit

zur Erlangung des Grades *Bachelor of Science (B.Sc.)*
im Studiengang Informatik

vorgelegt von
Miles Buhr

Erstgutachter: Prof. Dr. Matthias Thimm
Artificial Intelligence Group

Betreuer: Lars Bengel
Artificial Intelligence Group

Erklärung

Ich erkläre, dass ich die Bachelorarbeit selbstständig und ohne unzulässige Inanspruchnahme Dritter verfasst habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht. Die Versicherung selbstständiger Arbeit gilt auch für enthaltene Zeichnungen, Skizzen oder graphische Darstellungen. Die Bachelorarbeit wurde bisher in gleicher oder ähnlicher Form weder derselben noch einer anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Mit der Abgabe der elektronischen Fassung der endgültigen Version der Bachelorarbeit nehme ich zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate geprüft werden kann und ausschließlich für Prüfungszwecke gespeichert wird.

Der Veröffentlichung dieser Arbeit auf der Webseite des Lehrgebiets Künstliche Intelligenz und damit dem freien Zugang zu dieser Arbeit stimme ich ausdrücklich zu.

Für diese Arbeit erstellte Software wurde quelloffen verfügbar gemacht, ein entsprechender Link zu den Quellen ist in dieser Arbeit enthalten. Gleiches gilt für angefallene Forschungsdaten.

Bispingen, 10.10.2025 .....

(Ort, Datum) (Unterschrift)

Zusammenfassung

Ausgehend von Dung [15], hat sich mit dem Formalismus der abstrakten Argumentationsgraphen ein breites Feld innerhalb der Künstlichen Intelligenz entfaltet, in dem insb. an algorithmischen Fragestellungen zur Auswertung von Argumentationsgraphen geforscht wird. Im Zentrum dieses Formalismus stehen sog. Extensionen. Diese sind Mengen von Argumenten, die gemeinsam einen Konflikt bestehen. Thimm [29] hat einen Ansatz zur iterativen Konstruktion von Extensionen vorgestellt, der zur parallelisierten Berechnung verwendet werden kann. Die Grundlage für dieses Verfahren bilden minimale, nicht-leere zulässige Mengen von Argumenten, sog. *initiale Mengen* [32]. In dieser Bachelorarbeit sollen verschiedene Ansätze zur Berechnung initialer Mengen bzgl. ihrer Performanz verglichen werden. Bengel und Thimm [6] schlagen eine iterative Anwendung eines SAT-Solvers zur Berechnung initialer Mengen vor. Diesem Verfahren soll ein direkter Test auf Minimalität einer nicht-leeren, zulässigen Menge, vorgeschlagen von Thimm [29], gegenübergestellt werden. Das Testverfahren beruht auf polynomiellen Algorithmen und prüft, ob eine konfliktfreie Menge an Argumenten eine initiale Menge ist, oder nicht. Für die Anwendung des Tests wird sich ein weiteres Ergebnis von Thimm [29] zunutze gemacht, nach dem eine initiale Menge in einer starken Zusammenhangskomponente eines Argumentationsgraphen liegen muss. Außerdem werden zwei Modifikationen des iterativen SAT-Ansatzes von Thimm und Bengel [6], auf Basis der Zerlegung eines Argumentationsgraphen in seine starken Zusammenhangskomponenten, betrachtet.

Abstract

Based on Dung [15], the formalism of abstract argumentation graphs has developed into a broad field within artificial intelligence, in which research is being conducted in particular on algorithmic issues relating to the evaluation of argument graphs. At the center of this formalism are so-called extensions. These are sets of arguments that together are able to survive a conflict. With the principle of serialisation, an approach for the iterative construction of extensions that can be used for parallelized computation was proposed by Thimm [29]. Serialisation of semantics rely on minimal, non-empty admissible sets, so-called *initial sets* [32]. In this bachelor thesis, different approaches for calculating initial sets will be compared with regard to their runtimes. Bengel and Thimm [6] proposed an iterative application of a SAT solver for calculating initial sets. Also, based on the encoding by Bengel and Thimm, two alternative SAT-encodings of of initial sets, that explicitly consider strongly connected components of an argumentation graph, will be considered and evaluated. These purely SAT-based approches are going to be compared with a direct test for minimality of a non-empty, admissible set, proposed by Thimm [29]. This test procedure is based on polynomial algorithms.

Inhaltsverzeichnis

1 Einleitung

Das Feld der formalen Argumentation ist eine Unterkategorie innerhalb der Künstlichen Intelligenz bzw. der Wissensrepräsentation. Mit Erscheinen der grundlegenden Arbeit von Dung [15], in der ein Formalismus entwickelt wurde, mit dem aus einer widersprüchlichen Wissensbasis systematisch Schlüsse gezogen werden können, begann sich ein breites Forschungsfeld zu algorithmischen Fragen bezüglich des Formalismus der formalen Argumentation zu entfalten. Viele algorithmische Fragestellungen auf dem Gebiet der abstrakten Argumentation haben sich als äußerst komplex erwiesen. Typische Entscheidungsprobleme im Kontext von abstrakten Argumentationsgraphen und deren Semantiken liegen in der Komplexitätsklasse $\mathcal{NP}$ [16]. Diese Arbeit wird sich mit *initialen Mengen* in abstrakter Argumentation und schwerpunktmäßig mit deren Berechnung beschäftigen. Motiviert wird das Thema dadurch, dass mit dem Konstruktionsverfahren der *Serialisierung* [29, 6] ein neuartiger, nicht-deterministischeer Ansatz zur Berechnung von Extensionen vorgeschlagen wird, der das Potenzial zur *parallelen* Berechnung von Extensionen bietet. Ausgangspunkt der Serialisierung von Extensionen sind minimale, nicht-leere zulässige Mengen. Ausgehend von diesen initialen Mengen können die Extensionen der klassischen, zulässigkeitsbasierten Semantiken nach Dung serialisiert werden.

In der Arbeit sollen vier konkrete Ansätze zur Berechnung der initialen Mengen bezüglich ihrer Performanz vergleichend evaluiert werden. Ein weit verbreiteter Ansatz zur Berechnung von Extensionen sind SAT-gestützte Verfahren, bei denen konkrete Probleme im Kontext von abstrakten Argumentationsgraphen aussagenlogisch modelliert und mögliche Lösungen mittels SAT-Solvern berechnet werden [13]. In [6] wird eine Implementierung für die Serialisierung von Extensionen präsentiert. Darin werden initiale Mengen mittels iterativer Aufrufe eines SAT-Solvers berechnet. In dieser Arbeit werden zwei Modifikationen, bzw. Erweiterungen dieses bestehenden Ansatzes vorgeschlagen und implementiert. Diese Verfahren beruhen auf einem Ergebnis von Thimm [29], nach dem sich die initialen Mengen eines abstrakten Argumentationsgraphen innerhalb dessen starken Zusammenhangskomponenten lokalisieren lassen.

Diesen rein SAT-basierten Verfahren soll ein Ansatz gegenübergestellt werden, der auf einem Testverfahren für die Minimalität nicht-leerer zulässiger Mengen beruht. Dieser Ansatz wurde von Thimm [29] vorgeschlagen. Es wurde gezeigt, dass dieser Test auf Minimalität in der Klasse $\mathcal{P}$ liegt, es also in polynomieller Zeit entschieden werden kann, ob eine gegebene konfliktfreie Menge eine initiale Menge ist, oder nicht [29].

Die Arbeit ist folgendermaßen strukturiert: In Kapitel 2 werden zunächst die grundlegenden Konzepte und Notationen im Zusammenhang mit abstrakten Argumentationsgraphen und deren Semantiken dargestellt und erläutert. Ausserdem werden aussagenlogische Grundlagen und die SAT-Kodierung von Extensionen dargestellt. In Kapitel 3 werden zunächst der Begriff der initialen Mengen eingeführt und die strukturellen Eigenschaften initialer Mengen dargestellt und erläutert. Anschließend wird

das Verfahren der Serialisierung von Semantiken erläutert und damit die Motivation für die effiziente Berechnung initialer Mengen noch einmal dargelegt. In Kapitel 4 werden die unterschiedlichen algorithmischen Ansätze zur Berechnung initialer Mengen in abstrakten Argumentationsgraphen dargestellt. Kapitel 5 wird die Implementierung des *InitialSetSolvers* kompakt dargestellt. Anschließend werden in Kapitel 6 die durchgeführten Experimente und die Ergebnisse der Auswertung des Laufzeitverhaltens der in dieser Arbeit betrachteten algorithmischen Ansätze zur Berechnung initialer Mengen dargestellt. Die Arbeit schließt mit einer Diskussion der Ergebinsse und einer Zusammenfassung.

2 Grundlagen

2.1 Abstrakte Argumentationsgraphen

Im Folgenden werden einige grundlegende Notationen und Konzepte der formalen Argumentation nach Dung [15] dargestellt und erläutert. Zunächst wird die Struktur abstrakter Argumentationsgraphen vorgestellt. Anschließend wird die Evaluation abstrakter Argumentationsgraphen anhand sog. Semantiken dargestellt. Die Darstellung dieser Grundlagen orientiert sich an Dung [15] und Baroni et. al. [2].

Im Zentrum des von Dung entwickelten Formalismus stehen *abstrakte Argumentationsgraphen*. Diese bilden eine widersprüchliche, bzw. konfliktbehaftete Wissensbasis als einen gerichteten Graphen ab, dessen Knoten die Argumente darstellen und eine Kante von Argument a zu einem Argument b einen Angriff von a auf b darstellt:

Definition 2.1. *Ein Argumentationsgraph $F = (A, R)$ besteht aus einem Tupel aus einer endlichen Menge A an Argumenten und einer binären Relation auf der Menge der Argumente $R \subseteq A \times A$, welche die Angriffe zwischen Argumenten enthält. Ein Angriff des Arguments a auf ein Argument b wird auch als $a\mathcal{R}b$ notiert.*

Definition 2.2. *Sei $F = (A, R)$. Für alle $a \in A$ sei $a_F^- = \{b \in A \mid b\mathcal{R}a\}$ die Menge aller Argumente in A, die a angreifen und $a_F^+ = \{b \in A \mid a\mathcal{R}b\}$ die Menge aller Argumente in A, die von a angegriffen werden.*

Die Definition der Angriffsrelation lässt sich auch auf Mengen von Argumenten erweitern. Sei $F = (A, R)$ und $S, S' \subseteq A$, $a \in S, b \in S'$. Wenn $a\mathcal{R}b$ gilt, wird dies mit $S\mathcal{R}S'$ notiert. Dementsprechend kann auch Definition 2.2 auf Mengen angewandt werden. Für eine Menge S ist $S_F^- = \{a \in A \mid \exists b \in S : a\mathcal{R}b\}$ die Menge aller Elemente in A, die S attackieren. Die Menge $S_F^+ = \{a \in A \mid \exists b \in S : b\mathcal{R}a\}$ ist die Menge aller Argumente in A, die von mindestens einem Argument in S attackiert werden.

Beispiel 2.1

Als Beispiel sei in Abbildung 1 ein abstrakter Argumentationsgraph $F_1 = (A_1, R_1)$ dargestellt. A_1 ist eine Menge an Argumenten $A_1 = \{a, b, c, d\}$ und die Angriffsrelation $R_1 = \{(a, b), (b, a), (c, b), (c, d)\}$ enthält die in F_1 dargestellten Konflikte in Form von Angriffen zwischen Argumenten. Die Menge b^- enthält alle Argumente, die b angreifen, $b^- = \{a, c\}$. Die Menge aller Argumente, die von b angegriffen werden b^+ besteht aus einem einzigen Argument, $b^+ = \{a\}$.

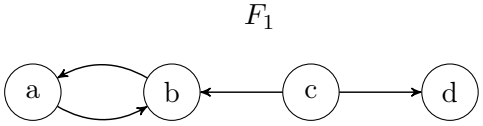

Abbildung 1: Der Argumentationsgraph F_1.

Für das Schlussfolgern, also der Auswertung eines Argumentationsgraphen mit Hinblick darauf, welche Argumente als gültig angesehen werden dürfen, wird in [15] das Kriterium der *Zulässigkeit* einer Menge von Argumenten definiert:

Definition 2.3. *Sei $F = (A, R)$ ein abstrakter Argumentationsgraph und $S \subseteq A$. S ist genau dann zulässig, wenn gilt:*

- $\forall a, b \in S : (a, b) \notin R$ *und* $(b, a) \notin R$
- $\forall b \in S_F^- : \exists c \in S$, *sodass* $(c, b) \in R$.

Danach gilt eine Menge $S \subseteq A$ als zulässig, gdw. zwei Forderungen erfüllt sind. Die erste Forderung besagt, dass sich keine Argumente innerhalb von S gegenseitig angreifen (**S ist konfliktfrei**). Als zweites wird gefordert, dass alle in S enthaltenen Argumente durch S **verteidigt** werden. Ein Argument $a \in S$, das durch ein Argument $b \in A$ angegriffen wird, wird dann verteidigt, wenn ein Argument $c \in S$ existiert, sodass $(c, b) \in R$ gilt.

Beispiel 2.2

Als Beispiel sei wieder der Argumentationsgraph F_1 betrachtet. Dieser enthält die folgenden Teilmengen von Argumenten, die konfliktfrei sind und alle ihre Argumente verteidigen, also zulässig sind: $\{\emptyset, \{a\}, \{c\}, \{a, c\}\}$. Die leere Menge $\emptyset$ ist trivialer Weise immer konfliktfrei und verteidigt sich immer selbst. Damit ist die leere Menge immer zulässig und jeder Argumentationsgraph $F = (A, R)$ besitzt mit der leeren Menge immer mindestens eine zulässige Menge.

Das Konzept der Verteidigung lässt sich mithilfe der *charakteristischen Funktion* formalisieren [15]:

Definition 2.4. *Sei $F = (A, R)$ ein Argumentationsgraph und $S \subseteq A$. Die charakteristische Funktion ist eine Abbildung $\Delta_F : 2^A \to 2^A$, sodass:*

$$\Delta_F(S) = \{a \in A | S \text{ verteidigt } a\}$$

Das Kriterium der Zulässigkeit lässt sich mithilfe der charakteristischen Funktion auch so ausdrücken, dass eine konfliktfreie Menge S genau dann zulässig ist, wenn $S \subseteq \Delta_F(S)$ gilt [15].

Eine Menge von Argumenten, die mindestens das Kriterium der Zulässigkeit erfüllt, wird als *Extension* bezeichnet. Die Zulässigkeit von Argumentmengen stellt eine Minimalanforderung an Mengen, dessen Argumente als legitim, bzw. haltbar aus einem Konflikt hervorgehen können. Dung [15] stellt eine Reihe von Anforderungen bezüglich zulässiger Mengen vor, mit denen das Ergebnis einer Argumentation differenzierter charakterisiert werden kann. Diese zusätzlichen Anforderungen an zulässige Mengen produzieren unterschiedliche Mengen von Argumenten, die einen Konflikt überstehen. Diese sogenannten *Semantiken* nach Dung [15] sind folgendermaßen definiert:

Definition 2.5. *Sei $F = (A, R)$ ein Argumentationsgraph.*

- *Eine Menge $S \subseteq A$ ist **vollständig** (co), gdw. sie zulässig ist und für jedes Argument a in A, das von S verteidigt wird, gilt $a \in S$.*
- *Eine Menge $S \subseteq A$ ist **präferiert** (pr), gdw. sie vollständig ist und es keine vollständige Menge $S' \subseteq A$ gibt, sodass $S \subset S'$ gilt, S also maximal bzgl. Mengeninklusion ist.*
- *Eine Menge $S \subseteq A$ ist **grundiert** (gr), gdw. sie vollständig ist und es keine vollständige Menge $S' \subseteq A$ gibt, sodass $S' \subset S$ gilt, S also minimal bzgl. Mengeninklusion ist.*
- *Eine Menge $S \subseteq A$ ist **stabil** (st), gdw. sie vollständig ist und für jedes Argument a in A entweder $a \in S$ gilt, oder es mindestens ein Argument $b \in S$ gibt, sodass $(b, a) \in R$.*

Zur Veranschaulichung dieser klassischen, zulässigkeitsbasierten Semantiken wird das folgende Beispiel betrachtet.

Beispiel 2.3

F_2 (siehe Abb. 2) enthält die folgenden vollständigen Mengen:

- $co(F_2) = \{\{a, c\}, \{a, c, f\}, \{a, c, g\}\}$

Jede Menge an Argumenten in $co(F_2)$ ist zulässig und enthält alle Argumente, die auch von der entsprechenden Menge verteidigt werden. Mit Ausnahme von $\{a, c\}$ sind die Mengen in $co(F_2)$ maximal bezüglich Mengeninklusion:

- $pr(F_2) = \{\{a, c, f\}, \{a, c, g\}\}$

Die Menge $\{a, c\}$ ist vollständig und minimal bezüglich Mengeninklusion und ist daher die grundierte Extension des Argumentationsgraphen:

- $gr(F_2) = \{\{a, c\}\}$

Ausserdem gilt für alle präferierten Extensionen $S \in pr(F_2)$, dass diese alle Argumente in $A \setminus S$ attackieren, und somit ebenfalls stabile Extensionen sind:

- $st(F_2) = \{\{a, c, f\}, \{a, c, g\}\}$

Definition 2.6. *Sei $F = (A, R)$ ein Argumentationsgraph und $\sigma \in \{cf, adm, co, pr, gr, st\}$. Mit $\sigma(F) \subseteq 2^A$ wird die Menge aller σ-Extensionen des Argumentationsgraphen F bezeichnet.*

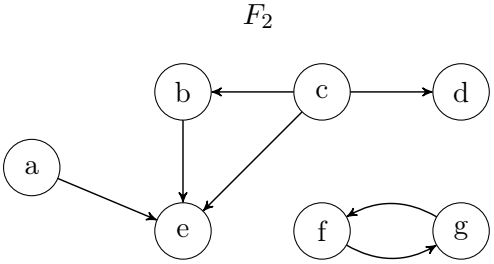

Abbildung 2: Der Argumentationsgraph F_2.

Ausgehend von einer Menge E an Argumenten, lässt sich ein Argumentationsgraph F in einen Teilgraphen, dem sog. E-Redukt von F, F^E, überführen [3]:

Definition 2.7. *Sei $F = (A, R)$ ein abstrakter Argumentationsgraph und $E \subseteq A$. Sei ausserdem E_F^+ die Menge aller Argumente in A, die von E attackiert werden. Das E-Redukt ist der Argumentationsgraph $F^E = (E^*, R \cap (E^* \times E^*))$ mit $E^* = A \setminus (E \cup E_F^+)$.*

Zur Veranschaulichung der Notation des E-Redukts, sei wieder der Argumentationsgraph F_2 betrachtet.

Beispiel 2.4

Sei $E = \{a, f\}$, das E-Redukt F_2^E würde folgendermaßen aussehen:

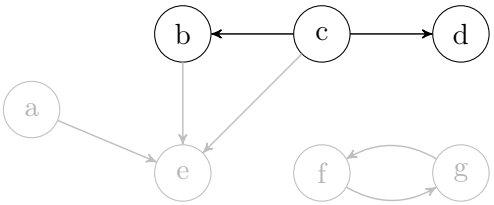

Abbildung 3: Das E-Redukt F_2^E.

Nachdem in diesem Abschnitt der Formalsimus der abstrakten Argumentationsgraphen vorgestellt und die wichtigsten Notationen dargestellt und erläutert wurden, werden im folgenden Abschnitt die aussagenlogischen Grundlagen zur Berechnung von Extensionen mittels SAT-Solvern dargelegt.

2.2 SAT Kodierungen für abstrakte Argumentationsgraphen

Ein häufig verwendeter Ansatz bei der Konstruktion von Extensionen ist die Reduktion eines Problems im Kontext abstrakter Argumentationsgraphen auf das *SAT-Problem* [13]. Um das SAT-Problem zu definieren, müssen zunächst einige grundle-

gende aussagenlogische Begriffe eingeführt werden. Bei der Darstellung dieser Grundlagen wird sich im Folgenden an der Darstellung in Beierle, Kern-Isberner [4] und Sauerwald, Thimm [26] orientiert. Anschließend wird dargelegt, wie mithilfe der Aussagenlogik bestimmte Extensionen eines Argumentationsgraphen modelliert werden können.

Die Aussagenlogik ist ein Formalismus bestehend u.a. aus einer *Signatur* Σ, einer Menge an *Formeln* $\mathcal{F}_\Sigma$, sowie einer Menge an *Interpretationen* $\mathcal{I}_\Sigma$. Die Signatur ist eine Menge von *Aussagenvariablen*, auch *Atome* genannt, welche als Bezeichner für die Elemente einer Wissensbasis dienen. Einer Aussagenvariablen kann ein *Wahrheitswert* aus der Menge $\mathcal{B} = \{true, false\}$ zugewiesen werden.

Definition 2.1. *Eine Interpretation ist eine Abbildung*

$$I : \Sigma \to \mathcal{B}$$

welche jeder aussagenlogischen Variablen einer Signatur Σ einen Wahrheitswert zuordnet. Die Menge aller Interpretationen einer Signatur Σ wird mit $\mathcal{I}_\Sigma$ bezeichnet.

Mithilfe von *Junktoren* $\{\neg, \wedge, \vee, \to\}$ lassen sich aus einer Menge von Aussagenvariablen komplexe *Formeln* bilden.

Definition 2.2. *Zur Bildung von aussagenlogischen Formeln gilt $a, \neg a \in \mathcal{F}_\Sigma$ für jedes $a \in \Sigma$. Für Formeln $A, B \in \mathcal{F}_\Sigma$ gilt, dass $\neg A, \neg B, A \wedge B, A \vee B$ und $A \to B \in \mathcal{F}_\Sigma$.*

Die Semantik der Junktoren ist folgendermaßen definiert:

Definition 2.3. *Sei $a \in \Sigma$, $A, B \in \mathcal{F}_\Sigma$ und $I \in \mathcal{I}_\Sigma$ eine Interpretation. Der Wahrheitswert einer Formel lässt sich mithilfe einer Erfüllungsrelation $\models \subseteq \mathcal{I}_\Sigma \times \mathcal{F}_\Sigma$ definieren:*

- *$I \models a$, falls $I(a) = true$*
- *Negation: $I \models \neg A$, falls $I \not\models A$*
- *Konjunktion: $I \models A \wedge B$, falls $I \models A$ **und** $I \models B$*
- *Disjunktion: $I \models A \vee B$, falls $I \models A$ **oder** $I \models B$*
- *Implikation: $I \models A \to B$, falls $I \models \neg A$ **oder** $I \models B$*

Für eine bestimmte Belegung aussagenlogischer Variablen einer Formel lässt sich damit auswerten, ob eine Formel erfüllt ist oder nicht. In diesem Kontext wird auch von *Modellen* gesprochen:

Definition 2.4. *Sei Σ eine aussagenlogische Signatur und $I \in \mathcal{I}_\Sigma$ eine Interpretation. Für eine aussagenlogische Formel $A \in \mathcal{F}_\Sigma$ heißt I Modell, gdw. $I \models A$. Die Menge aller Modelle einer Formel wird mit $\boldsymbol{Mod}_\Sigma(A)$ bezeichnet.*

Mit diesen aussagenlogischen Grundlagen lässt sich nun das SAT-Problem definieren.

SAT	
Eingabe:	Eine aussagenlogische Signatur Σ und eine Menge an Formeln $\mathcal{F}_\Sigma$ über Σ
Frage:	Existiert für eine Interpretation $I \in \mathcal{I}_\Sigma$, sodass für jede Formel $F \in \mathcal{F}_\Sigma$ $I \models F$ gilt?

Das SAT-Problem ist $\mathcal{NP}$-vollständig [14]. Mithilfe sog. SAT-Solver kann systematisch nach Interpretationen gesucht werden, die bestimmte Formeln erfüllen.

Für die folgende Darstellung, wie Semantiken eines Argumentationsgraphen als aussagenlogisches Problem abgebildet werden können, wird sich an der Darstellung in [13] orientiert.

Für die aussagenlogische Modellierung von Semantiken eines Argumentationsgraphen werden nun Formeln konstruiert, die genau dann erfüllt sind, wenn die dazu korrespondierende Menge an Argumenten eine Extension dieser Semantik darstellt. Das folgende Beispiel verdeutlicht, wie die zulässigen Extensionen eines Argumentationsgraphen aussagenlogisch modelliert und konstruiert werden können.

Beispiel 2.5

Es wird der Argumentationsgraph $F_1 = (A_1, R_1)$ betrachtet. Sei Σ_{F_1} eine Menge an Atomen und für jedes Argument $a \in A$ existiert ein entsprechendes Atom $a \in \Sigma_{F_1}$. Die aussagenlogische Modellierung der Zulässigkeit einer Menge von Argumenten lässt sich beispielsweise mittels folgender Formel kodieren (verändert übernommen aus Charwat et. al. [13], S. 34):

$$adm_F = \bigwedge_{a \in A} \Big((\neg a \vee (\bigwedge_{b \in a^-} \neg b)) \wedge (\neg a \vee \bigwedge_{b \in a^-} (\bigvee_{c \in b^-} c)) \Big) \tag{1}$$

Durch die Teilformel $cf = (\neg a \vee (\bigwedge_{b \in a^-} \neg b))$ wird sichergestellt, dass ein Argument a entweder verworfen wird, oder alle Argumente b, die a attackieren, verworfen werden. Dadurch wird die Konfliktfreiheit sichergestellt.

Die Teilformel $ad = (\neg a \vee \bigwedge_{b \in a^-} (\bigvee_{c \in b^-} c))$ stellt sicher, dass ein Argument a entweder verworfen wird, oder für jedes Argument b, welches a attackiert, ein Argument c akzeptiert wird, das den Angreifer b attackiert. Dadurch wird sichergestellt, dass jedes Argument, das akzeptiert wird, verteidigt wird. Die Konjunktion über beide Teilformeln sichert somit zu, dass Gleichung (1) nur dann erfüllt wird, wenn die entsprechende Interpretation mit einer zulässigen Menge an Argumenten korrespondiert.

Für den abstrakten Argumentationsgraphen F_1 erfüllen die folgenden Interpretationen Gleichung (1):

- $I_1 := a = false, b = false, c = false, d = false$
- $I_2 := a = true, b = false, c = false, d = false$

- $I_3 := a = false, b = false, c = true, d = false$
- $I_4 := a = true, b = false, c = true, d = false$

Die Interpretationen I_1, I_2, I_3 und I_4 korrespondieren mit den Argumentmengen $\emptyset, \{a\}$, $\{c\}$ und $\{a, c\}$. Diese entsprechen genau den zulässigen Mengen in F_1.

Die im vorangegangenen Beispiel dargestellte Korrespondenz zwischen den Modellen einer Gleichung und den Argumenten eines Argumentationsgraphen wird folgendermaßen formalisiert [7]:

Definition 2.5. *Sei $F = (A, R)$ ein abstrakter Argumentationsgraph, $a \in A$, Σ_F eine aussagenlogische Signatur, die für jedes Argument in A ein entsprechendes Atom enthält. Für eine Formel $\Psi_{\sigma,S} \in \mathcal{F}_{\Sigma_F}$ und eine Argumentmenge $S \subseteq A$ gilt, dass S eine σ-Extension ist, gdw. $\Psi_{\sigma,S}$ erfüllbar ist.*

3 Initiale Mengen

In diesem Kapitel werden initiale Mengen zunächst definiert und einige Eigenschaften von initialen Mengen dargestellt und erläutert. Anschließend wird mit der *Serialisierung* von Semantiken [29, 5] eine wichtige Anwendung für die effiziente Berechnung initialer Mengen vorgestellt.

3.1 Definition und Eigenschaften

Initiale Mengen sind nicht-leere, minimale zulässige Mengen von Argumenten. Der Begriff der initialen Menge wurde von Cayrol und Xu [32] geprägt.

Definition 3.1. *Sei $F = (A, R)$ ein abstrakter Argumentationsgraph. Eine Menge $S \subseteq A$ heißt initial, gdw. $S \neq \emptyset$ und S zulässig ist und keine echte Teilmenge $S' \subset S$ mit $S' \neq \emptyset$ existiert, die ebenfalls zulässig ist. Die Menge aller initialen Mengen von F wird mit $is(F)$ bezeichnet.*

Beispiel 3.1

$F_1 = (A_1, R_1)$ enthielt die folgenden zulässigen Mengen: $\{\emptyset, \{a\}, \{c\}, \{a, c\}\}$. Die leere Menge $\emptyset$ kann per Definition keine initiale Menge sein. Die Menge $\{a, c\}$ kann ebenfalls nicht inital sein, denn sowohl $\{a\} \subsetneq \{a, c\}$, als auch $\{c\} \subsetneq \{a, c\}$ sind zulässige, nichtleere Mengen. F_1 enthält somit zwei initiale Mengen, $\{a\}$ und $\{c\}$. Beide Mengen sind nicht-leer, zulässig und minimal bzgl. Mengeninklusion.

F_1

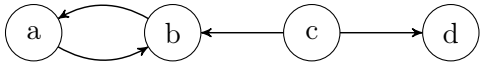

Abbildung 4: Der Argumentationsgraph F_1.

Anhand Beispiel 3.1 wird deutlich, dass initiale Mengen die Lösungen für die kleinstmöglichen Konflikte eines Argumentationsgraphen darstellen. Sowohl a als auch c sind Argumente, die in der Lage sind den Konflikt zu bestehen ohne dabei auf andere Argumente angewiesen zu sein. Die Menge $\{a, c\}$ ist ebenfalls zulässig, doch ist weder der Status bzgl. der Zulässigkeit von a abhängig von Argument c, noch umgekehrt. In [29] wird dies mit dem Begriff der *Relevanz* von Argumenten bzgl. der Zulässigkeit einer Menge von Argumenten bezeichnet.

Die initialen Mengen eines abstrakten Argumentationsgraphen $F = (A, R)$ lassen sich vollständig und eindeutig in drei unterschiedliche Klassen einordnen [29, 30]:

Definition 3.2. *Sei $F = (A, R)$ ein abstrakter Argumentationsgraph und $is(F)$ die Menge aller initialen Mengen von F. Eine initiale Menge $S \in is(F)$ lässt sich genau in eine der folgenden Klassen von initialen Mengen einordnen:*

- ***unattackiert**: $S \in is^{\nleftarrow}(F)$, gdw. $S_F^- = \emptyset$.*
- ***alleinstehend**: $S \in is^{\nleftrightarrow}(F)$, gdw. $S_F^- \neq \emptyset$ und kein $S' \in is(F)$ existiert, so dass $a \in S, b \in S'$ und $b\mathcal{R}a$.*
- ***herausgefordert**: $S \in is^{\leftrightarrow}(F)$, gdw. $S_F^- \neq \emptyset$ und ein $S' \in is(F)$ existiert, so dass $a \in S, b \in S'$ und $b\mathcal{R}a$.*

Zur Verdeutlichung dieser Klassifikation der initialen Mengen eines abstrakten Argumentationsgraphen sei das folgende Beispiel betrachtet.

Beispiel 3.2

Das Argument a ist eine initiale Menge und da kein Argument a attackiert, stellt a die (einzige) unattackierte initiale Menge in F_3 dar. Somit gilt $is^{\nleftarrow}(F_3) = \{\{a\}\}$.

Ebenso existiert eine alleinstehende initiale Menge $is^{\nleftrightarrow}(F_3) = \{h\}$. Das Argument h ist ebenso wie a eine initiale Menge, wird jedoch von d attackiert. Da d jedoch in keiner initialen Menge in F_3 enthalten ist, stellt $\{\{h\}\}$ somit eine alleinstehende initiale Menge dar.

Desweiteren existieren drei herausgeforderte initiale Mengen, $is^{\leftrightarrow}(F_3) = \{\{b, g\}, \{f, c\}\}$. Jeder dieser Mengen in $is^{\leftrightarrow}(F_3)$ ist eine initiale Menge und wird von mindestens einer weiteren initialen Menge attackiert.

F_3

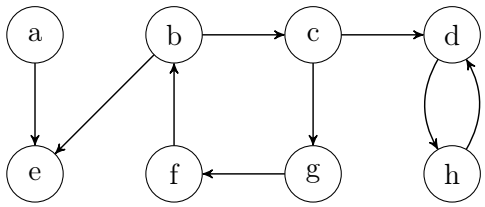

Abbildung 5: Der Argumentationsgraph F_3.

Aus der Definition initialer Mengen lässt sich die folgende Eigenschaft für unattackierte initiale Mengen $S \in is^{\nleftarrow}(F)$ ableiten:

Proposition 3.1. *Sei $F = (A, R)$ ein abstrakter Argumentationsgraph und $S \subseteq A$ eine unattackierte initiale Menge, $S \in is^{\nleftarrow}(F)$. Dann gilt, dass die Menge S exakt ein Argument enthält, also $|S| = 1$ gilt [29].*

Eine wichtige strukturelle Eigenschaft initialer Mengen wurde von Thimm [29] gezeigt. Demnach lässt sich eine initiale Menge S, immer in einer sog. *starken Zusammenhangskomponente* eines Argumentationsgraphen $F = (A, R)$ lokalisieren. Um

diese zu definieren, wird zuvor die Notation für die *Projektion* eines Argumentationsgraphen definiert.

Definition 3.3. *Sei $F = (A, R)$ ein Argumentationsgraph und $S \subseteq A$. Die Projektion von F auf S ist ein Argumentationsgraph $F|_S = (S, R \cap (S \times S))$.*

Mit dem Begriff der Projektion lassen sich nun starke Zusammenhangskomponenten für einen Argumentationsgraphen folgendermaßen definieren:

Definition 3.4. *Sei $F = (A, R)$ ein Argumentationsgraph, $S \subseteq A$ und $F|_S$ die Projektion von F auf S. $F|_S$ ist ein starke Zusammenhangskomponente, wenn für alle Argumente $a, b \in S$ ein Pfad zwischen a und b existiert und kein $S^{'}$ mit $S \subsetneq S^{'}$ existiert, das ebenfalls diese Bedingung erfüllt. Ein Pfad P zwischen a und b ist ein Tupel $P = (a_1, a_2, \ldots, a_n)$ mit $a_1 = a, a_n = b$, sodass $(a_i, a_{i+1}) \in R$ für alle $1 \leq i \leq n-1$.*

Proposition 3.2. *Sei $SCC(F)$ die Menge aller starken Zusammenhangskomponenten eines Argumentationsgraphen $F = (A, R)$ und $S \subseteq A$. Wenn S eine initiale Menge von F ist, dann existiert eine Projektion $F|_{A'} \in SCC(F)$ mit $A^{'} \subseteq A$, sodass $S \subseteq A^{'}$ [29].*

Proposition 3.2 besagt somit, dass jede initiale Menge in einer starken Zusammenhangskomponente des Ausgangsgraphen liegen muss. Enthält ein Argumentationsgraph mehr als eine starke Zusammenhangskomponente werden die Suchräume für zulässige Mengen verkleinert. Statt die gesamte Menge an Argumenten des Ausgangsgraphen zu betrachten, werden die (bei mehr als einer Zusammenhangskomponente) kleineren Argumentmengen der Zusammenhangskomponenten untersucht.

Beispiel 3.3

Der Argumentationsgraph $F_4 = (A_4, R_4)$ lässt sich in folgende starke Zusammenhangskomponenten zerlegen: $F_4|_{\{a,b,e\}}, F_4|_{\{f\}}, F_4|_{\{c,d\}}, F_4|_{\{g\}}, F_4|_{\{h\}}$. Die initialen Mengen dieses Graphen sind $is(F_4) = \{\{c\}, \{d\}, \{g\}\}$, mit $\{c\} \subseteq F_4|_{\{c,d\}}, \{d\} \subseteq F_4|_{\{c,d\}}$, und $\{g\} \subseteq F_4|_{\{g\}}$.

F_4

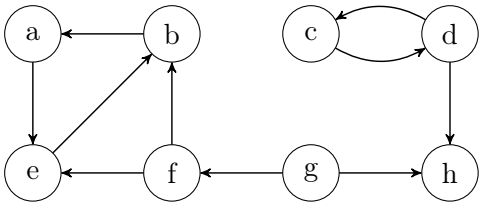

Abbildung 6: Der Argumentationsgraph F_4.

Nachdem nun initiale Mengen für abstrakte Argumentationsgraphen definiert und einige Eigenschaften initialer Mengen dargestellt wurden, soll nun anschließend das Konstruktionsverfahren der Serialisierung von Extensionen erläutert werden. Dieses Verfahren beruht auf der Auswahl initialer Mengen zur iterativen Konstruktion von Extensionen und motiviert daher die effiziente Berechnung initialer Mengen.

3.2 Initiale Mengen im Kontext der Serialisierung von Semantiken

Aufbauend auf den Arbeiten von Xu, Xu und Cayrol [32, 33] zu initialen Mengen und deren Beziehungen zu den (klassischen) zulässigkeitsbasierten Semantiken, hat Thimm [5, 29] mit der *Serialisierung* von Semantiken einen Formalismus zur iterativen Konstruktion von Extensionen für zulässigkeitsbasierte Semantiken vorgestellt. Da dieses Verfahren eine zentrale Anwendung für initiale Mengen darstellt, soll dieses im Folgenden kurz dargestellt und erläutert werden. Bei der folgenden Darstellung wird sich an Thimm [29, 30] orientiert.

Zunächst soll das Grundprinzip erläutert werden, nachdem Extensionen einer beliebigen serialisierbaren Semantik σ_{ser} konstruiert werden. In Algorithmus 1 wird beispielhaft die Serialisierung von Extensionen der vollständigen Semantik dargestellt.

Algorithmus 1: Grundprinzip Serialisierung

Eingabe $F = (A, R)$, *Selektionsfunktion* $\alpha_{\mathbf{co}}$, *Terminierungsfunktion* $\beta_{\mathbf{co}}$
Ausgabe $E \in \sigma_{\mathbf{co}}(F)$

```
1 E = ∅;
2 i = 1;
3 while is^↜(F) ≠ ∅ do
4 |   Wähle S_i ∈ is(F^E);
5 |   E = E ∪ S_i;
6 |   i++;
  end
7 return E;
```

Ausgehend von einem Argumentationsgraphen F wird eine initiale Menge $S_1 \in is(F)$ ausgewählt. Anschließend wird das S_1-Redukt von F, F^{S_1} gebildet und wiederum eine initiale Menge aus diesem Redukt, $S_2 \in is(F^{S_1})$, ausgewählt. Dieses Vorgehen wird solange wiederholt, bis keine unattackierten initialen Mengen mehr in einem Redukt existieren und damit die Terminationsbedingung $\beta_{\mathbf{co}}$ (s.u.) erfüllt ist.

Initiale Mengen können grundsätzlich als die atomaren Bausteine für nicht-leere, zulässige Mengen in einem Argumentationsgraphen betrachtet werden. Zur Veranschaulichung soll das folgende Beispiel betrachtet werden.

Beispiel 3.4

Der Argumentationsgraph F_5 besitzt die folgenden initialen Mengen, $is(F_5) = \{\{b\}, \{d\}, \{f\}, \{g\}\}$. Angenommen, zunächst wird die initiale Menge $S_1 = \{b\}$ gewählt. Anschließend wird das S_1-Redukt $F_5^{S_1}$ gebildet (Abb. 7 (b)).
Im Redukt $F_5^{S_1}$ sind drei initiale Mengen enthalten, $is(F_5^{S_1}) = \{\{d\}, \{f\}, \{g\}\}$, aus denen wieder eine ausgewählt wird. In diesem Fall wird $\{d\}$ gewählt. Das resultierende Redukt $F_5^{\{b,d\}}$ ist in Abb. 7 (c) abgebildet. Der Graph $F_5^{\{b,d\}}$ enthält eine initiale Menge, $\{f\}$. Die Vereinigung der im Serialisierungsprozess ausgewählten initialen Mengen $\{\{b\}, \{d\}, \{f\}\}$ ist eine nicht-leere, zulässige Menge in F_5.

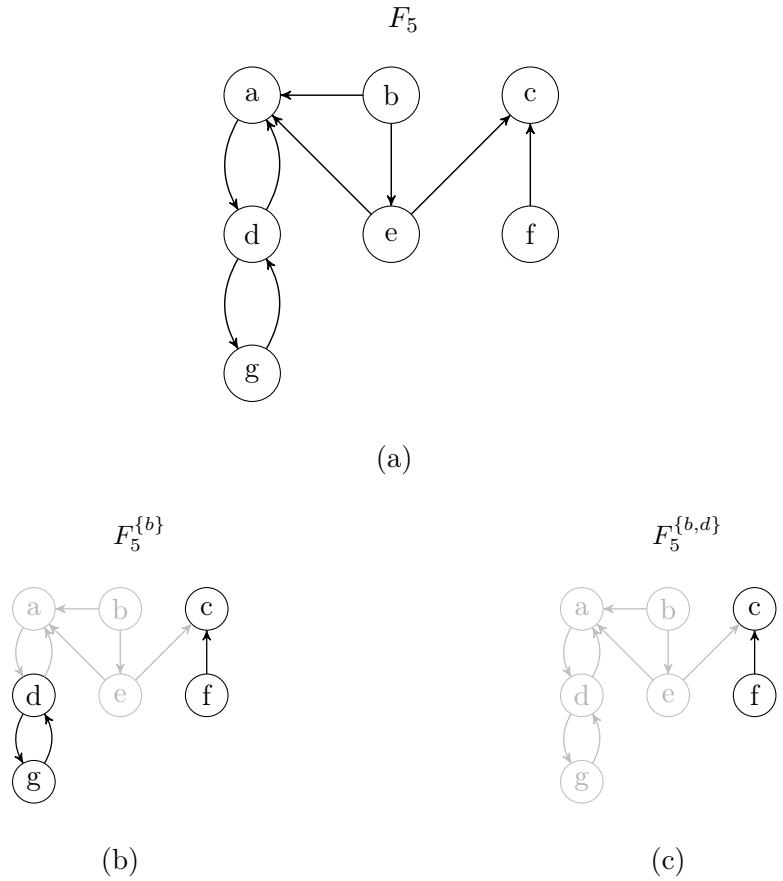

Abbildung 7: Serialisierungssequenz der zulässigen Menge $\{b, d, f\}$ im Argumentationsgraphen F_5.

Das Beispiel macht deutlich, dass die Serialisierung einer nicht-leeren, zulässigen Menge nicht eindeutig ist. Einerseits stehen während eines Serialisierungsschrittes

häufig mehrere initialen Mengen zur Auswahl und andererseits ist auch die Reihenfolge der Auswahl initialer Mengen nicht eindeutig. So könnte beispielsweise im vorangegangenen Beispiel anstelle der Menge $\{b\}$ auch zunächst die Menge $\{f\}$ als erstes ausgewählt werden. In beiden Fällen würde das resultierende Redukt die Mengen $\{d\}$ und $\{g\}$ enthalten. Da diese beiden Mengen sich gegenseitig attackieren, ist in Abhängigkeit, welche der Mengen ausgewählt wird, die jeweils andere Menge nicht im resultierenden Redukt enthalten. So entstehen in Abhängigkeit der gewählten initialen Mengen unterschiedliche Extensionen. Dies soll verdeutlichen, dass das Verfahren der Serialisierung grundsätzlich nicht-deterministisch ist. Ausserdem bietet der Umstand, dass die unterschiedliche Auswahl initialer Mengen zu unterschiedlichen Extensionen führt, das Potenzial zur parallelen Berechnung von Extensionen [6].

Die Konstruktion zulässiger Mengen stellt den verallgemeinerten Fall für das Verfahren der Serialisierung von Extensionen dar. Unabhängig davon, welche initialen Mengen ausgewählt werden und wann die Serialisierung gestoppt wird, werden immer zulässige Mengen entstehen.

Proposition 3.3. *Sei $F = (A, R)$. Jede nicht-leere, zulässige Menge S kann als Vereinigung von paarweise disjunkten Mengen S_i, $S = S_1 \cup \cdots \cup S_n$ formuliert werden, wenn gilt, dass $S_1 \in is(F)$ und für $1 < i \leq n$, $S_i \in is(F^{S_1 \cup \cdots \cup S_{n-1}})$ gilt [29].*

Durch Spezialisierungen mittels Selektions- und Terminationsfunktionen können Extensionen der spezielleren zulässigkeitsbasierten Semantiken konstruiert werden. Im Folgenden seien mit X, Y und Z die unattackierten $is^{\nleftarrow}(F)$, alleinstehenden $is^{\nleftrightarrow}(F)$ bzw. die herausgeforderten $is^{\leftrightarrow}(F)$ initialen Mengen eines abstrakten Argumentationsgraphen F bezeichnet [29]:

Definition 3.5. *Sei $\mathfrak{A}$ die universelle Menge von Argumenten. Eine Selektionsfunktion α ist definiert als eine Abbildung*

$$\alpha : 2^{2^{\mathfrak{A}}} \times 2^{2^{\mathfrak{A}}} \times 2^{2^{\mathfrak{A}}} \to 2^{\mathfrak{A}}$$

mit $\alpha(X, Y, Z) \subseteq X \cup Y \cup Z$, für alle $X, Y, Z \subseteq \mathfrak{A}$.

Mithilfe der Selektionsfunktion α kann die Auswahl initialer Mengen auf bestimmte Klassen initialer Mengen beschränkt werden. Sollen beispielsweise nur unattackierte initiale Mengen im Verlaufe des Serialisierungsprozesses ausgewählt werden, würde dies mit $\alpha(X, Y, Z) = X$ notiert.

Für die Charakterisierung der Serialisierung von Extensionen speziellerer Semantiken wird neben einer Auswahlfunktion α auch eine Terminierungsfunktion benötigt, mit der das Ende eines Serialisierungsprozesses angezeigt wird [29]:

Definition 3.6. *Sei $\mathfrak{F}$ die universale Menge aller abstrakten Argumentationsgraphen. Eine Terminierungsfunktion β ist eine Abbildung*

$$\beta : \mathfrak{F} \times 2^{\mathfrak{A}} \to \{0, 1\}$$

sodass $\beta : \mathfrak{F} \times 2^{2^{\mathfrak{A}}} = 1$ die Bedingung zum Stoppen eines Serialisierungsprozesses kodiert.

	$\sigma = \mathbf{adm}$	$\sigma = \mathbf{co}$	$\sigma = \mathbf{pr}$	$\sigma = \mathbf{gr}$	$\sigma = \mathbf{st}$
$\alpha_\sigma(X,Y,Z) =$	$X \cup Y \cup Z$	$X \cup Y \cup Z$	$X \cup Y \cup Z$	X	$X \cup Y \cup Z$
$\beta_\sigma(F,S) = 1$	1	$is^{\nleftarrow}(F) = \emptyset$	$is(F) = \emptyset$	$is^{\nleftarrow}(F) = \emptyset$	$F = (\emptyset, \emptyset)$
$\beta_\sigma(F,S) = 0$	/	sonst	sonst	sonst	sonst

Tabelle 1: Selektions- und Terminierungsfunktionen der Serialisierungen der klassischen Semantiken [29].

In Tabelle 1 ist eine Übersicht der Selektions- und Terminierungsfunktionen für die klassischen Dung Semantiken dargestellt. Abschließend soll zur Illustration die Serialisierung der Extension der grundierten Semantik in einem Argumentationsgraphen beispielhaft dargestellt werden.

Beispiel 3.5

Als Beispiel wird der Argumentationsgraph F_2 in Abb. 8 betrachtet. Für die Serialisierung der grundierten Extension werden gemäß der Auswahlfunktion $\alpha_{\mathbf{gr}}(X,Y,Z) = X$ in jedem Schritt ausschließlich unattackierte initiale Mengen ausgewählt. Das Verfahren endet, wenn der Argumentationsgraph, bzw. eines der entsprechenden Redukte keine unattackierten initialen Mengen enthält. Der Ausgangsgraph besitzt zwei unattackierte initiale Mengen, $is^{\leftarrow}(F_2) = \{\{a\}, \{c\}\}$. Als erstes wird $\{a\}$ gewählt und das Redukt $F_2^{\{a\}}$ gebildet (Abb. 8 (b)). Dieses besitzt nur noch die unattackierte initiale Menge $\{c\}$. Das folgende Redukt $F_2^{\{a,c\}}$ besitzt keine unattackierten initialen Mengen und somit ist die Terminationsbedingung erfüllt und es gilt $\mathbf{gr}(F_2) = \{a, c\}$.

Nachdem in diesem Kapitel die initialen Mengen definiert, sowie einige ihrer (strukturellen) Eigenschaften dargestellt wurden und mit dem Verfahren der Serialisierung von Extensionen eine wichtige Anwendung initialer Mengen gezeigt wurde, werden im folgenden Kapitel die in dieser Arbeit untersuchten Algorithmen zur Berechnung initialer Mengen vorgestellt.

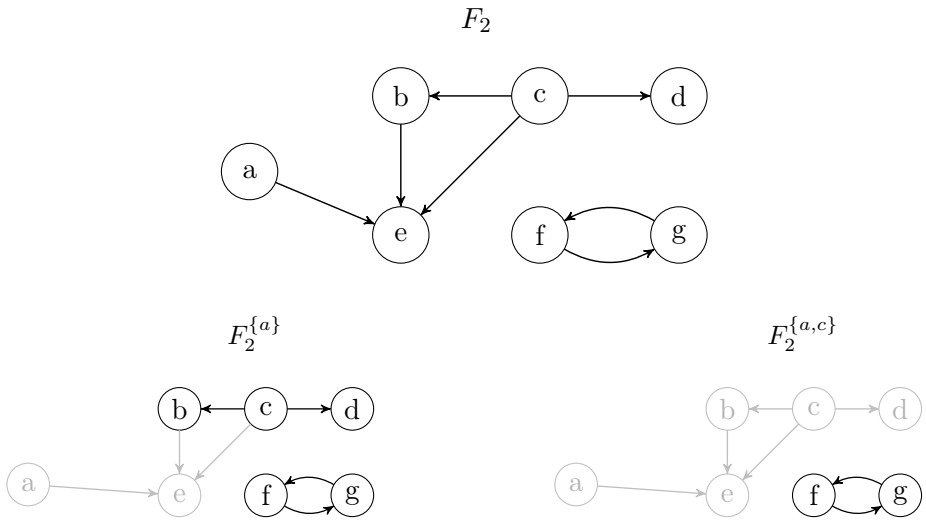

Abbildung 8: Serialisierung der grundierten Extension des Argumentationsgraphen F_2.

4 Algorithmen zur Berechnung initialer Mengen

In diesem Kapitel werden die untersuchten Algorithmen zur Berechnung initialer Mengen vorgestellt und erläutert. Zunächst werden in Kapitel 4.1 die rein SAT-basierten Ansätze vorgestellt und erläutert. Anschließend wird in Kapitel 4.2 der prozedurale Test auf Minimalität einer nicht-leeren, zulässigen Menge dargestellt.

4.1 SAT-Kodierungen für initiale Mengen

Ein häufig verwendeter Ansatz bei der Konstruktion von Extensionen ist die Reduktion eines Problems im Kontext abstrakter Argumentationsgraphen auf das *SAT-Problem*. In den folgenden zwei Unterabschnitten werden die in dieser Arbeit betrachteten Algorithmen zur Berechnung initialer Mengen auf Basis von SAT-Solvern vorgestellt. In Abschnitt 4.1.1 wird ein von Bengel und Thimm [6] entwickelter iterativer SAT-basierter Algorithmus vorgestellt. Anschließend werden in Abschnitt 4.1.2 zwei Modifikationen des iterativen SAT-Ansatzes präsentiert. Diese basieren auf der Zerlegung eines Argumentatationsgraphen in seine starken Zusammenhangskomponenten.

4.1.1 Iterative Berechnung initialer Mengen mittels SAT

Bengel und Thimm [6] haben, basierend auf Besnard et al. [7] und Niskanen und Jarvisalo [24], eine aussagenlogische Kodierung für initiale Mengen entwickelt. Hierbei werden die Argumente eines Argumentationsgraphen durch jeweils zwei Atome aussagenlogisch modelliert:

Definition 4.1. *Sei $F = (A, R)$ ein Argumentationsgraph. Für jedes Argument $a \in A$ enthält Σ_F jeweils zwei Atome I_a und O_a. Für die Atome in Σ_F wird vereinbart, dass $I_a = true$, wenn das Argument $a \in A$ akzeptiert wird. Für ein Atom O_a gilt, dass es true ist, wenn das Argument a abgelehnt wird, a also von einem bereits akzeptierten Argument angegriffen wird.*

Die Kodierungen nicht-leerer zulässiger Mengen bestehen aus 4 Komponenten (Gleichung (2) - (5)). Zunächst muss zugesichert werden, dass ein Argument nicht gleichzeitig akzeptiert und abgelehnt werden darf. Allerdings ist es durchaus möglich, dass für ein Argument nicht entschieden werden kann, ob es akzeptiert wird, oder nicht ($\neg I_a = true \wedge \neg O_a = true$). Darüberhinaus wird gefordert, dass ein Argument, das attackiert wird, entweder verworfen wird, oder die entsprechenden Angreifer verworfen werden:

$$\Psi_{rej} = \bigwedge_{a \in A} (\neg O_a \vee \neg I_a) \wedge \bigwedge_{a \in A} \bigwedge_{b \in a^-} (O_a \vee \neg I_b) \wedge \bigwedge_{a \in A} (\neg O_a \vee \bigvee_{c \in a^-} I_c) \tag{2}$$

Ausserdem muss zugesichert werden, dass die resultierenden Mengen sowohl konfliktfrei, als auch in der Lage sind sich selbst zu verteidigen, also zulässig sind.

$$\Psi_{cf} = \bigwedge_{a \in A} \bigwedge_{b \in a^-} \neg I_a \vee \neg I_b \tag{3}$$

$$\Psi_{adm} = \Psi_{rej} \wedge \Psi_{cf} \wedge \bigwedge_{a \in A} \bigwedge_{b \in a^-} \neg I_a \vee O_b \tag{4}$$

Abschließend wird eine Disjunktion [1] über den Elementen einer Menge gebildet, um so zu garantieren, dass die resultierenden Mengen nicht-leer sind:

$$\Psi_{\neg\emptyset,adm} = \Psi_{adm} \wedge \bigvee_{a \in A} I_a \tag{5}$$

Die resultierende aussagenlogische Formel (5) sichert zwar zu, dass die korrespondierenden Mengen von Argumenten nicht-leer sind, jedoch ist die Minimalität der zulässigen Mengen damit nicht garantiert. Um auch das Kriterium der Minimalität zu sichern, schlagen Bengel und Thimm [6] die *iterative* Anwendung eines SAT-Solvers vor. Dabei wird für eine nicht-leere Menge an Argumenten $S \subseteq A$, die mit einem entsprechenden Modell für (5) korrespondiert, iterativ geprüft, ob es eine echte nichtleere Teilmenge $S' \subset S$ gibt, dessen korrespondierende Interpretation ein Modell für (5) darstellt. Dieses Verfahren zur iterativen Minimierung der entsprechenden Modelle wird in Algorithmus 2 dargestellt. In der inneren **while**-Schleife (Zeilen 3-15) wird zunächst ein SAT-Aufruf gestartet. Die Funktion SOLVE() gibt einen boolschen Wert zurück, der angibt, ob ein Modell gefunden wurde oder nicht. Wurde ein Modell gefunden, werden alle Variablen $a \in \Sigma_F$, für die $I(a) = true$ gilt, dem SAT-Solver als Minimierungsklausel $\bigvee_{I(a)=true} \neg I_a$ übergeben (Zeile 12). Dadurch wird einerseits

[1] Die leere Disjunktion würde zu $false$ ausgewertet.

zugesichert, dass bei einem weiteren SAT-Aufruf nicht dasselbe Modell mehrfach ausgegeben wird und andererseits sichergestellt, dass bei den möglicherweise nachfolgend iterativen Aufrufen innerhalb der inneren **while**-Schleife nur Modelle ausgegeben werden, dessen korrespondierenden Argumentmengen kleiner sind. Gleichzeitig wird mittels temporärer Annahmen (ASSUME() Zeile 14) sichergestellt, dass keine Argumente, die nicht auch im Modell liegen, in die mögliche Lösung aufgenommen werden. Diese innere **while**-Schleife wird so lange durchlaufen, bis kein kleineres, nicht-leeres Modell gefunden werden kann. Abschließend werden die gefundenen Extensionen innerhalb der äußeren **while**-Schleife gesammelt und nach Ablauf des Algortihmus ausgegeben.

Algorithmus 2: MINIMIZEMODELS($F = (A, R)$)

```
   Eingabe F = (A, R)
   Ausgabe is(F)
   Ext ← ∅;
   minimizationClause ← ∅;
   SAT ← Solver(Ψ_{¬∅,adm});
 1 while true do
 2     foundExtension ← false;
 3     while true do
 4         if !SAT.SOLVE() then
 5             break;
 6         else
 7             foundExtension ← true;
 8             minimizationClause ← ∅;
 9             M ← SAT.GETMODEL();
10             foreach a ∈ A do
11                 if I_a ∈ M then
12                     minimizationClause.ADD(¬I_a);
13                 else
14                     SAT.ASSUME(¬I_a);
15             SAT.ADDCLAUSE(minimizationClause);
16     if foundExtension then
17         Ext.add(M);
18     else
19         break;
20 return Ext;
```

4.1.2 SCC-basierte Encodings

In diesem Abschnitt sollen, basierend auf dem Ansatz von Bengel und Thimm [6], zwei alternative SAT-Kodierungen vorgeschlagen werden, welche die Suche initialer Mengen auf die SCC's eines Argumentationsgraphen beschränken. Wie in Abschnitt 3.1 dargelegt, lassen sich initiale Mengen in den starken Zusammenhangskomponenten eines Argumentationsgraphen lokalisieren. Um den Suchraum und die Größe von Kandidatenmengen zu reduzieren, werden ausschließlich die Argumente innerhalb einer starken Zusammenhangskomponente für die SAT-Kodierung betrachtet.

Iterative Berechnung Für die Kodierung der starken Zusammenhangskomponenten eines Argumentationsgraphen wird die Gleichung (5) entsprechend erweitert. Sei dazu $F = (A, R)$ ein abstrakter Argumentationsgraph und $A^{'} \subseteq A$ mit $a \in A^{'}$ und $F|_{A'} \in SCC(F)$ die Projektion auf die starke Zusammenhangskomponente, die a enthält. Mit $SCC_F(a) = \{b \mid b \in A^{'}\}$, wird die die Menge aller Argumente bezeichnet, die sich in der starken Zusammenhangskomponenten in F befinden, in der sich auch a befindet.

$$\Psi_{\neg\emptyset,adm,SCC} = \Psi_{\neg\emptyset,adm} \wedge \bigwedge_{a \in A} \bigwedge_{b \in A \setminus SCC_F(a)} (\neg I_a \vee \neg I_b) \tag{6}$$

Für die Kodierung der starken Zusammenhangskomponenten werden diese zunächst mithilfe des Algorithmus von Tarjan [28] berechnet. Dieser Algorithmus berechnet die starken Zusammenhangskomponenten in linearer Laufzeit $\mathcal{O}(|A| + |R|)$.

Gleichung (6) blockiert die Akzeptanz zweier Argumente, die sich nicht in derselben starken Zusammenhangskomponente befinden. Gleichung (6) garantiert ebenfalls noch keine Minimalität der resultierenden Argumentmengen. Wie bereits angedeutet, muss dazu iterativ nach nicht-leeren, echten Teilmengen einer Kandidatenmenge gesucht werden (siehe Algorithmus 2).

Nicht-iterative Berechnung Neben dem iterativen Ansatz zur Berechnung initialer Mengen ist es außerdem möglich, als nicht-iterativ zu testen, ob eine Kandidatenmenge minimal ist, oder nicht. Der Vorteil an diesem Vorgehen wäre, dass dadurch die Anzahl der SAT-Aufrufe begrenzt werden.

Im Folgenden wird ein Ansatz beschrieben, mit dem die Anzahl an iterativen SAT-Aufrufen auf zwei Aufrufe pro Kandidatenmenge begrenzt wird. So muss der SAT-Solver nicht mehr iterativ die gefundenen Modelle minimieren, sondern pro gefundener Kandidatenmenge wird lediglich ein weiterer Aufruf genügen, um zu entscheiden, ob diese Kandidatenmenge minimal ist, oder nicht.

Ausgehend von der SAT-Kodierung für nicht-leere, zulässige Mengen $\Psi_{\neg\emptyset,adm,SCC}$ (6), wird eine zweiteilige SAT-Kodierung für *minimale*, nicht-leere zulässige Mengen vorgeschlagen.

Zunächst werden die Beschränkungen der Kandidatenmengen auf nicht-leere, zulässige Mengen innerhalb einer starken Zusammenhangskomponente mithilfe von Gleichung (6) $\Psi_{\neg\emptyset,adm,SCC}$ kodiert. In einem ersten SAT-Aufruf wird nun mittels (6) eine

nicht-leere, zulässige Kandidatenmenge S innerhalb eines SCC in F berechnet.

$$\Psi_{S,minTest} = \Psi^{S}_{\neg\emptyset,adm} \tag{7}$$

Eine Kandidatenmenge S ist dann eine initiale Menge, wenn $\mathbf{Mod}_{\Sigma_F}(\Psi_{S,minTest}) = \emptyset$, also kein Modell für Gleichung (7) existiert. Dazu wird die Kodierung nicht-leerer zulässiger Mengen $\Psi_{\neg\emptyset,adm}$ (5) auf eine Teilmenge S angewendet, statt auf die komplette Menge an Argumenten A, was mit $\Psi^{S}_{\neg\emptyset,adm}$ bezeichnet wird. Außerdem kann in Gleichung (7) auf die Kodierung der Beschränkung auf starke Zusammenhangskomponenten verzichtet werden, da durch Gleichung (6) sichergestellt wird, dass eine Kandidatenmenge S ausschließlich Argumente innerhalb der selben Zusammenhangskomponente enthält.

Algorithmus 3: NON-ITERATIVE SAT($F = (A, R)$)

```
   Eingabe F = (A, R)
   Ausgabe is(F)
   Ext ← ∅;
   complementClause ← ∅;
   SAT_Candidate ← Solver(Ψ_{¬∅,adm,SCC});
1  while true do
2      if !SAT_Candidate.SOLVE() then
3          break;
4      else
           SAT_min ← Solver(Ψ_{¬∅,adm});
5          S ← SAT_Candidate.GETMODEL();
6          complementClause ← S;
7          foreach a ∈ A do
8              if I_a ∈ M then
9                  minimizationClause.ADD(¬I_a);
10             else
11                 SAT.ASSUME(¬I_a);
12         SAT_Candidate.ADDCLAUSE(minimizationClause);
13         if !SAT_min.SOLVE() then
14             Ext.add(S);
15 return Ext;
```

Das Verfahren zur Berechnung initialer Mengen mittels des nicht-iterativen Tests wird in Algorithmus 3 dargestellt. Ein erster SAT-Solver ($SAT_{Candidate}$) berechnet zunächst eine nicht-leere, zulässige Menge innerhalb eines starken Zusammenhangskomponente im Argumentationsgraphen $F = (A, R)$. Wird eine solche Kandidaten-

menge S gefunden, wird die mittels Gleichung (7) ein zweiter SAT-Solver (SAT_{min}) aufgerufen, welcher prüft, ob die Menge S minimal ist. Die Beschränkung auf Argumente innerhalb von S wird mittels temporärer Annahmen (Zeile 11) sichergestellt.

4.2 Prozeduraler Test auf Minimalität einer zulässigen Menge

Neben dem rein reduktionsbasierten Ansatz zur Berechnung initialer Mengen mittels iterativer Aufrufe eines SAT-Solvers, wird nun ein direkter Test auf Minimalität nicht-leerer, zulässiger Mengen vorgestellt. Dieser Test wurde von Thimm [29] vorgeschlagen.

Die Frage, ob eine bestimmte konfliktfreie Menge an Argumenten eines Argumentationsgraphen eine initiale Menge ist, lässt sich als Entscheidungsproblem formulieren.

VER_{IS}	
Eingabe:	Ein abstrakter Argumentationsgraph $F = (A, R)$, $S \subseteq A$ und $S \in cf(F)$.
Frage:	Ist S eine initiale Menge?

In Thimm [29] wurde gezeigt, dass die folgende Proposition gilt:

Proposition 4.1. *$VER_{IS} \in \mathcal{P}$.*

Somit lässt sich in polynomieller Zeit entscheiden, ob eine konfliktfreie Menge von Argumenten eine initiale Menge ist. Der Beweis zu Proposition 4.1 ist konstruktiv und beschreibt ein Testverfahren, mit dessen Hilfe sich für eine konfliktfreie Menge ermitteln lässt, ob diese eine initiale Menge ist.

Das Testverfahren auf Minimalität einer zulässigen Menge macht sich zunutze, dass in polynomieller Zeit entschieden werden kann, ob eine Menge an Argumenten eine zulässige Menge enthält:

$\text{CONTAINS}_{\text{ADM}}$	
Eingabe:	Ein abstrakter Argumentationsgraph $F = (A, R)$, eine konfliktfreie Menge an Argumenten $S \subseteq A$ und ein Argument $a \in S$.
Frage:	Enthält S eine zulässige Menge $S' \subseteq S$, so dass $a \in S'$ gilt?

Lemma 4.1. *$\text{CONTAINS}_{ADM} \in \mathcal{P}$ (Lemma 1 in [29]).*

Der Beweis zu Lemma 4.1 ist ebenfalls konstruktiv, liefert also eine Konstruktionsvorschrift für eine zulässige Menge $S' \subseteq S$ mit $a \in S$, sofern diese existiert. Dazu wird zunächst geprüft, ob S bereits zulässig ist. Dies lässt sich mithilfe der *charakteristischen Funktion* testen [15]. Die Menge $\Delta_F(S)$ lässt sich konstruieren, indem für jedes Argument $a \in A$ geprüft wird, ob es ein Argument $b \in A$ gibt, welches a angreift. Ist dies der Fall wird geprüft, ob es ein Argument $c \in S$ gibt, welches wiederum das Argument b attackiert, also a verteidigt. Im *worst case* besitzt der Algorithmus also

Algorithmus 4: CHARAKTERISTISCHE FUNKTION $\Delta_F(S)$

```
   Eingabe F = (A, R), S ⊆ A
   Ausgabe Δ_F(S)
   Δ_F(S) ← ∅;
   attacked ← false;
 1 foreach a ∈ A do
 2     foreach b ∈ A do
 3         if (b, a) ∈ R then
 4             attacked ← true;
 5             foreach c ∈ S do
 6                 if (c, b) ∈ R then
 7                     Δ_F(S).add(a) ;
 8                     attacked ← false;
 9                     break;
10     if attacked == false then
           Δ_F(S).add(a) ;
11     attacked ← false;
12 return Δ_F(S);
```

eine Laufzeit in $\mathcal{O}(|A| * |A| * |S|) = \mathcal{O}(|A^3|)$ [30]. Die Berechnungsvorschrift für die charakteristische Funktion ist in Algorithmus 4 detailliert dargestellt.

Aus der Definition der Zulässigkeit folgt, dass eine konfliktfreie Menge S zulässig ist, wenn $S \subseteq \Delta_F(S)$ gilt. Für die Prüfung der Teilmengenrelation $S \subseteq \Delta_F(S)$ ist es lediglich notwendig, bei der Berechnung der charakteristischen Funktion $\Delta_F(S)$ nur die Argumente in S selbst zu betrachten, anstatt für jedes Argument $a \in A$ zu prüfen, ob $a \in \Delta_F(S)$. Dadurch lässt sich insb. bei sehr großen Argumentationsgraphen die Laufzeit für die Berechnung der charakteristischen Funktion im Durchschnitt minimieren.

In Abhängigkeit davon, ob S bereits zulässig ist, oder nicht, müssen zwei Fälle unterschieden werden. Angenommen, es gelte $S \subseteq \Delta_F(S)$, dann terminiert das Verfahren, mit dem Ergebnis, dass S eine zulässige Menge (sich selbst) enthält, mit $a \in S$. Ist S nicht zulässig, wird die Schnittmenge aus S und der Menge aller Argumente in A, die von S verteidigt werden, also der charakteristischen Funktion von S, gebildet: $S_1 = \Delta_F(S) \cap S$. Die Menge S_1 enthält alle Argumente, die in S liegen und auch von S verteidigt werden. Ist das Argument a nicht in S_1 enthalten, dann kann es keine zulässige Menge $S' \subseteq S$ geben, die a enthält.

Enthält S_1 das Argument a jedoch, dann müssen zwei Fälle unterschieden werden:

1. S_1 ist zulässig, S enthält also eine Menge die sowohl a enthält und zulässig ist. Das Verfahren endet.

2. S_1 enthält a, ist jedoch nicht zulässig [2]: $S_1 \not\subseteq \Delta_F(S_1)$.

Im 2. Fall wird $S_2 = \Delta_F(S_1) \cap S_1$ gebildet und die obigen Schritte wiederholt. Dieses Verfahren endet im *worst case* nach maximal $|S|$ Wiederholungen. Dieses Verfahren ist zusammenfassend in Algorithmus 5 ($\textsc{Contains}_{\text{ADM}}$(S)) dargestellt. Mittels $\textsc{Contains}_{\text{ADM}}$(S) lässt sich nun ein Testverfahren bilden, welches für eine konfliktfreie Menge entscheidet, ob diese eine nicht-leere und minimale, zulässige Menge, also eine initiale Menge ist (Algorithmus 6, IsInital(S)). Dazu wird für jedes Paar von Argumenten $a, b \in S$ mit $\textsc{Contains}_{\text{ADM}}$(S) geprüft, ob die Menge $S \setminus \{b\}$ eine zulässige Menge enthält. Falls ein solche Menge existiert, kann S keine initiale Menge sein. Andersherum ist S also eine initiale Menge, wenn sich keine solche Menge finden lässt [29].

Algorithmus 5: $\textsc{Contains}_{\text{ADM}}(S, a)$

Eingabe $F = (A, R), S \in cf(F), a \in A$
Ausgabe *true, falls* S *eine zulässige Menge* $S' \subseteq S$ *mit* $a \in S'$ *enthält,* $false$ *sonst.*

```
1 if a ∉ S then
2     return false;
3 C = Δ_F(S);
4 if S ⊆ C then
5     return true;
6 else
7     S_k = Δ_F(S) ∩ S;
8     return Contains_ADM(S_k, a);
```

Zur Veranschaulichung des Verfahrens wird der Argumentationsgraph $F_6 = (A_2, R_2)$ in Abb. 9 betrachtet.

Beispiel 4.1

Angenommen, es soll geprüft werden, ob die konfliktfreie Menge $S_T = \{a, c, d, e\}$ eine initiale Menge ist. Zunächst wird geprüft, ob $S_{T_0} = S_T \setminus \{e\} = \{a, c, d\}$ eine zulässige Menge enthält, in der das Argument a enthalten ist. Da das Argument a in S_{T_0} enthalten ist, wird die charakteristische Funktion $\Delta_F(S_{T_0})$ berechnet (Zeile 3, Algorithmus 5). Mittels Algorithmus 4 ergibt sich $\Delta_F(S_{T_0}) = \{a, d, e\}$. Damit ist die Menge S_{T_0} nicht zulässig, da $S_{T_0} \not\subseteq \Delta_F(S_{T_0})$ gilt (Zeile 4, Algorithmus 5). Die Menge ist also nicht in der Lage alle ihre Argumente zu verteidigen (c wird nicht verteidigt). Dementsprechend wird anschließend geprüft, ob die Menge $S_{T_1} = S_{T_0} \cap \Delta_F(S_{T_0}) = \{a, d\}$ eine zulässige Menge enthält, in der a enthalten ist (Zeilen 7,8 in Algorithmus 5). Die Prozedur wiederholt sich, es wird geprüft, ob $a \in S_{T_1}$ gilt und S_{T_1} zulässig

[2] Dieser Fall tritt beispielsweise ein, wenn ein Argument $b \in S_1$ von einem Argument c aus der Menge $S \setminus \Delta_F(S)$ verteidigt wird.

Algorithmus 6: IsInital(S)

Eingabe $F = (A, R), S \in cf(F), S \neq \emptyset$
Ausgabe *true, falls* $S \in is(F)$*,* $false$ *sonst.*

```
1  if |S| = 1 then
2      if S ⊆ Δ_F(S) then
3          return true;
4      else
5          return false;
6  foreach a ∈ S do
7      foreach b ∈ S, b ≠ a do
8          if Contains_ADM(S \ {b}, a) then
9              return false;
10 return true;
```

ist. Da $a \in S_{T_1}$ und $\Delta_F(S_{T_1}) = \{a, d, e\} \supseteq S_{T_1}$, gibt $\text{Contains}_{\text{ADM}}$ *true* aus. Somit terminiert auch IsInital (Algorithmus 6, Zeilen 3-4) und gibt $false$ aus, da mit $S_{T_1} \neq \emptyset$ eine echte Teilmenge von S_T gefunden wurde, die ebenfalls zulässig ist, S_T kann somit keine initiale Menge sein.

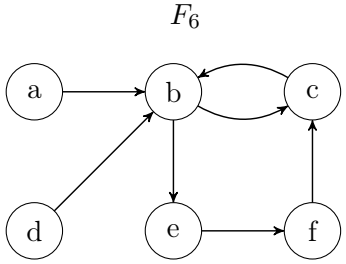

Abbildung 9: Der Argumentationsgraph F_6.

Im *worst case* beträgt die Rekursionstiefe von $\text{Contains}_{\text{ADM}}(S, a)$ $|S|$ Ebenen. In jeder dieser Rekursionsebenen wird potenziell die charakteristische Funktion $\Delta_F(S)$ berechnet. Somit gilt $\text{Contains}_{\text{ADM}}(S, a) \in \mathcal{O}(|S| * |A^3|) = \mathcal{O}(|A^4|)$.

In der Prozedur IsInital(S) wird paarweise über die Argumente in S iteriert, wobei in jeder Iteration ein Aufruf der Prozedur $\text{Contains}_{\text{ADM}}(S, a)$ erfolgt. Dies führt im *worst case* zu einem Laufzeitverhalten von IsInital(S)$\in \mathcal{O}(|S^2| * |A^4|) = \mathcal{O}(|A^6|)$. Für eine effiziente Implementierung dieses Testverfahrens sind also weitere Optimierungen notwendig. Die in dieser Arbeit gewählten Ansätze zur Optimierung zielen insbesondere auf die Reduzierung der Anzahl möglicher Kandidatenmengen,

sowie dessen Größe.

Optimierungsansätze Ausgangspunkt dieses Testverfahrens sind die Mengen eines Argumentationsgraphen, die mindestens das Kriterium der Konfliktfreiheit erfüllen. Um die konfliktfreien Mengen eines Argumentationsgraphen zu berechnen gibt es grundsätzlich mehrere Ansätze. Ein naives Verfahren bestünde beispielsweise darin, für einen Argumentationsgraphen $F = (A, R)$ die Potenzmenge der Argumente $2^A = \{S | S \subseteq A\}$ zu betrachten und jede Teilmenge auf Konfliktfreiheit zu untersuchen. Dies wäre jedoch sowohl laufzeit- als auch speicherplatztechnisch sehr ineffizient und würde das eigentliche Testverfahren sehr erheblich verlangsamen. Ein weitaus effizienterer Ansatz stellt der Einsatz eines SAT-Solvers dar. Darüberhinaus werden mittels SAT-Solver nicht-leere *zulässige* Mengen berechnet. Diese sind per Definition konfliktfrei und i.d.R. enthält ein Argumentationsgraph weniger zulässige als konfliktfreie Mengen. Dadurch wird der Suchraum für den eigentlichen Test eingeschränkt. Zusätzlich wird eine weitere Einschränkung des Suchraumes auf *starke Zusammenhangskomponenten* eines Graphen vorgenommen. Dies kann mittels der Gleichung 6 kodiert werden.

Damit lässt sich das Testverfahren nun folgendermaßen zusammenfassen: Zunächst werden die starken Zusammenhangskomponenten eines Argumentationsgraphen konstruiert. Darauffolgend werden innerhalb der starken Zusammenhangskomponenten mittels SAT-Solver nach nicht-leeren, zulässigen Mengen gesucht. Anschließend wird das eigentliche Testverfahren auf Minimalität der so gefundenen nicht-leeren, zulässigen Mengen durchgeführt.

Nachdem in diesem Kapitel die in dieser Arbeit betrachteten, algorithmischen Ansätze zur Berechnung initialer Mengen dargestellt wurden, wird im folgenden Kapitel die im Rahmen dieser Arbeit umgesetzte Implementierung kompakt dargestellt.

5 Implementierung

In diesem Kapitel wird die Implementierung der in Kapitel 4 dargestellten Verfahren zur Berechnung initialer Mengen beschrieben. Insbesondere wird die grundelegende Schnittstelle zur Nutzung des InitialSetSolvers dargestellt, sowie das Inputformat für die Instanzen der Argumentationsgraphen erläutert.

Die verschiedenen Algorithmen wurden in C++ implementiert. Die Implementierung (**InitialSetSolver**) ist quelloffen und via GitHub verfügbar [3]. Als Referenzimplementierung dient der *Serialisation-Solver* [6], grundlegende Datenstrukturen, die Implementierung des Algorithmus 2 und die Kodierung nicht-leerer, zulässiger Mengen wurden für den InitialSetSolver übernommen bzw. modifiziert. Als SAT-Solver wird der **CaDiCaL**-SAT-Solver von Biere et al. [8] in der Version 2.1.3 genutzt. Für die Berechnung der starken Zusammenhangskomponenten wurde die **Boost-Graph-Library** [27] verwendet.

Inputformat Der InitialSetSolver nutzt das `i23`-Format [19] für die kompakte Repräsentation von abstrakten Argumentatationsgraphen. Die Argumente eines abstrakten Argumentationsgraphen werden dabei als fortlaufender Index ganzzahliger Werte von 1 - n repräsentiert, wobei n die Anzahl der Argumente repräsentiert, wobei am Beginn der Datei die Anzahl der Argumente mithilfe einer **p-line** initialisiert wird:

```
p af <n>
```

Ein Angriff zwischen zwei Argumenten wird als Tupel der Indizes der beiden Argumente dargestellt. Zur Verdeutlichung wird im Folgenden der Argumentationsgraph 1 aus Abschnitt 2.1 erneut betrachtet.

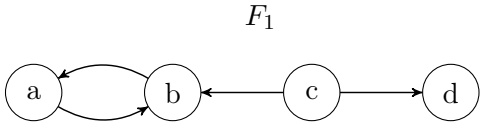

Abbildung 10: Der Argumentationsgraph F_1.

Die `i23`-Repräsentation des Argumentatiosngraphen F_1 würde folgendermaßen aussehen:

```
p af 4
1 2
2 1
3 2
3 4
```

[3] `https://github.com/ToyotaJochen/InitialSetSolver`

ICCMA Solver-Interface Der InitialSetSolver implementiert das ICCMA Solver-Interface [21]. Das Aufzählen aller Extensionen, einer bestimmten Semantik eines Argumentationsgraphen wird im ICCMA Solver-Interface als Problemklasse **EE (Enumerate Extensions)** bezeichnet. Die relevanten Parameter zur Bedienung des InitialSetSolvers über eine Kommandozeile werden in der folgenden Tabelle zusammengefasst:

Befehl	**Parameter**	**Erläuterungen**
`./InitialSetSolver`	keine	Ausgabe von Informationen zur Version.
	-p <problem>	Enumerate Extensions: • **EE-IT** - Iterative-SAT • **EE-ITSCC** - Iterative-SCC-SAT • **EE-NIT** - Non-Iterative-SCC-SAT • **EE-PROC** - Procedural
	-f <file>	Dateipfad der Instanz
	-h	Ausgabe des Hilfe-Textes

Tabelle 2: Interface des InitialSetSolvers

Ausgabe Die gefundenen initialen Mengen eines abstrakten Argumentationsgrapghen werden im Folgenden Format auf der Konsolenausgabe ausgegeben. Jede initiale Menge wird dabei als Tupel der entsprechenden Indizes der Argumente in eckigen Klammern dargestellt. Beispielhaft wird die Ausgabe für den Argumentatationsgraphen F_3 dargestellt:

```
[2, 7],[8],[3,6],[1]
```

Nachdem in diesem Kapitel die Bedienung und Ein- und Ausgabe des InitialSetSolvers in kompakter Form dargestellt wurden, werden im nächsten Kapitel die durchgeführten Experimente und Ergebnisse der empirischen Auswertung zu der Performanz der unterschiedlichen Algorithmen zur Berechnung initialer Mengen in abstrakten Argumentationsgraphen vorgestellt.

6 Evaluation

In diesem Kapitel sollen zunächst die Benchmark-Instanzen aus dem ICCMA 23 Datensatz vorgestellt und anhand einiger deskriptiver Statistiken dargestellt werden. Anschließend werden die durchgeführten Experimente erläutert und die Ergebnisse dargestellt und diskutiert.

6.1 Benchmark-Datensatz

Die Performanz der in Kap. 4 dargestellten Verfahren zur Berechnung initialer Mengen soll mittels des 2023er Datensatzes der *International Competition on Computational Models of Argumentation* [4] *(ICCMA)* [19] vergleichend evaluiert werden. Der ICCMA 2023 Benchmark-Datensatz [18] besteht aus insgesamt 329 Instanzen von abstrakten Argumentationsgraphen. Diese stammen aus 14 Kategorien, beispielsweise gibt es Kategorien für Argumentationsgraphen basierend auf verschiedenen Graphenmodellen. Dazu zählen etwa die Modelle von Barabasi und Albert für skalenfreie Netzwerke [1], das Modell von Watts und Strogatz für sog. *Small-World* Netzwerke [31], oder auch Erdös-Renyi Zufallsgraphen, bei dem Graphen anhand einer Wahrscheinlichkeit p für das Vorhandensein einer Kante zwischen zwei zufällig ausgewählten Knoten, erzeugt werden [25]. Neben unterschiedlichen Graphenmodellen gibt es unterschiedliche Kategorien von Instanzen, welche mittels spezieller Graphengeneratoren erzeugt wurden. In den Kategorien *StableGenerator* und *GroundedGenerator* beispielsweise, besitzen die erzeugten Instanzen jeweils eine hohe Anzahl stabiler Extensionen, bzw. eine sehr große grundierte Extension [19, 12].

Eine Übersicht über die verschiedenen Kategorien des ICCMA 2023 Datensatzes ist in Tabelle 3 abgebildet. Für jede Kategorie sind neben einer Kurzbeschreibung der Eigenschaften der jeweiligen Instanzen, die durchschnittliche Anzahl an Argumenten, die durchschnittliche Anzahl an Attacken, sowie die durchschnittliche Anzahl an starken Zusammenhangskomponenten angegeben.

[4] `https://argumentationcompetition.org`

Graphenkategorie	Eigenschaften	$\overline{\lvert A \rvert}$	$\overline{\lvert R \rvert}$	$\overline{\lvert SCC_F \rvert}$
ABA2AF[23]	Abstrakte Argumentationsgraphen, welche auf Grundlage einer regelbasierten Wissensbasis, ausgehend von Annahmen und einer dazugehörigen logischen Theorie, generiert werden.	434.760	312129.8	15.92
AdmBuster[9]	Argumentationsgraphen mit Fokus auf starker Zulässigkeit.	526733.3	790099	526733.33
AFgen[17]	Argumentationsgraphen auf Basis eines Zufallsgraphenmodells, welches auf bestimmte Komplexitätscharakteristiken abstrakter Argumentationsgraphen fokussiert ist.	230.882	18567.235	1
Barabasi-Albert[11]	Skalenfreie Netzwerke	141	203.762	87.48
crustiG2io[22]	Hohe Anzahl verschachtelter Graphen (inner-outer Graph Model)	46542.188	4400634.562	217
Datalog[34]	Argumentationsgraphen basierend auf Datalog Wissensbasen.	3122.88	5175286.44	1073.44
Erdös-Rényi[11]	Erdös-Renyi Zufallsgraphen	289.12	26571.160	1
GroundedGenerator[12]	Graphen mit sehr großen grundierten Extensionen.	2301.72	81461.84	343
Planning2AF	Argumentationsgraphen, welche auf Basis aussagenlogischer Formeln generiert wurden.	816.16	1571.88	646.32
SCCGenerator[12]	Hohe Anzahl an SCC	3654.84	556483.52	41.32
SemBuster[10]	Graphen mit hoher Anzahl an semi-stabilen Extensionen.	2890	1469724.3	1926.66
StableGenerator[12]	Graphen mit hoher Anzahl an stabilen Extensionen	973.8	29752.6	24.16
Traffic	reale Verkehrsnetze	2534.04	5326.08	569.8
Watts-Strogatz[11]	Small-World Graphen	268	2397	2.36

Tabelle 3: Deskriptive Statistiken zu den Graphenkategorien des ICCMA 23 Datensatzes, eigene Berechnungen.

6.2 Experimente

Um die Laufzeiten der verschiedenen algorithmischen Ansätze zur Berechnung initialer Mengen in abstrakten Argumentationsgraphen zu vergleichen, wird für jede Instanz des ICCMA 2023 Datensatz die Menge aller initialen Mengen brechnet. Das ICCMA Solver Interface bezeichnet diese Problemklasse mit **Enumerate-Extensions (EE)**.

Als Referenzimplementierung dient der *Serialisation-Solver* [6], welcher den iterativen SAT-Ansatz **EE-IT** implementiert. Die weiteren Ansätze werden mittels des im vorigen Abschnittes vorgestellten *InitialSetSolvers* getestet. Die Berechnung initialer Mengen mittels des iterativen SAT-Ansatzes auf Grundlage der starken Zusammenhangskomponenten, dem nicht-iterativen SAT-Ansatz auf Basis der starken Zusammenhangskomponenten bzw. mittels des prozeduralen Testverfahrens werden im Folgenden mit **EE-ITSCC**, **EE-NITSCC**, bzw. **EE-PROC** bezeichnet.

Die Experimente wurden auf einem Server der Fernuniversität Hagen durchgeführt. Es handelt sich um ein System mit einer Intel CPU (Haswell) mit 8 Kernen (3,4 GHz), 48 GB RAM und Ubuntu 20.04.6 (LTS) als Betriebssystem. Die Zeitbegrenzung für die Berechnung wurde auf 400 Sekunden pro Instanz gesetzt. Für die Durchführung der Benchmarks wird das Python-Tool *probo2* [20] verwendet. Die Ausgaben des *InitialSetSolvers* wurden anhand der Ausgaben des *Serialisation-Solvers* auf Korrektheit geprüft. Die Rohddaten der Experimente sind ebenfalls via GitHub [5] verfügbar.

6.3 Ergebnisse

In diesem Abschnitt werden die Ergebnisse der bezüglich der Laufzeiten der unterschiedlichen Algorithmen zur Berechnung initialer Mengen in abstrakten Argumentationsgraphen vorgestellt und diskutiert.

In der Abbildung 11 sind die Laufzeiten der unterschiedlichen Algorithmen in Abhängigkeit der Anzahl gelöster Instanzen auf dem gesamten ICCMA 2023 Datensatz dargestellt. In den Abbildungen 12 und 13 werden die Laufzeiten für den kompletten ICCMA 2023 Datensatz in Abhängigkeit der Anzahl an Attacken bzw. der Anzahl der Argumente der jeweiligen Instanzen dargestellt. In der Tabelle 4 werden die Anzahl an Timeouts, differenziert nach den Graphenkategorien des ICCMA 2023 Datensatzes, dargestellt. Außerdem werden in den Abbildungen 14 und 15 die Laufzeiten der verschiedenen Algorithmen, differenziert nach den unterschiedlichen Graphenmodellen bzw. Graphengeneratoren innerhalb des ICCMA 2023 Datensatzes dargestellt.

Bevor in den folgenden Unterabschnitten auf die Ergebnisse mit Fokus auf die einzelnen Algorithmen eingegangen wird, soll zunächst allgemein auf die Ergebnisse eingegangen und innerhalb der unterschiedlichen Graphenkategorien insb. die herausfordernsten Kategorien insgesamt identifiziert werden.

[5] `https://github.com/ToyotaJochen/InitialSetSolver/tree/main/results`

6.3.1 Überblick

Grundsätzlich lässt sich kein (linearer) Zusammenhang zwischen den Laufzeiten und der Anzahl der Attacken, bzw. der Anzahl der Argumente eines abstrakten Argumentationsgraphen identifizieren (s. Abbildungen 12 und 13). Basierend auf den Timeouts (s. Tabelle 4) stellen die Instanzen aus den Kategorien AdmBuster, crustiG2io, Datalog und StableGenerator die herausfordernsten Graphenkategorien dar. In der Kategorie AdmBuster konnten nur ca. die Hälfte aller Instanzen innerhalb der Zeitbegrenzung gelöst werden. In der Kategorie Datalog liegt der Anteil gelöster Instanzen zwischen 0% (**EE-NITSCC**) und 60% (**EE-IT**). Mit Ausnahme des **EE-IT**-Ansatzes konnten in der Kategorie crustiG2io lediglich ca. 20% der Instanzen gelöst werden. Bei der Kategorie StableGenerator fällt auf, dass über alle Ansätze hinweg lediglich 52% der Instanzen gelöst werden können, die Laufzeiten bei den Instanzen die gelöst werden konnten, jedoch relativ niedrig sind (s. Abbildung 15). Ein ähnliches Bild bezüglich der Laufzeiten ergibt sich für die gelösten Instanzen der Kategorie AdmBuster und Datalog (mit Ausnahme sehr weniger Instanzen, s. Abbildung 14).

In den Kategorien ABA2AF und SCCGenerator konnten alle Instanzen über alle Ansätze hinweg innerhalb des Zeitlimits gelöst werden. Auch in den Kategorien GroundedGenerator und Planning2AF gab es mit Ausname jew. eines Ansatzes keine Timeouts. Ausserdem weisen die Kategorien Watts-Strogatz, AFgen und SemBuster (mit Ausnahme von **EE-NITSCC**) insgesamt relativ geringe Anzahlen an Timeouts auf.

Bezüglich der Laufzeiten lässt sich generell sagen, dass für den Großteil aller Instanzen die Laufzeiten aller Ansätze relativ ähnlich sind (s. Abbildung 11). Die Unterschiede in den Laufzeiten zwischen den Ansätzen lassen sich eher nachvollziehen, bei der Betrachtung der Ergebnisse in den einzelnen Graphenkategorien. Diese Unterschiede werden in den folgenden Unterabschnitten, jeweils mit Fokus auf die unterschiedlichen algorithmischen Ansätze, dargestellt.

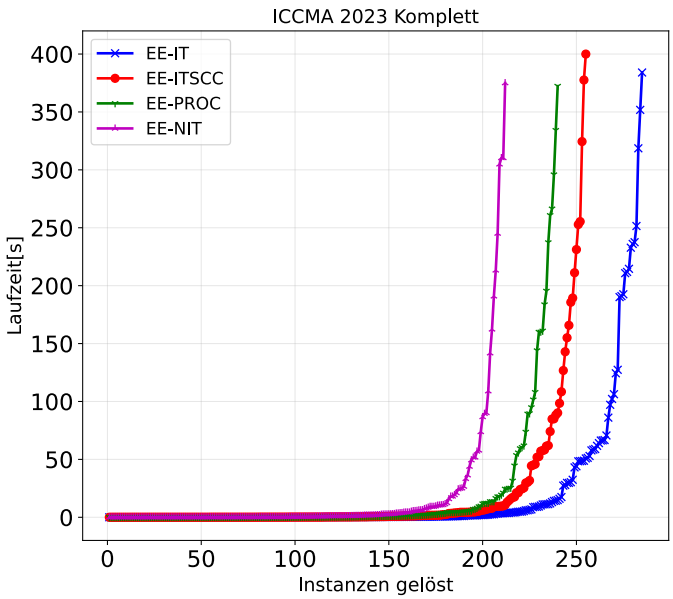

Abbildung 11: Laufzeiten der verschiedenen Algorithmen zur Berechnung initialer Mengen auf dem kompletten ICCMA 23 Datensatz.

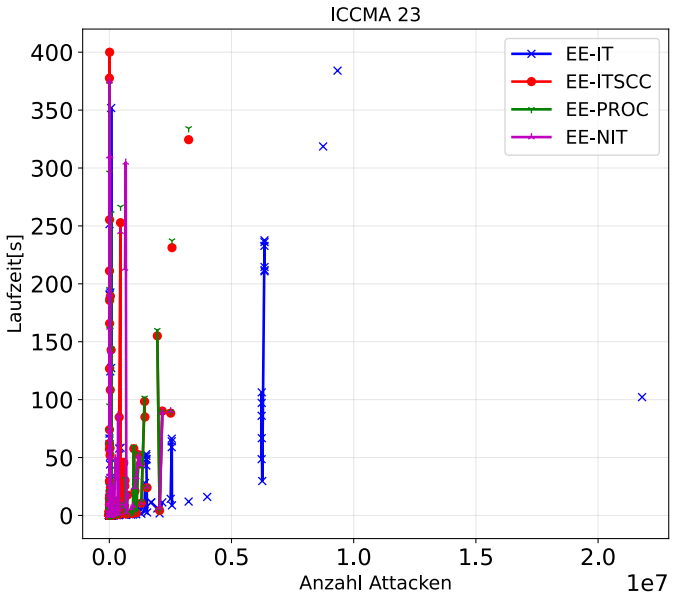

Abbildung 12: Laufzeitverhalten in Abhängigkeit der Anzahl an Attacken.

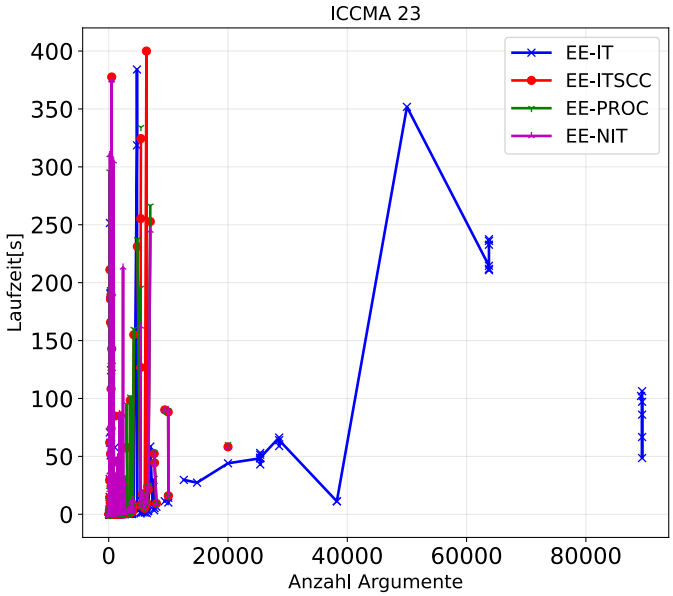

Abbildung 13: Laufzeitverhalten in Abhängigkeit der Anzahl an Argumenten.

Graphenkategorie	**EE-IT**	**EE-ITSCC**	**EE-PROC**	**EE-NITSCC**
ABA2AF	0(0)	0(0)	0(0)	0(0)
AdmBuster	7(0.46)	7(0.46)	8(0.53)	7(0.46)
AFgen	2(0.12)	2(0.12)	5(0.29)	2(0.12)
Barabasi-Albert	0(0)	0(0)	0(0)	10(0.4)
crustiG2io	3(0.09)	26(0.81)	26(0.81)	26(0.81)
Datalog	10(0.4)	12(0.48)	22(0.88)	25(1)
Erdös-Rényi	5(0.2)	5(0.2)	5(0.2)	10(0.4)
GroundedGenerator	0(0)	0(0)	5(0.2)	0(0)
Planning2AF	0(0)	0(0)	0(0)	2(0.08)
SCCGenerator	0(0)	0(0)	0(0)	0(0)
SemBuster	0(0)	2(0.13)	2(0.13)	7(0.47)
StableGenerator	13(0.52)	13(0.52)	13(0.52)	13(0.52)
Traffic	0(0)	3(0.12)	5(0.2)	11(0.44)
Watts-Strogatz	4(0.16)	3(0.12)	3(0.12)	3(0.12)
Σ	44(0.13)	73(0.22)	94(0.29)	116(0.35)

Tabelle 4: Anzahl an Timeouts, relative Werte in Klammern.

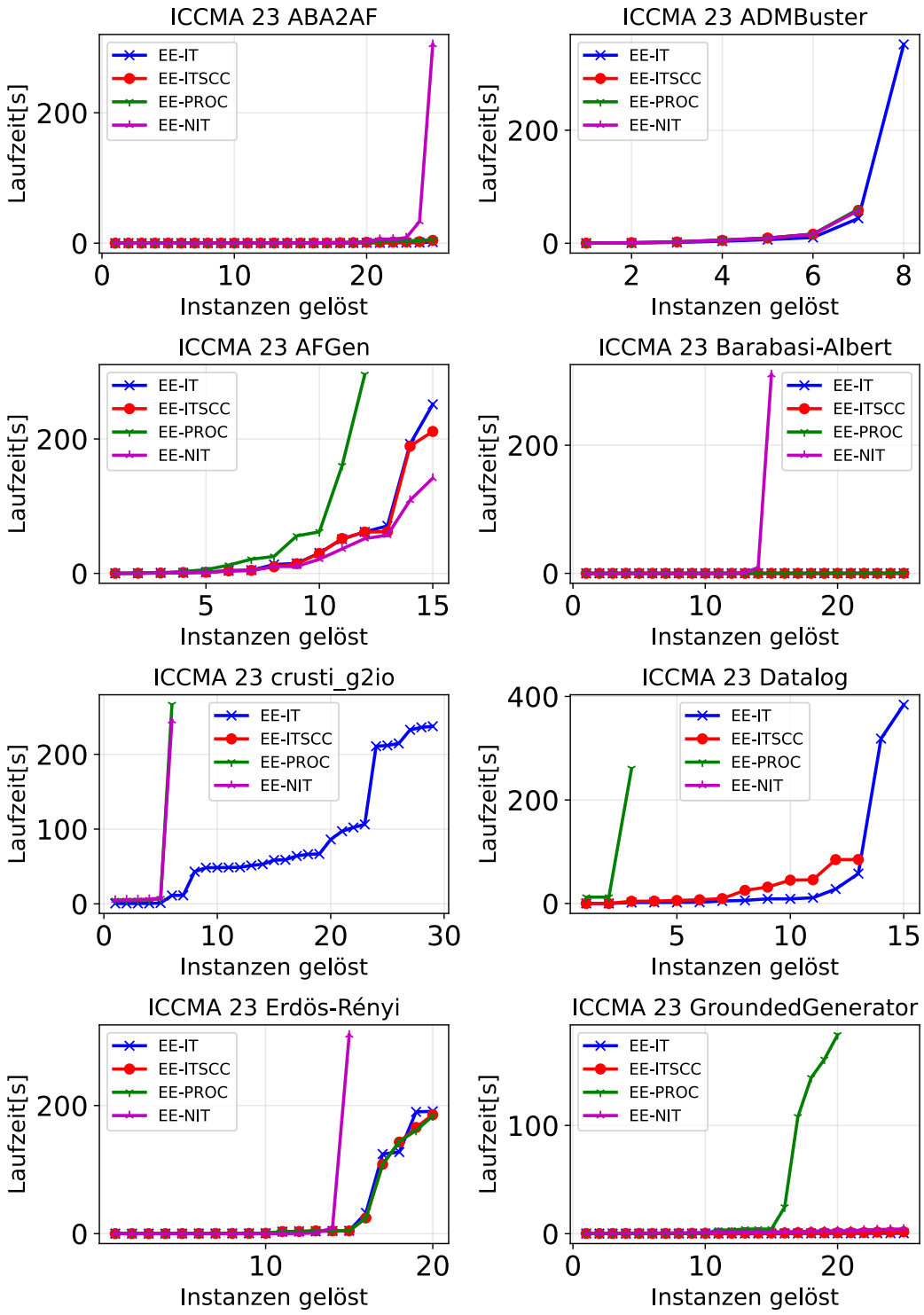

Abbildung 14: Laufzeiten der verschiedenen Algorithmen zur Berechnung initialer Mengen, differenziert nach den Kategorien des ICCMA'23 Datensatzes (a).

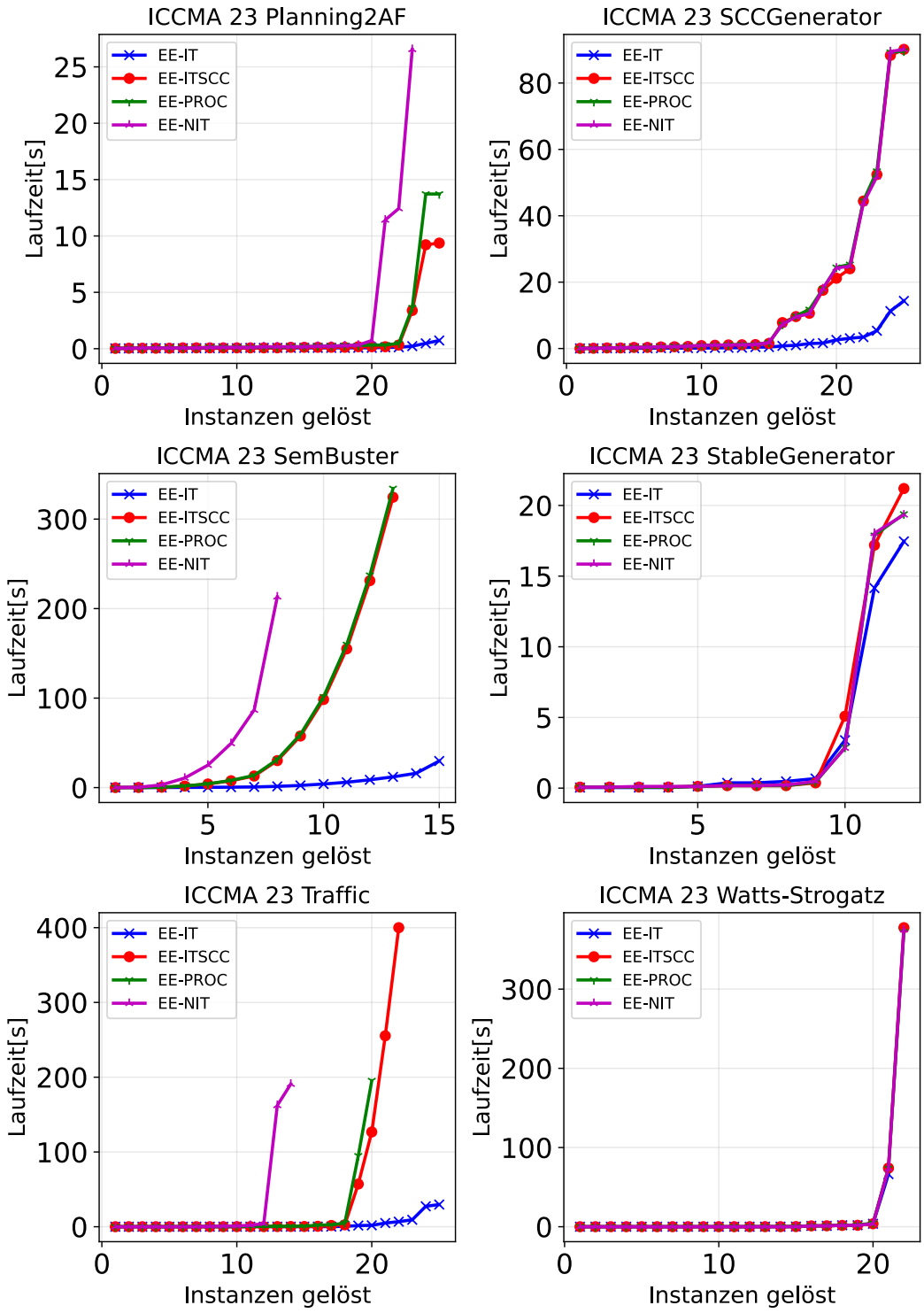

Abbildung 15: Laufzeiten der verschiedenen Algorithmen zur Berechnung initialer Mengen, differenziert nach den Kategorien des ICCMA'23 Datensatzes (b).

6.3.2 Vergleich der Ansätze

Im Folgenden werden die Unterschiede in der Performanz der einzelnen algorithmischen Ansätze zur Berechnung initialer Mengen in abstrakten Argumentationsgraphen kompakt dargestellt.

EE-IT Der iterative SAT-Ansatz, bei dem Modelle zulässiger, nicht-leerer Mengen iterativ minimiert werden, ist insgesamt der Algorithmus mit der stärksten Performanz. In der Abbildung 11 ist deutlich zu erkennen, dass dieser Ansatz in der Lage ist, innerhalb des Zeitlimits von 400 Sekunden mehr Instanzen zu lösen, als die anderen Algorithmen. Insgesamt ist dieser Ansatz in der Lage 87% aller Instanzen des ICCMA 2023 Datensatzes innerhalb des Zeitlimits zu lösen. In den Kategorien crustiG2io und Datalog war dieser Ansatz in der Lage deutlich mehr Instanzen zu lösen als die anderen algorithmischen Ansätze.

Mit Ausnahme der Graphenkategorie Watts-Strogatz (Small-World Netzwerke), ist der iterative SAT-Ansatz, ohne Betrachtung der starken Zusammenhangskomponenten, der Ansatz mit der geringsten Anzahl an Timeouts.

Für den Vergleich der Laufzeiten mit den anderen Ansätzen sollen nun im Folgenden die Abbildungen 14 und 15 betrachtet werden. In den Kategorien AdmBuster, AFgen, Erdös-Renyi, StableGenerator und Watts-Strogatz sind die Laufzeiten dieses Ansatzes relativ identisch zu den anderen Ansätzen. In den Kategorien SCCGenerator, SemBuster und Traffic zeigt der Ansatz bei einem Drittel bis zur Hälfte der jeweiligen Instanzen, eine deutlich höhere Performanz im Vergleich zu den anderen Ansätzen. Bei Instanzen der Kategorie SemBuster und Traffic ist der Unterschied in den Laufzeiten am stärksten ausgeprägt. Bei ca. einem Drittel der Instanzen dieser Kategorien beträgt der Unterschied in den Laufzeiten bereits mehrere Hundert Sekunden.

EE-ITSCC Der iterative SAT-Ansatz auf Basis der starken Zusammenhangskomponenten zeigt insgesamt eine schlechtere Performanz als der iterative SAT-Ansatz ohne Berücksichtigung der starken Zusammenhangskomponenten **EE-IT**. Insgesamt konnten 78% aller Instanzen des ICCMA 2023 Datensatzes innerhalb des Zeitlimits gelöst werden.

Insgesamt zeigt dieser Ansatz bei den Instanzen, die gelöst werden konnten, ein ähnliches Laufzeitverhalten wie die Ansätze **EE-IT** und dem prozeduralen Test auf Minimalität **EE-PROC**. Bei der Betrachtung des Laufzeitverhaltens, differenziert nach den Graphenkategorien (s. Abbildungen 14, 15) zeigen sich jedoch auch Unterschiede. In den Kategorien crustiG2io, SCCGenerator, SemBuster und Traffic bei vergleichbarer Performanz zu dem prozeduralen Test **EE-PROC** deutlich schlechter ab als der iterative SAT-Ansatz ohne Betrachtung der starken Zusammenhangskomponenten **EE-IT**.

EE-NITSCC Der nicht-iterative SAT-Ansatz mit Berücksichtigung der starken Zusammenhangskomponenten eines Abstraktionsgraphen, zeigt im Vergleich die niedrigste Performanz, insbesondere gemessen an der Anzahl an Timeouts. Der Anteil an Timeouts über alle Graphenkategorien hinweg beträgt 116 (35%). Somit konnten nur ca. zwei Drittel aller Instanzen des ICCMA 2023 Datensatzes innerhalb des Zeitlimits gelöst werden. In der Kategorie Datalog konnte keine Instanz innerhalb des Zeitlimits gelöst werden. Auch in den Kategorien Barabasi-Albert, Erdös-Renyi, SemBuster und Traffic schneidet die Performanz dieses Ansatzes deutlich schlechter ab. Eine Ausnahme stellt die Kategorie AFgen dar, hier war der nicht-iterative SAT-Ansatz bei ca. zwei Drittel der Instanzen in der Lage die Laufzeiten der übrigen Ansätze zu unterbieten (s. Abbildung 14). In der Kategorie GroundedGenerator war die Laufzeit dieses Ansatzes vergleichbar mit den rein SAT-basierten Ansätzen **EE-IT** und **EE-ITSCC** und somit deutlich stärker als der prozedurale Test auf Minimalität **EE-PROC**. In den übrigen Kategorien ist die Laufzeit dieses Ansatzes vergleichbar mit dem Laufzeitverhalten der Ansätze **EE-ITSCC** und **EE-PROC**.

EE-PROC Insgesamt zeigt der prozedurale Test auf Minimalität, mit Blick auf die Anzahl an Timeouts, eine schlechtere Performanz als die iterativen SAT-Ansätze. Die Anzahl an Timeouts beträgt insgesamt 94 (29%) über alle Kategorien des Datensatzes hinweg (s. Tabelle 4). Somit war dieser Ansatz insgesamt in der Lage 71% aller Instanzen des ICCMA 2023 Datensatzes zu lösen. In den Kategorien AFgen, Datalog und GroundedGenerator schneidet dieser Ansatz deutlich schlechter ab als die iterativen SAT-Ansätze (s. Abbildung 14). Auffällig jedoch ist, dass das Laufzeitverhalten dieses Ansatzes häufig sehr ähnlich dem des iterativen SAT-Ansatzes mit Berücksichtigung der starken Zusammenhangskomponenten **EE-ITSCC** ist. So liegen etwa in den Kategorien ABA2AF, Barabasi-Albert, Erdös-Renyi, SCCGenerator, StableGenerator und insbesondere SemBuster die Laufzeiten der beiden Ansätze sehr nah zusammen.

Zusammenfassung Nach Auswertung der Experimente zeigt sich, dass der iterative SAT-Ansatz ohne Betrachtung der starken Zusammenhangskomponenten **EE-IT**, der performanteste algorithmische Ansatz zur Berechnung initaler Mengen in abstrakten Argumentationsgraphen ist. Der nicht-iterative SAT-Ansatz mit Berücksichtigung der starken Zusammenhangskomponenten **EE-NITSCC** hat sich als der am wenigsten performante Ansatz dargestellt. Der iterative SAT-Ansatz mit Berücksichtigung der starken Zusammenhangskomponenten in einem abstrakten Argumentationsgraphen (**EE-ITSCC**) und der prozedurale Test auf Minimalität einer zulässigen Menge (**EE-PROC**) zeigen eine relativ ähnliche Performanz, mit Ausnahme der Graphenkategorien AFgen, Datalog und GroundedGenerator, in denen der Ansatz **EE-ITSCC** besser abschneidet.

Nachdem in diesem Kapitel die Ergebnisse des Vergleichs der in dieser Arbeit dargelegten algorithmischen Ansätze zur Berechnung initialer Mengen dargestellt wurden, folgen eine Diskussion und Zusammenfassung dieser Arbeit.

7 Diskussion und Ausblick

Aus den Ergebnissen der Experimente (Kapitel 6) geht deutlich hervor, dass der iterative SAT-Ansatz zur Kodierung und Berechnung initialer Mengen von Bengel und Thimm [6] die stärkste Performanz bietet. Insbesondere konnte die SAT-Kodierung der Beschränkung auf die starken Zusammenhangskomponenten eines abstrakten Argumentationsgraphen keinen Performanzvorteil realisieren. Ein Grund dafür könnte in der Art der Kodierung der starken Zusammenhangskomponenten liegen. Bei Argumentationsgraphen mit einer großen Anzahl an Argumenten bei gleichzeitig hoher Anzahl an starken Zusammenhangskomponenten steigt die Anzahl an (binären) Klauseln zur Kodierung der Beschränkung der Lösungsmenge auf starke Zusammenhangskomponenten stark an (s. Gleichung (6) in Kapitel 4.1.2). Auch muss beachtet werden, dass für die Berechnung der starken Zusammenhangskomponenten eines abstrakten Argumentatationsgraphen, insbesondere bei sehr großen Graphen mit einer hohen Anzahl an Argumenten und Attacken, eine gewisse Laufzeit anfällt, bevor die eigentliche Berechnung initialer Mengen stattfinden kann.

Neben alternativen SCC-Kodierungen in diesem Kontext, wäre es auch interessant zu ermitteln, inwiefern die Wahl des SAT-Solvers Auswirkungen auf die Permormanz der unterschiedlichen Ansätze hätte.

Bezüglich des nicht-iterativen SAT-Ansatzes wäre es darüberhinaus sinnvoll, eine Kodierung zu entwickeln, bei der gezielt nach minimalen, nicht-leeren Mengen gesucht wird. Dazu müsste in erster Linie eine Kodierung der nicht-Zulässigkeit von Argumentmengen entwickelt werden.

Die Arbeit wird im folgenden Kapitel mit einer kompakten Zusammenfassung abgeschlossen.

8 Zusammenfassung

In dieser Arbeit wurden unterschiedliche Ansätze zur Berechnung initialer Mengen in abstrakten Argumentationsgrapghen vorgestellt, implementiert und evaluiert. Initiale Mengen stellen Lösungen für atomare Konflikte in abstrakten Argumentatationsgraphen dar. Darüberhinaus bilden initiale Mengen die Grundbausteine der Extensionen der klassischen, zulässigkeitsbasierten Semantiken. Mit dem nicht-deterministischen Konstruktionsverfahren der Serialisierung von Extensionen der klassischen, zulässigkeitsbasierten Semantiken von Thimm, existiert ein wichtiger Anwendungsfall für die effiziente Berechnung initialer Mengen

Die Evaluation der in dieser Arbeit betrachteten algorithmischen Ansätze zur Berechnung initaler Mengen in abstrakten Argumentationsgraphen zeigte, dass die Beschränkung der Suche nach initialen Mengen auf die starken Zusammenhangskomponenten keinen Performanzvorteil realisieren. Insgesamt zeigte der iterative SAT-Ansatz von Bengel und Thimm [6], bei dem eine gefundene, zulässige Menge iterativ mittels weiterer SAT-Aufrufe minimiert wird, die beste Performanz. Der prozedurale

Test auf Minimalität einer zulässigen Menge zeigte bei höherer Anzahl an Timeouts häufig ein ähnliches Laufzeitverhalten wie der iterative SAT-Ansatz mit Beschränkung auf die starken Zusammenhangskomponenten. Die niedrigste Performanz stellte der nicht-iterative SAT-Ansatz, mit Beschränkung auf die starken Zusammenhangskomponenten, dar.

Literatur

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[2] Pietro Baroni, Martin Caminada, Massimiliano Giacomin, et al. Abstract argumentation frameworks and their semantics. In *Handbook of formal argumentation*, pages 159–236. College Publications, 2018.

[3] Ringo Baumann, Gerhard Brewka, and Markus Ulbricht. Revisiting the Foundations of Abstract Argumentation – Semantics Based on Weak Admissibility and Weak Defense. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(03):2742–2749, April 2020.

[4] Christoph Beierle and Gabriele Kern-Isberner. *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen*. Computational Intelligence. Springer Fachmedien Wiesbaden, Wiesbaden, 2019.

[5] Lars Bengel and Matthias Thimm. Serialisable Semantics for Abstract Argumentation. In Francesca Toni, Sylwia Polberg, Richard Booth, Martin Caminada, and Hiroyuki Kido, editors, *Frontiers in Artificial Intelligence and Applications*. IOS Press, September 2022.

[6] Lars Bengel and Matthias Thimm. Towards Parallelising Extension Construction for Serialisable Semantics in Abstract Argumentation. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning*, pages 732–736, 8 2023.

[7] Philippe Besnard, Sylvie Doutre, and Andreas Herzig. Encoding Argument Graphs in Logic. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, volume 443, pages 345–354. Springer International Publishing, Cham, 2014. Series Title: Communications in Computer and Information Science.

[8] Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. CaDiCaL 2.0. In Arie Gurfinkel and Vijay Ganesh, editors, *Computer Aided Verification*, pages 133–152, Cham, 2024. Springer Nature Switzerland.

[9] Martin Caminada and Paul Dunne. Strong admissibility revisited: Theory and applications. *Argument & Computation*, 10(3):277–300, 2020.

[10] Martin Caminada, Bart Verheij, G Danoy, M Seredynski, R Booth, B Gateau, I Jars, and D Khadraoui. On the existence of semi-stable extensions. *argumentation*, 3:4, 2010.

[11] Federico Cerutti, Massimiliano Giacomin, and Mauro Vallati. Generating structured argumentation frameworks: Afbenchgen2. In *Computational Models of Argument*, pages 467–468. IOS Press, 2016.

[12] Federico Cerutti, Nir Oren, Hannes Strass, Matthias Thimm, and Mauro Vallati. A benchmark framework for a computational argumentation competition. In *Computational Models of Argument*, pages 459–460. IOS Press, 2014.

[13] 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, March 2015.

[14] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery.

[15] 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, September 1995.

[16] Wolfgang Dvorák and Paul E Dunne. Computational problems in formal argumentation and their complexity. *JOURNAL OF APPLIED LOGICS-IFCOLOG JOURNAL OF LOGICS AND THEIR APPLICATIONS*, 4(8):2557–2622, 2017.

[17] Yong Gao. A random model for argumentation framework: Phase transitions, empirical hardness, and heuristics. In *IJCAI*, pages 503–509, 2017.

[18] Matti Järvisalo, Tuomo Lehtonen, and Andreas Niskanen. Iccma 2023: Benchmarks and raw results. *10.5281/zenodo.8348039*, September 2023.

[19] Matti Järvisalo, Tuomo Lehtonen, and Andreas Niskanen. ICCMA 2023: 5th International Competition on Computational Models of Argumentation. *Artificial Intelligence*, 342:104311, 2025.

[20] Jonas Klein and Matthias Thimm. probo2: A Benchmark Framework for Argumentation Solvers. In *Computational Models of Argument*, pages 363–364. IOS Press, 2022.

[21] Jean-Marie Lagniez, Emmanuel Lonca, Jean-Guy Mailly, and Julien Rossit. The Fourth International Competition on Computational Models of Argumentation.

[22] Jean-Marie Lagniez, Emmanuel Lonca, Jean-Guy Mailly, and Julien Rossit. A new evolutive generator for graphs with communities and its application to abstract argumentation. In *First International Workshop on Argumentation and Applications (Arg&App 2023)*, volume 3472. CEUR-WS, 2023.

[23] Tuomo Lehtonen, Johannes P Wallner, and Matti Järvisalo. From structured to abstract argumentation: Assumption-based acceptance via af reasoning. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 57–68. Springer, 2017.

[24] Andreas Niskanen and Matti Järvisalo. Algorithms for dynamic argumentation frameworks: An incremental sat-based approach. In *ECAI 2020*, pages 849–856. IOS Press, 2020.

[25] Erdos Renyi. On random graph. *Publicationes Mathematicate*, 6:290–297, 1959.

[26] Kai Sauerwald and Matthias Thimm. *Logik. Skript zur gleichnamigen Vorlesung an der Fernuniversität in Hagen.* 2025.

[27] Jeremy Siek, Lie-Quan Lee, and Andrew Lumsdaine. Boost graph library, 2000.

[28] Robert Tarjan. Depth-first search and linear graph algorithms. In *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, pages 114–121, 1971.

[29] Matthias Thimm. Revisiting initial sets in abstract argumentation. *Argument & Computation*, 13(3):325–360, October 2022.

[30] Matthias Thimm. *Formale Argumentation. Skript zur gleichnamigen Vorlesung an der Fernuniversität in Hagen.* 2025.

[31] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440–442, 1998.

[32] Yuming Xu and Claudette Cayrol. Initial sets in abstract argumentation frameworks. *Journal of Applied Non-Classical Logics*, 28(2-3):260–279, July 2018.

[33] Yuming Xu, Lidong Xu, and Claudette Cayrol. Structural Analysis of Extension-Based Argumentation Semantics with Joint Acceptability. In Beishui Liao, Thomas Ågotnes, and Yi N. Wang, editors, *Dynamics, Uncertainty and Reasoning*, pages 1–20, Singapore, 2019. Springer Singapore.

[34] Bruno Yun, Madalina Croitoru, Srdjan Vesic, and Pierre Bisquert. Dagger: Datalog+/-argumentation graph generator. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1841–1843, 2018.