FernUniversität in Hagen
Fakultät für Mathematik und Informatik

AIG
Artificial Intelligence Group

Unvollständiges Abstraktes Dialektisches Argumentieren

Masterarbeit

zur Erlangung des Grades Master of Science (M.Sc.)
im Studiengang Informatik

vorgelegt von

Eike-Christian Weiss

Erstgutachter:	Prof. Dr. Matthias Thimm
	Artificial Intelligence Group
Betreuer:	Kenneth Skiba
	Artificial Intelligence Group

Erklärung

Ich erkläre, dass ich die Masterarbeit 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 Arbeit 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 Arbeit nehme ich zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate geprüft werden kann und ausschließlich für Prüfungszwecke gespeichert wird.

	Ja	Nein
Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	☐	☒
Der Veröffentlichung dieser Arbeit auf der Webseite des Lehrgebiets Künstliche Intelligenz stimme ich zu.	☐	☒
Der Text dieser Arbeit ist unter einer Creative Commons Lizenz (CC BY-SA 4.0) verfügbar.	☐	☒
Der Quellcode ist unter einer GNU General Public License (GPLv3) verfügbar.	☐	☒
Die erhobenen Daten sind unter einer Creative Commons Lizenz (CC BY-SA 4.0) verfügbar.	☐	☒

München, den 26.02.2024
(Ort, Datum)

(Unterschrift)

Abstract

A problem in artificial intelligence (AI) is incomplete information. This work examines the problem in the research area of formal argumentation, specifically using *abstract dialectical frameworks* (ADF).

The impact of incomplete information on the ADFs was analyzed. Incompleteness can occur, for example, through the course of an argumentation with changing positions and arguments. The uncertainty of a statement arises, for example, when combining databases with inconsistent values for an object. One finding is that the components of the ADFs have a strong interaction with each other. This limits the analog application of existing approaches to dealing with incomplete information.

This work presents two approaches that contribute to dealing with the problem. The *adaptive ADFs* eliminate the lack of flexibility to the dynamics of an argumentation. The reasoning-patterns introduced make the framework more adaptable in combination with the so-called goal setting function. This means that changing positions and arguments can be taken into account immediately.

By *approximating* uncertain statements, the evaluation of an argumentation is successful. To do this, all three-valued interpretations are arranged in a lattice according to information and truth value and evaluated with a special operator. If an interpretation contains an uncertain statement, the interpretation can be approximated using the corresponding neighbors of the lattice. As a result, the ADFs can deal with uncertainty and achieve a result.

The approaches presented enable an AI system to deal more robustly with incomplete information.

Kurzfassung

Ein Problem in der künstlichen Intelligenz (KI) ist unvollständige Information. Diese Arbeit untersucht das Problem auf dem Forschungsgebiet der formalen Argumentation, konkret an den *abstrakt dialektischen Frameworks* (ADF).

Analysiert wurde die Auswirkung von unvollständiger Information auf die ADFs. Unvollständigkeit kann beispielsweise durch den Verlauf einer Argumentation mit sich ändernden Positionen und Argumenten auftreten. Die Unsicherheit eines Statements entsteht z.B. beim Zusammenfügen von Datenbanken mit inkonsistenten Werten bzgl. eines Objekts. Eine Feststellung ist, dass die Komponenten der ADFs eine starke Wechselwirkung untereinander besitzen. Hierdurch wird die analoge Anwendung bereits vorhandener Ansätze im Umgang mit unvollständiger Information eingeschränkt.

In dieser Arbeit werden zwei Ansätze präsentiert, die zur Handhabung des Problems beitragen. Mit den *adaptiven ADFs* wird die mangelnde Flexibilität gegenüber Veränderungen bei einer Argumentation beseitigt. Die eingeführten Argumentationsmuster machen das Framework in Kombination mit der Zielvorgabefunktion anpassungsfähiger. Dadurch können sich verändernde Positionen und Argumente sofort berücksichtigt werden.

Mit der *Approximation* von Statements, deren Existenz nicht garantiert ist, gelingt die Auswertung einer Argumentation mit Unsicherheiten. Dazu werden alle dreiwertigen Interpretationen in einem Gitter nach Informations- und Wahrheitswert geordnet und mit einem speziellen Operator ausgewertet. Enthält eine Interpretation ein unsicheres Statement, kann die Interpretation mit den entsprechenden Nachbarn des Gitters angenähert werden. Infolgedessen können die ADFs mit Unsicherheiten umgehen und ein fundiertes Ergebnis erzielen.

Die vorgestellten Ansätze ermöglichen einem KI-System den robusteren Umgang mit unvollständiger Information.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AL	Aussagenlogik
AF	abstraktes Argumentation Framework
ADF	abstrakt dialektisches Framework
bspw.	beispielsweise
DNF	disjunktive Normalform
EStG	Einkommensteuergesetz
gdw.	genau dann, wenn
geg.	gegeben
iAF	unvollständig abstraktes Argumentation Framework
KI	künstliche Intelligenz
KNF	konjunktive Normalform
PL1	Prädikatenlogik 1. Stufe
unbes.	unbeschränkt
Whd	Wiederholung
zshg.	zusammenhängend

🖼	interaktive Grafik
↩	Rücksprung-Verlinkung

1 Einleitung

Die Argumentation ist fester Bestandteil der gewöhnlichen zwischenmenschlichen Kommunikation. Durch den Austausch von Argumenten können Probleme verdeutlicht, Standpunkte klar gemacht und Schlussfolgerungen gezogen werden. Folglich ist die formale Argumentation auch ein relevantes und viel beachtetes Forschungsgebiet der künstlichen Intelligenz (KI). Vereinfacht ausgedrückt, versucht die KI auf einem maschinellen System (i.d.R. Computer) menschenähnliche Schlüsse nachzubilden.

Ein zentrales Problem dabei ist der Umgang eines KI-Systems mit unvollständiger Information. Im schlechtesten Fall kann das System in einer unbekannten Situation versagen. In bestimmten Anwendungsdomänen wie bspw. der Luftfahrt führt dies zu Risiken und Gefahren, wenn das System nicht mehr in der Lage ist Entscheidungen zu treffen. In weniger kritischen Domänen leidet das Vertrauen ins System, wenn unzuverlässige oder fehlerhafte Entscheidungen getroffen werden. Folglich ist der adäquat maschinelle Umgang mit unvollständiger Information eine Herausforderung für die KI-Forschung und -Entwicklung.

Diese Arbeit beschäftigt sich mit dem unvollständigen abstrakt dialektischen Argumentieren. Aufbauend auf den abstrakten Argumentation Frameworks (AFs) von Dung [1], haben Brewka und Woltran in [2] mit den abstrakt dialektischen Frameworks (ADFs) einen Formalismus eingeführt, mit dem sich dialektische Argumentation auf einem Computer nachbilden lässt. Konkret führt unvollständige Information im Kontext der dialektischen Argumentation dazu, dass der Argumentationsgraph nicht (zutreffend) konstruiert werden kann. Eine Möglichkeit des Umgangs von ADFs mit unvollständiger Information wird in dieser Arbeit untersucht.

1.1 Motivation

Es gibt Situationen, in denen im Rahmen einer konkreten Problemstellung eine Entscheidung getroffen werden muss. Häufig wird in diesen Situationen auf Experten des entsprechenden Gebiets oder auf maschinelle Systeme zurückgegriffen. Überschreitet

das Problem eine bestimmte Komplexität, ist es schwierig zu einer fundierten Entscheidung zu kommen. Insbesondere maschinelle Systeme stoßen an ihre Grenzen, wenn entscheidungsrelevante Informationen unsicher oder sogar unvollständig sind. Jede Anwendungsdomäne ist von den Problemen betroffen, deren Ursache in unvollständiger Information begründet ist, wenn Information nicht absolut vorliegt.

Bei einer Argumentation können Fakten ausgespart, Argumente verkürzt oder Positionen überholt sein. Der Verlauf einer Argumentation erfolgt dynamisch, d.h., dass argumentierende Parteien sich ändernden Argumentationsstrukturen anpassen können (sog. *Argumentationsdynamik*). Der Umgang mit der Argumentationsdynamik gelingt Menschen in alltäglichen Situationen problemlos. Ein maschinelles System, dessen Grundlage seine Wissensbasis ist, kann mit Dynamik, die zu unbekannten neuen Situationen führt, nicht gut umgehen. Häufig muss ein System erst auf den konkreten Fall angepasst oder trainiert werden.

Ein weiteres Problem stellen auch die zugrundeliegenden Daten dar, wenn beispielsweise dieselben Objekte mit unterschiedlichen Werten in verschiedenen Datenbanken abgelegt sind. Beim Zusammenführen der Daten kann sich herausstellen, dass der Wert des Objekts inkonsistent ist. Es handelt sich dann um unsichere Information, mit denen auch ein maschinelles System umgehen muss.

Die Motivation für diese Arbeit liegt in der Erforschung des Umgangs mit unvollständiger Information im Bereich des formalen Argumentierens. Konkret werden dazu die ADFs und deren Eigenschaften in Bezug auf unvollständige Informationen und unsichere Daten untersucht. Im Ergebnis bietet diese Arbeit einen Vorschlag zur Adaptivität der ADFs an und zeigt wie die Modelle der ADFs und ihre Auswertung mit Unsicherheit umgehen können. Der wissenschaftliche Beitrag soll einen besseren Umgang mit unvollständiger Information zeigen, der zur Lösung der o.g. Probleme beiträgt.

1.2 Formale Argumentation

Das (abstrakt) dialektische Argumentieren gehört zum Teilgebiet des formalen Argumentierens der KI. Die abstrakte Argumentation ist in den letzten Jahrzehnten zu

einem stark erforschten Thema der KI-Forschung geworden. Sie findet Anwendung in den Bereichen Decision Support [3], Legal Reasoning [4] und E-Government [5].

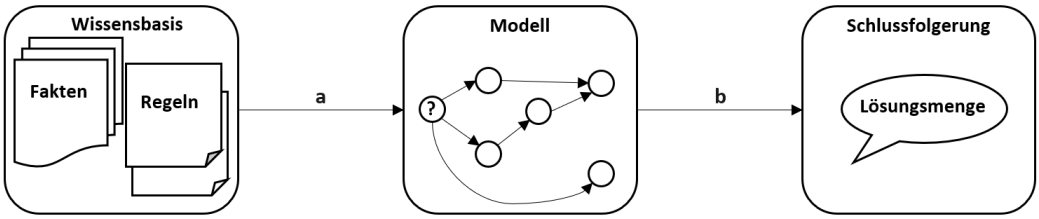


Abbildung 1.1: Die Abbildung skizziert den Aufbau eines wissensbasierten Systems. Die Pfeile stellen Übergänge dar, für die algorithmische Lösungen benötigt werden. Der Übergang (a) konstruiert aus einer gegebenen Wissensbasis ein geeignetes Modell (Argumentationsgraph). Der Übergang (b) wertet dieses Modell aus und findet mögliche Lösungsmengen (*Inferenz*).

Nach Bench-Capon und Dunne [6] *beschäftigt sich die Argumentationstheorie damit, wie Argumente im Kontext einer Frage vorgeschlagen, diskutiert und gelöst werden können stets unter dem Aspekt, dass mehrere unterschiedliche Meinungen vertretbar sind.*

Der Verlauf einer herkömmlichen Argumentation entspricht folgendem Schema:

1. Die These ist die Aussage (Position), die unterstützt oder verteidigt werden soll.
2. Argumente sind Gründe (Beweise), die verwendet werden, um die These zu unterstützen.
3. Gegenargumente sind alternative Standpunkte (Einwände), die gegen die These vorgebracht werden.
4. Die Schlussfolgerung ist das Ergebnis, basierend auf der Bewertung der präsentierten Argumente und Gegenargumente.

Das Ziel der Formalisierung von Argumentation ist, dass menschenähnliche Schlussfolgerungen automatisiert gezogen werden können.

Dieser Prozess ist schematisch in Abbildung 1.1 dargestellt. Er benötigt eine Wissensbasis, bestehend aus Argumenten und (Auswertungs-) Regeln. Anhand der Wissensbasis kann eine abstrakte Repräsentation in Form eines Modells (Argumentationsgraph)

erfolgen. Die Argumente des Graphen werden unter Einhaltung bestimmter Regeln ausgewertet, d.h., dass nach Mengen von Argumenten gesucht wird, die unter derselben (Argumentations-) Semantik akzeptiert sind. Die zu einer Semantik gehörenden akzeptierten Argumentmengen werden als Extensionen bezeichnet. Abschließend lassen sich hieraus Schlussfolgerungen ableiten.

Der Inhalt der Arbeit gliedert sich in folgende Kapitel:

- Kapitel 2 stellt die notwendigen Grundlagen vor, insbesondere der abstrakten Argumentation Frameworks (AF) 2.1 und der Aussagenlogik (AL) 2.2 sowie der abstrakt dialektischen Frameworks (ADF) 2.3.
- Kapitel 3 skizziert die konkrete Situation, die unvollständige Information bei ADFs mit sich bringt. Unvollständige Information wird aufgeschlüsselt in bisher unbekannte Information 3.1 und unsichere Information 3.2.
- Kapitel 4 stellt einen Ansatz für *adaptive ADFs* (kurz: ζADF) vor, mittels dessen eine Argumentation mit hoher Argumentationsdynamik direkt ausgewertet werden kann.
- Kapitel 5 zeigt den Umgang von ADFs mit unvollständiger Information. Dabei wird ein unsicheres Statement durch seine Nachbarn im aufgestellten Bi-Gitter approximiert.
- Kapitel 6 fasst die wichtigsten Ergebnisse der Arbeit zusammen und bietet Ausblicke für zukünftige Arbeiten an.

2 Grundlagen

In diesem Kapitel werden die benötigten formalen Grundlagen vorgestellt. Die Grundlage dieser Arbeit sind die abstrakt dialektischen Frameworks (ADFs), die eine Verallgemeinerung der abstrakten Argumentation Frameworks (AFs) darstellen. Dabei weisen sie ihren Statements Akzeptanzbedingungen in Form von aussagenlogischen Formeln zu. Es werden:

- in Abschnitt 2.1 die AFs aus der Arbeit [1] von Dung betrachtet und
- in Abschnitt 2.2 die Aussagenlogik (AL) behandelt.

Diese Kombination führt schließlich zu den ADFs, auf die im Abschnitt 2.3 eingegangen wird.

2.1 Argumentation Framework

Ausgangspunkt dieser Arbeit ist die Abbildung von Argumentation auf einem Computersystem. Menschenähnlich soll ein System Argumente und Gegenargumente unter den Bedingungen eines Dialogs mehrerer Beteiligten entwickeln und abwägen können, um zu einer Schlussfolgerung zu gelangen.
Es gibt eine Reihe von Ansätzen, die sich mit der internen Struktur eines Arguments (Behauptung, Begründung, Beweis) beschäftigt haben. Im Gegensatz zu diesen Ansätzen hat Dung in seiner Arbeit [1] mit den AFs einen Ansatz gewählt, bei dem nur die externe Struktur von Argumenten in Gänze betrachtet wird. Dieser Ansatz konzentriert sich auf die Beziehungen, die Argumente zueinander haben. Er fasst diese Konzeption mit dem folgenden Sprichwort zusammen:

> „*The one who has the last word laughs best.*“ [1, S. 322]

Mit anderen Worten: Wer das letzte Wort hat, der hat Recht. Das Ziel ist es herauszufinden, welches der Argumente als „letztes“ übrig bleibt. Die Idee dahinter ist, dass eine These glaubhaft ist, wenn erfolgreich gegen ihre Gegenargumente argumentiert werden kann. Dieser Abschnitt gibt einen Überblick über das von Dung eingeführte Framework und dessen Semantiken.

2.1.1 Konzeption

Der von Dung in [1] vorgebrachte Ansatz eines AFs gehört zu den am häufigsten genutzten Ansätzen. Mit dem namensgebenden Wort *abstrakt* ist gemeint, dass ein Argument losgelöst von seiner internen Struktur (Aufbau) betrachtet wird. Die Argumente werden wie Container (Entitäten) betrachtet, die beliebig befüllt werden können. Folglich ist die externe Struktur der Argumente zueinander (Relationen) ein wichtiger Aspekt dieses Frameworks.
Zwischen jeweils einem Paar von Argumenten kann eine Beziehung in Form einer Relation bestehen. Nach dem Prinzip eines „last argument standing“ sind diese binären Relationen immer als Angriffe zu interpretieren. Das abstrakte Konzept der AFs erlaubt die Abbildung eines Dialogs zwischen mehreren Parteien als Modell (Argumentationsgraph). Zur Darstellung wird ein *gerichteter Graph*[1] verwendet, dessen Knoten die Argumente und dessen Kanten die Relationen einer Argumentation repräsentieren. Folgendes Beispiel einer fiktiven Situation soll als Einführung dienen.

Parteien	**Aussagen**	**Zuordnung**
Beteiligter 1	„Ich konnte fahren, weil meine Ampel auf grün stand. Der andere muss rot gehabt haben.“	$\widehat{=} a$
Beteiligter 2	„Das ist quatsch. Meine Ampel zeigte grün an, deshalb hatte ich Vorfahrt.“	$\widehat{=} b$
Zeuge 1	„Der Beteiligte 2 hatte lediglich einen festen Grünpfeil ohne Lichtsignal.“	$\widehat{=} c$

Tabelle 2.1: Aussagen der Parteien über den Unfallhergang und deren Zuordnung

Beispiel 2.1.1 (Unfallsituation)**.**
Zwischen zwei Verkehrsteilnehmern kommt es an einer Kreuzung mit Ampelanlage zu einem Unfall. Es soll anhand der Aussagen der (Unfall-) Beteiligten und des Zeugen aus Tabelle 2.1 geschlussfolgert werden, wer der Unfallverursacher ist. Die Beteiligten gehen davon aus, dass sie im Recht sind und zweifeln die Aussage des jeweils anderen an.

Die interne Struktur der Aussagen respektive Argumente ist zwar unterschiedlich, kann jedoch aufgrund der Abstraktionsfähigkeit der AFs ohne weitere Anpassung

[1] siehe Definition A.1.1 auf Seite 91.

in das Framework übernommen werden. Das entsprechende AF hat drei Argumente $A = \{a, b, c\}$ und die Relationen $R = \{(a, b), (b, a), (c, b)\}$. Die Gegensätzlichkeit der Argumente zueinander wird durch ihre Relation ausgedrückt. Die Abbildung 2.1 gibt den entsprechenden (Argumentations-) Graph wieder.

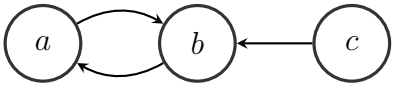


Abbildung 2.1: Der Argumentationsgraph des dazugehörigen AFs. Die Argumente werden als Knoten repräsentiert. Zwischen diesen kann eine (Angriffs-) Relation existieren, welche mittels Pfeil dargestellt wird.

⌟

Intuitiv stellt sich nun die Frage, welches Argument das „letzte“ ist und somit die anderen Argumente zurückweist. Nach [7, S. 307] kann die Argumentation als *Evaluations- und Folgerungsprozess* betrachtet werden. Mit den AFs kann ein Konflikt zwischen Parteien einer Argumentation dargestellt werden. Dieses Modell muss in einem nächsten Schritt ausgewertet werden, um akzeptierte Argumente zu ermitteln. Der Prozess wird in der Fortsetzung des Beispiels weitergeführt.

Beispiel. (Fortsetzung)
Der Argumentationsgraph aus Abbildung 2.1 besitzt ein Argument, welches für sich alleine steht. Das Argument c hat demnach keine eingehende Kante (Pfeil), folglich kein entgegenstehendes Argument (Gegenargument). Deshalb wird c als glaubhaft erachtet, solange es nicht angegriffen wird. Ein Angriff ist eine Relation zwischen zwei Argumenten. Graphisch wird dies als Pfeil vom Startknoten (Angreifer) zum Endknoten (Angegriffenen) dargestellt. Notiert wird ein Angriff als $(a, b) \in R$, d.h., das Argument a greift b an (vgl. obige Abbildung). Formal ausgedrückt besitzt jeder Knoten, dessen Eingangsgrad größer Null ist, einen Angreifer[2].

Offensichtlich gibt es zwischen den Argumenten a und b einen Konflikt. Beide Argumente greifen sich gegenseitig an, wodurch es keinem der beiden gelingt als glaubhaft zu gelten. Eine Reihe von Angriffen[3] stellt eine Besonderheit dar. Die Angriffsreihe $(b, a), (c, b) \in R$ gibt an, dass a von b angegriffen wird, wodurch die Glaubhaftigkeit von a angezweifelt wird. Das Argument b wird jedoch selbst von c angegriffen

[2] $indeg(v) > 0$; siehe auch Definition A.1.3 auf Seite 91
[3] Angriffsreihe: $(v_2, v_1), (v_3, v_2), \ldots, (v_n, v_{n-1}) \in R$

und dadurch ebenfalls mit Zweifel belegt. Eine solche Konstellation wird Verteidigung genannt und kann auf die ursprüngliche Idee *des letzten Arguments* zurückgeführt werden. Da b als Angreifer von a selbst nicht glaubhaft ist, kann er a nicht erschüttern.

Konkret auf das Beispiel der Unfallsituation bezogen, widerspricht der Beteiligte 2 dem Beteiligten 1. Die Aussage des Beteiligten 2 wird aber durch den Zeugen widerlegt. Das von b angegriffene Argument a wird von c verteidigt und bleibt dadurch glaubhaft. ⌟

Durch das Beispiel wurde intuitiv vermittelt, wie die AFs eingesetzt werden können. Bisher wurden vorkommende Begriffe nur vage formuliert, sodass die entsprechenden Definitionen im Folgenden nachgeholt werden. Einige Definitionen entstammen der Graphentheorie, diese werden an gegebener Stelle verlinkt und sind im Anhang unter A.1 zu finden.

Ein AF ist ein Paar, bestehend aus einer endlichen Menge[4] von Argumenten und einer Menge von binären (Angriffs-) Relationen zwischen diesen Argumenten. Argumente sind anfechtbare Entitäten, die sich gegenseitig angreifen können. Dies führt zu folgender Definition:

Definition 2.1.1 (abstraktes Argumentation Framework aus [1, Def. 2])**.**
Ein abstraktes Argumentation Framework ist ein Paar $\mathrm{AF} = (\mathrm{A}, \mathrm{R})$, wobei $\mathrm{A} \subseteq \mathcal{U}$ eine endliche Menge an Argumenten aus dem Universum $\mathcal{U}$ und $\mathrm{R} \subseteq \mathrm{A} \times \mathrm{A}$ eine (Angriffs-) Relation aus eben diesen Argumenten ist.

Anmerkung (Angriffsnotationen)**.**
Die Notation $(a, b) \in \mathrm{R}$ (kurz: (a, b)) gibt an, dass das Argument a das Argument b angreift. Die Notation S bezeichnet eine Menge an Argumenten $S \subseteq A$. Analog gilt für eine Argumentmenge, dass (S, b) einen Angriff aus S auf b und (b, S) einen Angriff von b auf S darstellt.

Eine wichtige Erkenntnis aus Beispiel 2.1.1 ist die Glaubhaftigkeit eines Arguments. Ein Argument ist glaubhaft, wenn es kein Gegenargument gibt oder das Gegenargument ebenfalls durch einen Angriff in Zweifel gezogen wird. Dies führt zu folgender Definition.

[4] Es handelt sich um eine endliche Menge aus einem beliebig großen Universum $\mathcal{U}$.

Definition 2.1.2 (Verteidigung; nach [1, S. 326])**.**
Sei $(a, b) \in R$ eine Angriffsrelation, dann wird das Argument b verteidigt genau dann, wenn (gdw.) $\exists x \in A : (x, a) \in R$ gilt. Auch $x = b$ ist eine legitime Verteidigung (*Selbstverteidigung*).

Das Beispiel 2.1.1 besitzt zwei Argumente a und c, die nicht unmittelbar einen Konflikt haben. Solche Argumente sind konfliktfrei, da sie nicht in (Angriffs-) Relation zueinander stehen. Graphisch kann dies in Abbildung 2.1 durch eine fehlende Kante zwischen den Knoten a und c nachvollzogen werden.

Definition 2.1.3 (konfliktfrei [1, Def. 5])**.**
Sei $AF = (A, R)$ ein Argumentation Framework. Eine Teilmenge $S \subseteq A$ heißt konfliktfrei gdw. $\nexists (a, b) \in S : a$ greift b an. In der Menge S gibt es keine Argumente, die sich (gegenseitig) angreifen.

Bisher wurde der Begriff „glaubhaft“ verwendet, der nunmehr durch den Begriff *akzeptabel* und dessen Definition ersetzt wird.

Definition 2.1.4 (akzeptabel [1, Def. 6 Abs. 1])**.**
Sei $AF = (A, R)$ ein Argumentation Framework. Ein Argument $a \in S$ ist akzeptabel (*acceptable*) gdw. $\forall b \in A : (b, a)$ gilt (S, b). Mit anderen Worten, wenn für jedes Argument $b \in A$, das a angreift, gilt: b wird von S angegriffen (a wird von S verteidigt).

Häufig soll nicht ein einziges Argument betrachtet werden, sondern eine Menge an Argumenten, die zulässig ist.

Definition 2.1.5 (zulässig [1, Def. 6 Abs. 2])**.**
Sei $AF = (A, R)$ ein Argumentation Framework. Die Menge $S \subseteq A$ ist zulässig (*admissible*) gdw. S konfliktfrei ist und jedes Argument in S akzeptabel ist.

Diese grundlegenden Definitionen erlauben es, dass bestimmte Regeln (Argumentationssemantiken) aufgestellt werden, anhand derer sich Mengen von Argumenten eines AFs (sog. *Extensionen*) auswählen lassen.

2.1.2 Semantiken

Im Rahmen dieser Arbeit werden nur die Semantiken, die Dung in [1] aufgeführt hat, betrachtet. Das sind die bevorzugte, stabile, grundierte und vollständige Semantik. Weitere Semantiken sind z.B. in der Arbeit [8] von Baroni et al. aufgeführt.

Von besonderem Interesse sind alle Semantiken, mit denen ein *Agent* zu einer Schlussfolgerung kommt. Insbesondere ist das, dass leichtgläubige (*credulous*) und das skeptische (*skeptical*)[5] Entscheidungsproblem. Diese beiden sind auch zu Vergleichszwecken relevant.
Ersteres entspricht genau einer bevorzugten (*preferred*) Semantik bei einem AF. Dabei ist eine bevorzugte Semantik eine zulässige Menge an Argumenten, die keine echte Teilmenge einer anderen zulässigen Argumentenmenge ist.

Definition 2.1.6 (bevorzugt [1, Def. 7])**.**
Sei $AF = (A, R)$ ein Argumentation Framework. Eine Menge an Argumenten $S \subseteq A$ ist eine bevorzugte (*preferred*) Extension gdw. S die maximal zulässig Menge der Argumente des AFs ist.

In dem AF aus Beispiel 2.1.1 wäre $S = \{a, c\}$ eine Extension, die sich der bevorzugten Semantik zuordnet lässt. Konkret bedeutet dies, dass die Argumente a und c zusammen eine These stützen, die geglaubt werden kann. In diesem Fall wird die These:

„Ich [Beteiligter 1] konnte fahren, weil meine Ampel auf grün stand. Der andere [Beteiligter 2] muss rot gehabt haben."

unterstützt durch das Argument:

„Der Beteiligte 2 hatte lediglich einen festen Grünpfeil ohne Lichtsignal".

Ein plausibler Schluss für einen leichtgläubigen Agent wäre, dass der Unfallverursacher der Beteiligte 2 ist. Zu beachten ist, dass jedes AF mindestens eine bevorzugte Extension hat, wobei auch die leere Menge als Lösung gilt.

Die bisherigen Definitionen werden durch das folgende Beispiel verdeutlicht.

[5] Schreibweise aus der verwendeten Literatur übernommen.

Beispiel 2.1.2.
Dazu sei $AF = (\{a, b, c, d, e\}, \{(a, b), (a, d), (b, b), (c, d), (d, e)\})$ gegeben. Der korrespondierende Graph ist in Abbildung 2.2 dargestellt.

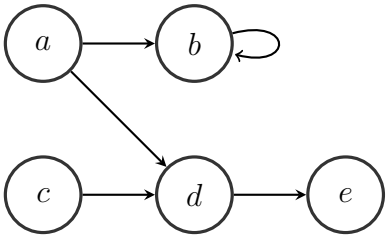


Abbildung 2.2: Argumentationsgraph zum gegebenen AF

Zuerst werden nach Definition 2.1.3 die konfliktfreien Mengen des AFs bestimmt. Zum Beispiel ist $S = \{a, c\}$ eine entsprechende Menge, da sich die beiden Argumente a und c nicht gegenseitig angreifen. Alternativ wären auch $S = \{a, e\}$, $S = \{c, e\}$ oder $S = \{a, c, e\}$ konfliktfreie Mengen. Ein Gegenbeispiel ist die Menge $S = \{a, c, d\}$, weil d von a und c angegriffen wird. Ein AF kann unterschiedliche konfliktfreie Mengen haben.

Eine akzeptable Menge nach Definition 2.1.4 ist $S = \{a, c, e\}$. Das Argument e wird zwar von d angegriffen, jedoch wird e von a bzw. c verteidigt. Sowohl a als auch e sind dabei Elemente aus S. Alternativ wäre auch $S = \{a, c\}$ eine akzeptable Menge.

Eine zulässige Menge gemäß Definition 2.1.5 ist eine konfliktfreie Menge, in der jedes Argument akzeptabel ist. Dies sind u.a. die Mengen $S = \{a\}$, $S = \{a, c\}$ und $S = \{a, c, e\}$.

Eine besondere Bedeutung kommt der bevorzugten Semantik zu, da ein leichtgläubiger Agent eine Extension nach Definition 2.1.6 als mögliche Lösung erachtet. Bereits notierte zulässige Mengen sind $S = \{a\}$, $S = \{a, c\}$ und $S = \{a, c, e\}$. Hierbei ist allerdings nur $S = \{a, c, e\}$ die maximal zulässige Menge, sodass $E = \{a, c, e\}$ die bevorzugte Extension ist. ⌟

Um den Ansatz von Dung gegenüber anderen Ansätzen besser vergleichbar zu gestalten, wird die stabile Semantik eingeführt. Eine stabile Semantik existiert, falls jedes Argument in A, das nicht in S enthalten ist, von S angegriffen wird.

Definition 2.1.7 (stabil [1, Lemma 14])**.**
Sei $AF = (A, R)$ ein Argumentation Framework. Eine konfliktfreie Menge von Argumenten S heißt stabile (*stable*) Extension gdw. $a \in A \setminus S : (S, a)$. Dies ist gegeben, wenn die Argumente in S alle Argumente außerhalb von S angreifen.

Einfach ausgedrückt $S = \{A \mid A$ wird nicht von S angegriffen$\}$ [1, Lemma 14].
Die stabile Extension für Beispiel 2.1.1 auf Seite 6 ist $E = \{a, c\}$ und für das Beispiel 2.1.2 ist es $E = \{a, c, e\}$.
Jede stabile Extension ist zugleich eine bevorzugte Extension, dies gilt jedoch nicht umgekehrt [1, Lemma 15].[6]

Es wird eine charakteristische Funktion eingeführt, um die Fixpunkt Theorie zu nutzen. Ein Fixpunkt liegt hier vor, wenn sich das Ergebnis der charakteristischen Funktion zur vorherigen Eingabe nicht verändert. Dies stellt eine elegante Möglichkeit dar, um später die grundierte Semantik einzuführen.

Definition 2.1.8 (charakteristische Funktion [1, Def. 16])**.**
Die charakteristische Funktion σ eines AF wird definiert als

$$\sigma_{AF} : 2^A \to 2^A,$$
$$\sigma_{AF}(S) = \{A \mid A \text{ ist akzeptabel bzgl. } S\}$$

Das Gegenstück zum Problem einer leichtgläubigen Entscheidung ist das skeptische Entscheidungsproblem. Bei den AFs entspricht dies einer Extension, die sich einer grundierten Semantik zuordnen lässt. Gegeben ist dies durch die folgende Definition:

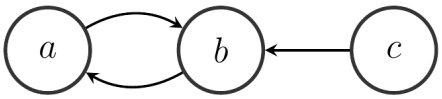
Definition 2.1.9 (grundiert [1, Def. 20])**.**
Sei $AF = (A, R)$ ein Argumentation Framework. Eine grundierte (*grounded*) Extension ist der kleinste Fixpunkt von σ_{AF}.

Der kleinste Fixpunkt ist das erste Ergebnis, bei dem unterschiedliche Eingaben zur selben Auswertung führen. Die Menge S ist folglich zulässig und die minimale Menge bzgl. der Argumentmenge A eines AFs.

[6] vgl. dazu später Tabelle 2.2 auf Seite 14.

Unter Verwendung der charakteristischen Funktion ist für das Beispiel 2.1.1 die grundierte Extension $E = \{a, c\}$. Die entspricht dem kleinsten Fixpunkt wie folgend gezeigt wird:

$\sigma(\emptyset) = \{c\}$,
$\sigma(\{a\}) = \{a, c\}$,
$\sigma(\{a, b\}) = \sigma(\{a\}) = \{a, c\}$.



Wiederholung des Graphen

Für einen skeptischen Agent ergeben die Aussagen von a und c eine plausible Schlussfolgerung.

Das Beispiel 2.1.1 wird abgewandelt, sodass nur die Argumente a und b übrig bleiben, weil der Zeuge seine Aussage zurücknimmt. Die grundierte Semantik ergibt dann folgende Extension $E = \emptyset$. Das Ergebnis ist die leere Menge, denn ein skeptischer Agent erkennt hier nur sich widersprechende Aussagen. Er kann deshalb keine Aussage akzeptieren. Das Ergebnis scheint auf den ersten Blick merkwürdig zu sein, ist unter dem Aspekt eines skeptischen Agenten jedoch plausibel, weil er nur akzeptiert was am wenigsten fragwürdig ist. Es werden also nur die Argumente akzeptiert, deren Annahme er nicht vermeiden kann.

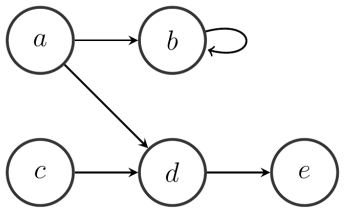
Abschließend wird noch die vollständige Semantik, die eine Verbindung zwischen der leichtgläubigen und skeptischen Extension darstellt, eingeführt [1, S. 329]. Eine vollständige Semantik ist eine zulässige Menge, die nur Argumente aus S verteidigt.

Definition 2.1.10 (vollständig [1, Def. 23])**.**
Sei $AF = (A, R)$ ein Argumentation Framework. Eine zulässige Menge an Argumenten $S \subseteq A$ heißt vollständige (*complete*) Extension gdw. jedes Argument, das akzeptabel ist, auch ein Element von S ist.

Mit anderen Worten: Eine konfliktfreie Menge S ist eine vollständige Extension gdw. $S = \sigma_{AF}(S)$ [1, Lemma 24].

Ein rationaler Agent würde jedes Argument in die Extension aufnehmen, welches er verteidigen kann. Für Beispiel 2.1.1 ist $E_{comp} = \{a, c\}$ und für Beispiel 2.1.2 ist eine konfliktfreie Menge $S = \{a, c, e\}$ unter Verwendung der charakteristischen Funktion $\sigma(S) = \{a, c, e\}$, sodass $E_{comp} = \{a, c, e\}$.

Zusammenfassend ist in Tabelle 2.2 für einen Auszug an Teilmengen $S \subseteq A$ des Beispiels 2.1.2 (der Graph ist nebenstehend nochmals aufgeführt) angegeben, ob es sich um eine konfliktfreie 2.1.3 und eine zulässige 2.1.5 Menge handelt und ob eine bevorzugte 2.1.6, vollständige 2.1.10, stabile 2.1.7 und grundierte 2.1.9 Extension vorliegt.



S	$\sigma(S)$	konfliktfrei	zulässig	vollständig	bevorzugt	stabil	grundiert
$\emptyset$	$\{a, c\}$	✓	✓	✗	✗	✗	✗
$\{a\}$	$\{a, c\}$	✓	✓	✗	✗	✗	✗
$\{a, c\}$	$\{a, c\}$	✓	✓	✓	✓	✗	✓
$\{a, b\}$	$\{a, b, c\}$	✗	✗	✗	✗	✗	✗
$\{a, c, e\}$	$\{a, c, e\}$	✓	✓	✓	✓	✓	✗
$\{a, b, c, e\}$	$\{a, b, c, e\}$	✗	✗	✗	✗	✗	✗
$\{a, b, c, d, e\}$	$\{a, b, c, d, e\}$	✗	✗	✗	✗	✗	✗

Tabelle 2.2: Vorkommende Extensionen für die Mengen S aus Beispiel 2.1.2

Es gibt zwei bevorzugte Extensionen $E_{pref} = \{a, c\}, \{a, c, e\}$ und zwei vollständige Extensionen $E_{comp} = \{a, c\}, \{a, c, e\}$ sowie eine stabile Extension $E_{stab} = \{a, c, e\}$. Es gibt des Weiteren eine grundierte Extension: Das ist $\sigma^1(\emptyset) = \sigma^2(\{a\}) = \{a, c\}$ folglich $E_{grou} = \{a, c\}$. Durch die obige Tabelle 2.2 werden die Zusammenhänge zwischen den Extensionen visualisiert.

2.2 Aussagenlogik

Die ursprüngliche Logik entstammt der Philosophie und untersucht Fragestellungen wie beispielsweise „*Was ist Wahrheit?*“. Die Logik auf dem Gebiet der Informatik ist auf das *Machbare* eines maschinellen Systems ausgerichtet. Ein Teilgebiet ist die Aussagenlogik (AL), die sich mit der formalen Analyse von Aussagen beschäftigt. Dies soll durch das folgende Beispiel verdeutlicht werden.

Beispiel 2.2.1 (Tux)**.**

$A =$ „Tux ist ein Vogel.“
$B =$ „Ein Vogel kann fliegen.“

Im allgemeinen Sprachgebrauch versteht man unter einer Regel eine zusammengesetzte Aussage der Form:

Wenn A gilt, **dann** gilt auch B.

Dabei ist A die Prämisse und B die Konklusion. In der AL sind A und B aussagenlogische Formeln, die jeweils nur aus einer Aussagenvariablen bestehen. Die semantische Bedeutung von A und B entspricht der jeweiligen Aussage. Diesen kann ein Wahrheitswert $\{true, false\}$[7] zugeordnet werden. Die Aussagen werden hier, ohne weitere formale Interpretationsmethoden, intuitiv bejaht, d.h. zu *true* ausgewertet. Es kann die Regel $A \rightarrow B$ aufgestellt werden, die sprachlich formuliert bedeutet

„Wenn Tux ein Vogel ist, dann kann er fliegen."

Diese neue Aussage lässt sich über die Schlussregel *Modus Ponens* ableiten.

$$\frac{\begin{array}{l}\text{A}\\ \text{A} \rightarrow \text{B}\end{array}}{\text{B}}$$

Der sich ergebende Schluss ist: „Tux kann fliegen". Verändert sich die Wissensbasis z.B. durch Hinzunahme neuer Informationen:

C = „Tux ist ein Pinguin."
D = „Ein Pinguin kann nicht fliegen."

können sich neue Schlussfolgerungen ergeben. Beide Aussagen werden wieder intuitiv bejaht und die Regel C → D aufgestellt, die sprachlich formuliert

„Wenn Tux ein Pinguin ist, dann kann er nicht fliegen."

bedeutet. Der über den Modus Ponens gezogene Schluss lautet: „Tux kann nicht fliegen".

[7] später genauer in Definition 2.2.3

Beide Schlüsse stehen offensichtlich im Widerspruch zueinander. Eine einmal bewiesene Schlussfolgerung kann in der klassischen Logik nicht revidiert werden. Diese Art der Wissensvergrößerung ist monoton wachsend und führt ggf. zu falschen Schlüssen.

In dem Beispiel Tux 2.2.1 wurde in einem anfänglichen Schritt aus gegebenem Wissen neues Wissen „Tux kann fliegen“ abgeleitet. Die Ableitung von neuem Wissen erfolgte über die Schlussregel des *Modus Ponens* durch die Aussagen A und B und der Regel $A \rightarrow B$. Die Repräsentation von Wissen wurde mit den Variablen A, B, C, D und dem Symbol $\rightarrow$ („Wenn-Dann-Regel“) vorgenommen. Den Variablen wurden dabei die entsprechenden Aussagen zugewiesen. Es wurde nur der Ausschnitt betrachtet, der durch die Wissensbasis wiedergegeben werden konnte. Die Information, dass es sich bei Tux, um das Linux Maskottchen handelt, folglich um kein reales Tier, wurde nicht gegeben und somit auch nicht betrachtet.

Bei jedem auswertenden System stellen sich im Allgemeinen zwei Fragen:

1. Welche Informationen werden benötigt, damit eine fundierte Schlussfolgerung erfolgen kann (vgl. *Qualifikationsproblem*)?
2. Wie können die zugrundeliegenden Informationen korrekt ausgewertet werden (vgl. *Inferenz*)?

Die Logik bietet einen exakten Formalismus zur Repräsentation und Verarbeitung von Wissen in Form von Axiomen und Theoremen. Die folgenden Abschnitte Syntax 2.2.1, Semantik 2.2.2 und Eigenschaften 2.2.3 behandeln Grundlagen der AL. Für die Erstellung wurde die Arbeit [7] von Beierle und Kern-Isberner und die Arbeit [9] von Schöning verwendet.

2.2.1 Syntax

Zur Repräsentation von Sachverhalten der realen Welt auf einem maschinellen System werden fest definierte Zeichen benutzt. Die Signatur bildet sich nach festen Regeln aus genau diesen Zeichen und wird üblicherweise mit Sigma (Σ) notiert. So gebildete (maschinelle) Objekte besitzen noch keine Assoziation. Aus dem Beispiel Tux 2.2.1 können folgenden Zeichenketten entnommen werden:

$$\Sigma = \{Vogel, Pinguin, fliegen\}.$$

Die Zeichen repräsentieren Objekte der Welt, weshalb meist ein kontextspezifischer Name verwendet wird.

Definition 2.2.1 (Signatur; nach [7, Def. 3.22])**.**
Eine Signatur Σ ist eine nicht-leere Menge aus Namen, die nach bestimmten Regeln aus Zeichen gebildet werden. Die aussagenlogische Signatur Σ ist eine Menge von Bezeichnern (Identifikatoren), genannt Aussagenvariablen.

Eine Signatur stellt die Grundlage dar, um Wissen eines maschinellen Systems aufbauen zu können. Eine Wissensbasis $\mathcal{W}$ ist eine Teilmenge von Formeln über einer gegebenen Signatur Σ, d.h. $\mathcal{W} \subseteq \text{Formeln}(\Sigma)$.

Eine Menge an Formeln ist z.B.

$$a = \top, b = \top, c = \neg d, d = \neg c, e = a \wedge b \wedge (c \vee d) \tag{2.1}$$

Die Aussagenvariablen a, b, c, d, e können schrittweise mittels Junktoren zu komplexeren Formeln zusammengesetzt werden. Die komplexe Formel $a \wedge b \wedge (c \vee d) \rightarrow e$ bedeutet sprachlich formuliert:

„Wenn die Prämissen a,b und c oder d erfüllt sind,
dann folgt die Konklusion e“.

Die Aussagenvariablen können für sich alleine stehen oder wie oben miteinander verknüpft werden. Der Aufbau einer Formel erfolgt wie in Definition 2.2.2 rekursiv mithilfe der Junktoren aus Tabelle 2.3 auf der nächsten Seite.

Definition 2.2.2 (Formeln; nach [7, Def. 3.23])**.**
Für eine aussagenlogische Signatur Σ wird die Menge $Formel(\Sigma)$ der aussagenlogischen Formeln wie folgt gebildet:

- Eine atomare Formel ist eine aussagenlogische Formel, die nur aus einer Aussagenvariablen besteht. Die Menge aller atomaren Formeln wird mit $\mathbb{A}$ bezeichnet.

- Falls A und B aussagenlogische Formeln sind, dann sind auch mit Junktoren verknüpfte Konstrukte aus A und B aussagenlogische Formeln. Die Menge aller Formeln wird mit $\mathbb{F}$ bezeichnet. Es gilt $\mathbb{A} \subseteq \mathbb{F} \subseteq Formel(\Sigma)$.

Es gilt folgende Präzedenz der Junktoren in aufgeführter Reihenfolge: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$. Als Verknüpfungsjunktoren werden insbesondere $\wedge$ und $\vee$ bezeichnet. Sei $A \in \mathbb{A}$ eine atomare Formel, dann sind A (positives Literal) und seine Negation $\neg A$ (negatives Literal) Literale.[8]

Symbol	**Junktor**	**Bedeutung**
$\neg$	Negation	„nicht“
$\wedge$	Konjunktion	„und“
$\vee$	Disjunktion	„oder“
$\rightarrow$	Implikation	„wenn A, dann B“
$\leftrightarrow$	Koimplikation	„A genau dann, wenn B“

Tabelle 2.3: Übersicht der Junktoren

2.2.2 Semantik

Erst durch die Semantik erhält die eben beschriebene Syntaktik eine Bedeutung. Methodisch kann dies durch die Verwendung von Wahrheitstafeln erfolgen. Dafür wird jeder Aussagenvariable ein Boolescher Wert zugewiesen. Beispielhaft ist dies mit der Formel $a \wedge b \wedge (c \vee d) \rightarrow e$ auf Seite 93 anhand einer Wahrheitstafel A.1 erfolgt.

Formeln können beliebig kompliziert miteinander verknüpft werden, daher ist es zielführend, die Formeln in semantisch äquivalente, aber technisch einfachere Formeln zu transformieren. Es gelten die Äquivalenzen aus Tabelle A.2 auf Seite 94. Alle aufgeführten Äquivalenzen können mittels Wahrheitstafeln nachgeprüft werden. Exemplarisch gezeigt wird dies mit der Wahrheitstafel 2.4 für die deMorgansche Regel.

Um eine Verbindung zwischen syntaktischer und semantischer Ebene herzustellen, wird eine Interpretation benötigt. Ob eine Formel *wahr* oder *falsch* ist, hängt von ihrer Interpretation (Belegung) ab. Die Aussagenvariable a repräsentiert z.B. das Objekt *natürliche Person*. Die Aussage a kann entweder wahr oder falsch sein und erhält somit einen Wahrheitswert.

[8] Literal: Siehe Definition A.2.2.

F	G	$F \wedge G$	$\neg(F \wedge G)$	$\neg F \vee \neg G$
0	0	0	1	1
0	1	0	1	1
1	0	0	1	1
1	1	1	0	0

Tabelle 2.4: Nachweis der Äquivalenz mittels Wahrheitstafel für die deMorgansche Regel

Definition 2.2.3 (klassische Wahrheitswerte; [7, S. 34])**.**
Die Wahrheitswerte der klassischen Logik sind zweiwertig (*binär*). Die Menge der Wahrheitswerte wird mit $BOOL = \{true, false\}$ bezeichnet. Alternativ wird auch $\{t, f\}$, $\{T, F\}$ $\{wahr, falsch\}$, $\{w, f\}$ oder $\{1, 0\}$ benutzt.

Dies entspricht der Beantwortung einer „Ja-Nein-Frage“. Es gibt zwei Möglichkeiten einer Aussage genau einen Wahrheitswert zuzuordnen:

Die Aussage ist entweder *true* (wahr) oder die Aussage ist *false* (falsch).

Jeder Aussagenvariablen wird ein Wahrheitswert zugeordnet, diese Zuordnung wird Interpretation I (alternativ v) genannt. Eine Interpretation ist eine Abbildung $I : \mathbb{A} \mapsto BOOL$, die zunächst für eine Teilmenge der atomaren Formeln definiert ist. Um die Interpretation auf die Menge $\mathbb{F}$ erweitern zu können, werden komplexe Formeln zuerst auf ihre bereits definierte atomare Formel reduziert. Danach wird mit Hilfe der Funktion aus Definition 2.2.4 induktiv die komplexe Formel wahrheitsfunktional (vgl. später Definition 2.2.5) ausgewertet, sodass $I : \mathbb{F} \mapsto BOOL$. Die Menge aller Interpretationen unter einer Signatur wird mit $\mathbb{I}_\Sigma$ (alternativ $\mathcal{V}$) bezeichnet.
Eine Interpretation I_1 der Formel aus 2.1 ist:

$$I_1(a) = true \qquad I_1(b) = true \qquad I_1(c) = true$$
$$I_1(d) = false \qquad I_1(e) = true$$

Alternativ ist auch jede andere Zeile der Wahrheitstafel A.1 im Anhang eine mögliche Belegung, z.B. Interpretation I_2:

$$I_2(a) = true \qquad I_2(b) = false \qquad I_2(c) = true$$
$$I_2(d) = true \qquad I_2(e) = false$$

2.2.3 Eigenschaften

Nachdem die Formeln eine Belegung erhalten haben, wird eine Funktion benötigt, die angibt, ob eine Formel unter einer Interpretation wahr oder falsch ist. Bei einer atomaren Formel $F \in \mathbb{A}$ erfolgt dies direkt mittels Wahrheitswertefunktion.

Definition 2.2.4 (Wahrheitswertefunktion [7, Def. 3.8])**.**
In einer klassischen Logik ist für jede Interpretation I eine Wahrheitswertefunktion

$$[\![\_]\!]_I : Formel(\Sigma) \to BOOL$$

definiert. $[\![a]\!]$ ist der Wahrheitswert von a unter der Interpretation I.

Komplexe Formeln $F \in \mathbb{F}$ hingegen müssen inklusive der Junktoren wahrheitsfunktional ausgewertet werden. Dabei wird jeder Junktor durch eine entsprechende Funktion interpretiert, die Wahrheitswerte auf Wahrheitswerte abbildet [7, S. 34]. Dabei gelten die folgenden Bedingungen:

Definition 2.2.5 (wahrheitsfunktional [7, Def. 3.9])**.**
Die Wahrheitswertefunktion $[\![\_]\!]_I$ interpretiert die Junktoren wahrheitsfunktional, wenn gilt:

$$[\![\neg A]\!]_I = \begin{cases} true, & \text{falls } [\![A]\!]_I = false \\ false, & \text{sonst} \end{cases}$$

$$[\![A \wedge B]\!]_I = \begin{cases} true, & \text{falls } [\![A]\!]_I = true \text{ und } [\![B]\!]_I = true \\ false, & \text{sonst} \end{cases}$$

$$[\![A \vee B]\!]_I = \begin{cases} true, & \text{falls } [\![A]\!]_I = true \text{ oder } [\![B]\!]_I = true \\ false, & \text{sonst} \end{cases}$$

$$[\![A \to B]\!]_I = \begin{cases} true, & \text{falls } [\![\neg A]\!]_I = true \text{ oder } [\![B]\!]_I = true \\ false, & \text{sonst} \end{cases}$$

$$[\![A \leftrightarrow B]\!]_I = \begin{cases} true, & \text{falls } [\![A]\!]_I = [\![B]\!]_I \\ false, & \text{sonst} \end{cases}$$

Eine Interpretation I erfüllt eine aussagenlogische Formel F genau dann, wenn ihr Wahrheitswert in I zu *true* ausgewertet wird [7, S. 35]. Die Erfüllungsrelation kann nunmehr mit Hilfe der Wahrheitswertefunktion definiert werden:

Definition 2.2.6 (Erfüllungsrelation [7, Def. 3.10])**.**
Für eine Interpretation $I \in \mathbb{I}_\Sigma$ und eine Formel $F \in Formel(\Sigma)$ gilt:

$$I \models_\Sigma F \text{ gdw. } [\![F]\!]_I = true$$

Ist eine Formel unter einer Interpretation erfüllt, so ist I ein Modell von F.

Definition 2.2.7 (Modell; nach [7, Def. 3.11])**.**
Sei $F \in Formel(\Sigma)$ eine Formel und $I \in \mathbb{I}_\Sigma$ eine Interpretation. Wertet die Erfüllungsrelation F unter I zu *true* aus, dann ist I ein Modell von F, geschrieben $I \models_\Sigma F$. Analog gilt dies auch für Formelmengen FM. Dabei gilt $I \models_\Sigma FM$ *gdw.* $I \models_\Sigma F$ für jedes $F \in FM$.

Obige Interpretation I_1 ist ein Modell der folgenden Formel:

$$a \wedge b \wedge (c \vee d) \rightarrow e.$$

Komplexe Formeln werden häufig in eine (semantisch) äquivalente, aber technisch einfachere Formel überführt. Die Bedingung der folgenden Definition muss dabei erfüllt sein:

Definition 2.2.8 (semantische Äquivalenz; [7, Def. 3.33])**.**
Zwei Formeln F und G sind semantisch äquivalent, falls für alle Interpretationen I gilt: $[\![F]\!]_I \equiv [\![G]\!]_I$.

Zum Beispiel sind die Formeln $\neg(F \wedge G) \equiv \neg F \vee \neg G$ aus Tabelle 2.4 auf Seite 19 semantisch äquivalent, da sie unter jeder Interpretation gleich ausgewertet werden.

Anmerkung.
Eine besondere Art von Formeln sind solche, für die alle möglichen Interpretationen wahr sind. Zum Beispiel ist jede Formel $A \vee \neg A$ in allen möglichen Interpretationen wahr. Derartige Formeln heißen allgemeingültig bzw. Tautologie $\top$. [7, S. 35]
Die Kontradiktion $\bot$ ist dazu komplementär.

2.3 Abstrakt dialektisches Framework

Mit den abstrakt dialektischen Frameworks (ADFs) stellen Brewka und Woltran in [2], [10] einen Ansatz vor, durch den die ursprünglichen AFs ausdrucksstärker und besser modellierbar werden.[9]

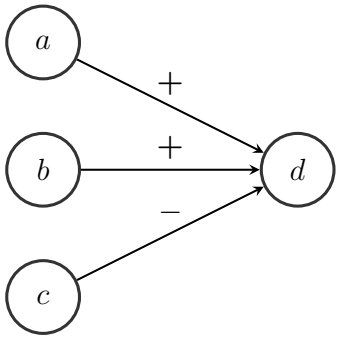


Abbildung 2.3: Der Graph stellt das Statement d dar, das zwei Unterstützer (+) und einen Angreifer (−) hat. Dies lässt sich im ordinären Argumentationsgraph eines AFs nicht darstellen. Der Graph ist aus [12, Seite 239] entnommen.

Eine Motivation für die ADFs soll anhand des Beispiels Unfallsituation 2.1.1 von Seite 6 gegeben werden. Zur Erinnerung: Durch die vorhandenen Aussagen konnte die Verkehrssituation rekonstruiert werden, um auf den Unfallverursacher zu schließen. Jede Aussage konnte direkt als Argument in das erstellte AF übernommen und in Relation zu den anderen Argumenten gesetzt werden. Die „stärksten“ bzw. sich am besten verteidigenden Argumente setzten sich dabei durch. Diese Form der Argumentation bei den AFs ist statisch. Damit ist gemeint, dass die Argumentation nur auf eine Art ablaufen kann, nämlich dass ein Argument stets ein anderes angreift. Grundsätzlich gibt es aber nicht nur Argumente, die einander angreifen, sondern auch Argumente *für* eine und Argumente *wider* eine These. Diese Motivation soll durch die Fortsetzung des Beispiels der Unfallsituation verdeutlicht werden.

Beispiel 2.3.1. (Fortsetzung)
Das Beispiel 2.1.1 wird fortgesetzt, indem es um eine weitere Aussage ergänzt wird.

Zeuge 2: „Der Beteiligte 1 hatte tatsächlich grün.“

[9] Eine Aufzählung weiterer Ansätze zur Anpassung der AFs findet sich in [11].

Diese Aussage stützt das Argument des Beteiligten 1. Es handelt sich folglich um ein unterstützendes Argument (Proargument). ⌟

Es ist erkennbar, dass ein besser an die Argumentationssituation angepasstes Framework benötigt wird. Die Beziehungen (Relationen) zwischen den Argumenten sind offensichtlich nicht immer Angriffe. Ein solches Framework wird mit den ADFs angeboten, indem sie die AFs durch Kombination mit der AL verallgemeinern. Insbesondere heißt dies, dass eine Relation zwischen zwei Statements (vormals Argumente) allgemeiner gesehen werden kann, z.B. auch als Unterstützung. Um das zu erreichen wird jedem Statement eine Akzeptanzbedingung zugewiesen, die die Beziehung der Statements konkretisiert.

Die Idee eines ADFs nach [12, S.239] ist in Abbildung 2.3 illustriert und wird hier weiter ausgeführt. Wie muss die Akzeptanzbedingung von d ausgestaltet sein, damit d unter spezifizierten Bedingungen akzeptiert ist? Es kommen nachfolgend mehrere Möglichkeiten in Betracht, die zeigen wie fein Akzeptanz(en) in einem ADF ausgeprägt sein können.

$\neg c \wedge (a \wedge b)$	kein Angreifer und alle Unterstützer
$\neg c \wedge (a \vee b)$	kein Angreifer und mindestens ein Unterstützer
$\neg c \vee (a \wedge b)$	kein Angreifer oder alle Unterstützer
$\neg c \vee (a \vee b)$	kein Angreifer oder mindestens ein Unterstützer
$(\neg c \wedge (a \wedge b)) \vee (a \wedge b)$	mehr Unterstützer als Angreifer

2.3.1 Grundprinzip

Eine Erweiterung bzw. Verallgemeinerung des AFs ist das ADF. Dieses Framework erlaubt verschiedene Relationen zwischen den Statements, wodurch ADFs spezifischer in einer Domäne eingesetzt werden können. Auch bei ADFs erfolgt die technische Darstellung mittels gerichteter Graphen, wobei Statements als Knoten und Relationen zwischen diesen als gerichtete Kanten (Pfeile) dargestellt werden. Die Pfeile werden nicht mehr nur als Angriffe, sondern als spezifizierte Relationen der Statements zu ihren Vorgängern bzw. Eltern (*parents*) betrachtet. Dadurch sind Relationen nicht notwendigerweise Angriffe, sondern stellen abstrakte Beziehungsgeflechte dar. Die gerichteten Kanten werden als *Links* bezeichnet. Die genaue Beziehung zwischen zwei

Statements, wobei ein Statement Vorgänger des anderen ist, wird durch eine Akzeptanzbedingung festgelegt.[10]

Beispiel 2.3.2.
Es sei folgendes ADF nach Definition 2.3.1 auf der nächsten Seite gegeben:
$DF = (\{a, b, c\}, \{(a, c), (a, b), (b, c), (b, a)\}, \{\mathcal{C}_a = \neg b, \mathcal{C}_b = \neg a, \mathcal{C}_c = \neg a \vee b\})$
Der dazu korrespondiere Argumentationsgraph ist in Abbildung 2.4 dargestellt.

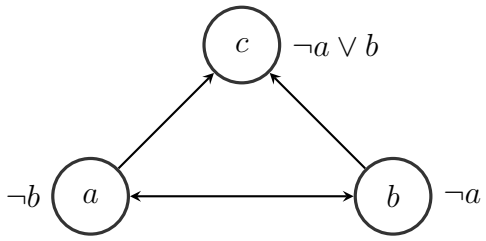


Abbildung 2.4: Argumentationsgraph eines einfachen ADFs

Die Kanten stellen ein Beziehungsgeflecht unter den Statements dar. Eine Kante bzw. Link von a zu c bedeutet, dass a der Vorgänger von c ist. Die konkrete Beziehung zwischen diesen beiden Statements wird erst durch die jeweilige Akzeptanzbedingung C_s gegeben, wobei $s \in S$ ist. In dem Graphen wird dies durch eine aussagenlogische Formel neben dem jeweiligen Knoten dargestellt. Diese Bedingung gibt an, ob ein Statement akzeptiert ist oder nicht.
Das Statement a kann nur akzeptiert werden, wenn b nicht akzeptiert wird und umgekehrt. Die Links[11] zwischen (a, b) bzw. (b, a) können hier als Angriffe interpretiert werden. In diesem Fall ist a nur akzeptiert, wenn b nicht akzeptiert wird und andersrum. Wohingegen c akzeptiert wird, wenn a nicht oder b akzeptiert wird. ⌟

Auch mit den ADFs lassen sich Argumentationen jeglicher Art darstellen. Die Akzeptanz eines Statements s hängt dabei von der Relation zu den Eltern (-knoten) $par(s)$ ab. Wie in der AL gibt es auch hier eine Funktion, die den Wahrheitswert eines Statements auswertet. Dies geschieht anhand der Akzeptanzfunktion $\mathcal{C}$ über den Status der Eltern des Statements s. Alle Statements stammen ebenfalls aus einem beliebig großen, aber endlichen Universum $\mathcal{U}$.

[10] Ausnahme ist eine Schleife (s, s) bei der das Statements selbst auch Vorgänger ist.
[11] Für die Darstellung der Links (a, b) und (b, a) wird verkürzend auch der Doppelpfeil $\leftrightarrow$ benutzt.

Definition 2.3.1 (Abstrakt Dialektisches Framework aus [10, Def. 1])**.**
Ein ADF ist ein Tripel $DF = (S, L, \mathcal{C})$, wobei

$S \subseteq \mathcal{U}$ ist eine endliche Menge an abstrakten Statements (Argumenten, Knoten),

$L \subseteq S \times S$ ist eine Menge an Links und

$\mathcal{C} = \{\mathcal{C}_s\}_{s \in S}$ ist eine Menge von totalen Funktionen $\mathcal{C}_s : 2^{par(s)} \to \{t, f\}$, für jedes Statement s. C_s wird Akzeptanzbedingung von s genannt.

Damit Semantiken eines Frameworks nicht manuell ausgewertet werden müssen, kommt auch hier die Fixpunkt Theorie zur Anwendung. Konkret wird die *Approximation Fixpoint Theory* aus [13] genutzt.
Ein Gitter ist eine partiell geordnete Menge $\mathcal{G} = (S, P)$, die für zwei Elemente $\{a, b\} \in S$ die folgenden Bedingungen unter der Ordnung P erfüllt:

$$\forall a, b \in S : sup(a, b) \neq \emptyset^{12} \text{ und } \forall a, b \in S : inf(a, b) \neq \emptyset^{13} \tag{2.2}$$

Die zweiwertige Logik kann entsprechend ihres Wahrheitswerts geordnet werden. Sei $\mathcal{G} = (S_1, P_1)$ ein Gitter, mit $S_1 = \{\mathcal{V} : \{a, b\} \mapsto BOOL\}$ und $P_1 = \leq_t$. Dann bezeichnet $\mathcal{V}$ die Menge aller zweiwertigen Interpretationen über $\{a, b\}$ und $\leq_t$ die Ordnung nach dem Wahrheitswert $f < t$. Des Weiteren seien x, y zwei Interpretationen aus $\mathcal{V}$, so haben diese dasselbe Infimum und Supremum.

$$x := \{a \mapsto t, b \mapsto f\}, \quad y := \{a \mapsto f, b \mapsto t\}$$
$$sup(a \mapsto t, b \mapsto t), \quad inf(a \mapsto f, b \mapsto f)$$

Eine wichtige Eigenschaft eines Gitters ist die Möglichkeit, ein Element zu approximieren (Approximationsgleichung).

$$\begin{gathered}\text{Ein Element } c \in S_1 \text{ wird durch ein Paar } (a, b) \in \mathcal{G} \text{ approximiert,} \\ \text{wenn gilt: } a \leq c \leq b.\end{gathered} \tag{2.3}$$

[13] Supremum (join) – kleinste obere Schranke
[13] Infimum (meet) – größte unterste Schranke

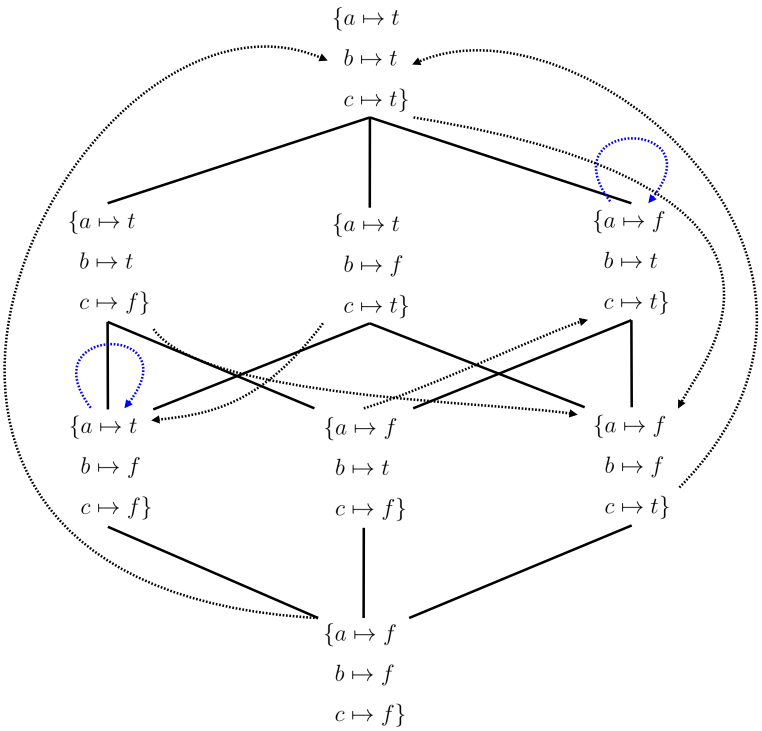


Abbildung 2.5: Das Hassediagramm (bzw. Gitter $\mathcal{G}$) aller zweiwertigen Interpretationen des Beispiels 2.3.2 auf Seite 24. Die aktualisierten Zuweisungen des G-Operators sind durch gestrichelte Pfeile dargestellt. Benachbarte Elemente des Gitters sind durch Linien miteinander verbunden. Der Operator hat genau zwei blau markierte Fixpunkte.

Die Statements eines ADFs können mit einer zweiwertigen Interpretation $v \in \mathcal{V}_2$ belegt werden. Die zugrundeliegende Wissensbasis $\mathcal{W}$ enthält die Akzeptanzbedingungen $\mathcal{C}_s$ für alle $s \in S$. Die Auswertung kann mit einer charakteristischen Funktion erfolgen. Grundsätzlich kann dies die Akzeptanzfunktion $\mathcal{C}$, oder ein anderer Operator übernehmen. In folgender Definition wird der Operator G als Funktion eingeführt, der die Auswertung übernimmt.

Definition 2.3.2 (Operator G [12, Def. 3.3])**.**
Sei $DF = (S, L, \mathcal{C})$ ein ADF. Der Operator $G_{DF} : \mathcal{V}_2 \to \mathcal{V}_2$ bekommt als Eingabe eine zweiwertige Interpretation $v \in \mathcal{V}_2 : S \to \{t, f\}$ und gibt eine mittels Akzeptanzbedingung aktualisierte Interpretation zurück.

$$G_{DF}(v) \ : \ S \to \{t, f\} \text{ mit } s \mapsto v(\mathcal{C}_s)$$

Die Funktionsweise ist in Abbildung 2.5 auf der vorherigen Seite dargestellt. Der Operator erhält die Interpretationen als Eingabe und gibt den aktualisierten Status, durch Auswertung der Akzeptanzbedingung der Statements, zurück. Beispielsweise wird $G_{DF}(\{a \mapsto f, b \mapsto f, c \mapsto f\})$ ausgewertet zu $\{a \mapsto t, b \mapsto t, c \mapsto t\}$.

Wie in der AL können den Statements über eine Interpretation Werte zugeordnet werden. Dabei ist $v : S \mapsto BOOL$ eine zweiwertige Interpretation über S, die jedem Statement den Wert $true$ oder $false$ zuordnet.

$$\begin{gathered}\text{Ein ADF besitzt ein zweiwertiges Modell gdw.} \\ v(s) = v(\mathcal{C}_s) \text{ für jedes } s \in S \text{ gilt.}\end{gathered} \tag{2.4}$$

Das heißt, wenn jedes Statement denselben Wahrheitswert wie seine Akzeptanzfunktion hat.

Beispiel. (Fortsetzung)
Den Aussagenvariablen a, b, c aus Beispiel 2.3.2 auf Seite 24 werden die zweiwertigen Interpretationen $\mathcal{V}_2$ aus Tabelle 2.5 zugeordnet. Die Akzeptanzbedingungen der Statements werden durch den Operator G aus Definition 2.3.2 ausgewertet. Die Abbildung 2.5 und die Tabelle 2.5 beschreiben denselben Auswertungsprozess durch G.

a	b	c	$G(\mathcal{C}_a)$	$G(\mathcal{C}_b)$	$G(\mathcal{C}_c)$
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	0	1	1
0	**1**	**1**	**0**	**1**	**1**
1	**0**	**0**	**1**	**0**	**0**
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	0	0	1

Tabelle 2.5: Auf der linken Seite der Tabelle werden den Aussagenvariablen (Statements) Boolesche Werte zugeordnet und auf der rechten Seite die Werte nach Auswertung der Akzeptanzbedingungen $\mathcal{C}_s$ durch den Operator G. Die fett gedruckten Zeilen entsprechen den Fixpunkten des Operators.

⌟

Betrachtet wird nun eine dreiwertige (ternäre) Logik mit den Werten *true* (t), $false$ (f) und $undefined$ (u)[14]. Eine solche Logik mit allen Interpretationen $\mathcal{V}_3 = \{v : S \to \{t, f, u\}\}$ kann nach ihrem Informationsgehalt $\leq_i$ geordnet werden, wobei $u <_i t$, $u <_i f$ gilt und $\leq_i$ der reflexiv transitive Abschluss von $<_i$ ist. Die partiell geordnete Menge $(\mathcal{V}_3, \leq_i)$ bildet das Gitter $\mathcal{G}_3$.[15] Das niedrigste Element (nach der Ordnung) dieses Gitters ist die Interpretation $v_u : S \to \{u\}$, die allen Statements den Wert u zuweist.

Um eine Formel wie in der AL „wahrheitsfunktional“ auszuwerten, wird der Operator Π_i eingeführt. Der Operator Π_i findet die größte untere Schranke (Infimum) zweier Elemente des Gitters $\mathcal{G}_3$.

Definition 2.3.3 (Konsens Operator; *meet operation* aus [10, S. 804])**.**

$$\Pi_i \quad = \quad \begin{cases} true, & \text{falls } true\ \Pi_i\ true \\ false, & \text{falls } false\ \Pi_i\ false \\ undefined, & \text{sonst} \end{cases}$$

Die $\leq_i$-maximalen Elemente des Gitters $\mathcal{G}_3$ sind genau die zweiwertigen Interpretationen $\mathcal{V}_2$, die jedoch kein gemeinsames Supremum haben. Auch für $\mathcal{G}_3$ gilt, dass ein Element des Gitters durch ein Paar desselben Gitters approximiert werden kann (siehe Approximationsgleichung 2.3). Zum Beispiel erzeugt die dreiwertige Interpretation $v : \{a \mapsto u, b \mapsto t\}$ für die Menge $\{a, b\}$ die Approximationsmenge $\{w_1, w_2\}$. Die Menge w_1 besitzt die klassisch-logischen Wahrheitswerte mit

$$w_1 = \{\{a \mapsto t, b \mapsto t\}, \{a \mapsto f, b \mapsto t\}\}$$

und die Menge w_2 enthält die nicht-klassischen Werte

$$w_2 = \{a \mapsto u, b \mapsto u\}.$$

[14] alternativ: *unbekannt*, *unsicher*, u, $\frac{1}{2}$

[15] Wenn aus dem Kontext ersichtlich ist, welches Gitter gemeint ist, wird der Index weggelassen.

Das Gitter erfüllt mangels Supremum nicht die Voraussetzung der Gleichung 2.2 – konkret liegt ein unteres Halbgitter (*meet-semilattice*) vor.[16] Als Approximation des Operators G wird der Operator Γ nach [12, Corollary 3.6] eingeführt.

Definition 2.3.4 (Operator Γ [12, Corollary 3.6])**.**
Sei $DF = (S, L, \mathcal{C})$ ein ADF. Der Operator $\Gamma_{DF} : \mathcal{V}_3 \to \mathcal{V}_3$ ist die absolute Approximation von G_{DF}. Für ein gegebenes DF und eine Interpretation $v \in \mathcal{V}_3$ erzeugt der Operator $\Gamma_{DF}(v)$ eine aktualisierte Interpretation wie folgt:

$$\Gamma_{DF}(v) \ : \ S \to \{t, f, u\} \text{ mit } s \mapsto \Pi_i\{w(\mathcal{C}_s) \mid w \in [v_2]\}$$

Mit anderen Worten: Der Γ-Operator liefert für jedes Statement $s \in S$ den durch Π_i erzeugten (nicht-klassischen) Wahrheitswert der jeweiligen Akzeptanzbedingung $\mathcal{C}_s$ unter einer dreiwertigen Interpretation zurück.

$$\Gamma_{DF}(v \in \mathcal{V}_3) : S \to \{t, f, u\} \text{ mit } s \mapsto v(s) \ \Pi_i \ v(\mathcal{C}_s) \text{ für jedes } s \in S.$$

2.3.2 Semantiken der ADFs

Die Semantiken der ADFs werden mithilfe der Fixpunkt Theorie (AFT) bestimmt. Dies erfolgt mittels dreiwertiger Logik $\{true, false, undefined\}$ und des Γ-Operators aus Definition 2.3.4. Die Semantiken lassen sich über den Fixpunkt des Operators Γ bestimmen, ihre Definitionen sind aus [12, Definition 3.8] übernommen.

Definition 2.3.5 (Semantiken)**.**
Sei $DF = (S, L, \mathcal{C})$ ein ADF und $v : S \to \{t, f, u\}$ eine Interpretation.

1. v ist vollständig für DF gdw. $v = \Gamma_{DF}(v)$.
2. v ist zulässig für DF gdw. $v \leq_i \Gamma_{DF}(v)$.
3. v ist bevorzugt für DF gdw. v ist $\leq_i$-maximal zulässig.
4. v ist grundiert für DF gdw. v ist der $\leq_i$-kleinste Fixpunkt von Γ_{DF}.

[16] Wie die Approximation eines unsicheren Statements gem. Gleichung 2.3 zufriedenstellend gelingt, wird später in Abschnitt 5.3.1 erklärt und mittels Abbildung 5.5 gezeigt.

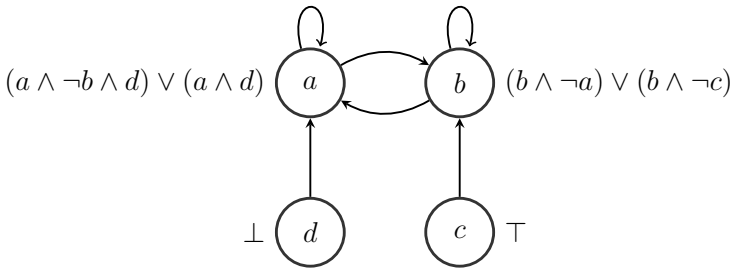


Abbildung 2.6: Die Aussagen der Verkehrsteilnehmer als Argumentationsgraph eines ADFs.

Beispiel. (Adaption: Unfallsituation)
Das Beispiel 2.3.1 wird dahingehend verändert, dass ein Versicherungsbeauftragter den Fall begutachtet. Die Aussagen der beiden Beteiligten greifen sich gegenseitig an.[17] Des Weiteren ist davon auszugehen, dass sich die Aussagen der Beteiligten in gewisser Weise selbst stützen. Zeuge 1 unterstützt mit seiner Aussage den Beteiligten 1 und Zeuge 2 widerspricht der Aussage des Beteiligten 2. Letztere Aussage wird der Aussagenvariable d zugeordnet. Die Akzeptanzbedingung der Statements eines ADFs können beliebig ausgestaltet werden. Beispielsweise ergibt sich folgendes ADF:

$$\begin{aligned} DF = (&\{a, b, c, d\}, \{(a, a), (a, b), (b, b), (b, a), (c, b), (d, a)\}, \\ &\{\mathcal{C}_a = (a \wedge \neg b \wedge d) \vee (a \wedge d), \mathcal{C}_b = (b \wedge \neg a) \vee (b \wedge \neg c), \\ &\mathcal{C}_c = \top, \mathcal{C}_d = \bot\}) \end{aligned}$$

Die Akzeptanzbedingung von a bedeutet, dass a akzeptiert ist, wenn es mehr Aussagen gibt, die für diese Aussage sprechen als dagegen oder wenn die Aussagen a und d sich decken. Die Bedingung von b kann wie folgt interpretiert werden, b ist akzeptiert, wenn die Aussage a oder c nicht akzeptiert ist. Die Aussagen beider Beteiligten a und b sind selbst-stützend, d.h. sie müssen wahr sein, um akzeptiert zu werden. Bei dem Zeugen 1 handelt es sich um einen fremden Dritten, d.h., dass die Aussage c immer als wahr angenommen bzw. zu *true* ausgewertet wird. Wohingegen der Zeuge 2, als Beifahrer, direkt in den Unfall involviert ist. Deshalb kann die Aussage d für die

[17] Um ein AF in ein ADF zu transformieren, sind alle eingehenden Kanten (Angriffe) bei der Akzeptanzbedingung als negierte Literale zu konjugieren (vgl. [10, S. 803]).

Versicherung nicht berücksichtigt werden und wird stets zu $false$ ausgewertet. Der korrespondierende Graph ist in Abbildung 2.6 dargestellt.

Um die Extensionen dieses ADFs zu erhalten, werden die Fixpunkte via Γ-Operator ermittelt. Die Fixpunkte sind in Tabelle 2.6 zusammengefasst, zusätzlich ist das vollständige Gitter inklusive der Übergänge des Γ-Operators im Anhang A.3 zu finden.[18]

	a	b	c	d	$\Gamma(\mathcal{C}_a)$	$\Gamma(\mathcal{C}_b)$	$\Gamma(\mathcal{C}_c)$	$\Gamma(\mathcal{C}_d)$
v_1	u	u	w	f	u	u	w	f
v_2	u	f	w	f	u	f	w	f
v_3	f	u	w	f	f	u	w	f
v_4	f	f	w	f	f	f	w	f
v_5	f	w	w	f	f	w	w	f

Tabelle 2.6: Auf der linken Seite sind die dreiwertigen Interpretationen v_1-v_5 und auf der rechten Seite die Werte des Γ-Operators wiedergegeben.

Erzeugt die Aktualisierung der Interpretation durch den Γ-Operator keinen Booleschen Wert, so enthält die Auswertung ein Ergebnis mit nicht-klassischen Wahrheitswerten. Es kommen folgende Extensionen vor, die sich den Semantiken gemäß Definition 2.3.5 zuordnen lassen:

Nr.1: vollständige Semantiken v_4, v_5 und v_1 sowie v_2 und v_3, da $v = \Gamma_{DF}(v)$. Somit ist $E_{comp} = \{c\}, \{b, c\}$.

Nr.2: zulässige Semantiken sind die Interpretationen v_1 bis v_5. Also $E_{admi} = \{c\}, \{b, c\}$.

Nr.3: bevorzugte Semantiken mit v_4 und v_5, da es sich um $\leq_i$-maximale Fixpunkte handelt. Die Extensionen sind $E_{pref} = \{c\}, \{b, c\}$.

Nr.4: grundierte Semantik, da v_1 der $\leq_i$-kleinster Fixpunkt von Γ_{DF} ist. Die Extension ist $E_{grou} = \{c\}$.

[18] Die Übersichtlichkeit geht aufgrund des exponentiellen Wachstums $3^{|S|}$ schnell verloren, deshalb ist das Gitter in den Anhang ausgelagert.

Gemäß [10, Theorem 1] ist jede bevorzugte Extension auch eine vollständige Extension und die grundierte ist die $\leq_i$-kleinste Extension. Je nach Art des verwendeten Agenten kann ein Versicherungsgutachter unterschiedliche Schlussfolgerungen ziehen.

Ein skeptischer Agent nimmt die Interpretation $v_1 := \{a \mapsto u, b \mapsto u, c \mapsto w, d \mapsto f\}$, die genau der grundierten Extension entspricht. Die Aussagen der Unfallbeteiligten a und b könnten sowohl wahr oder falsch sein. Für den skeptischen Agenten sind alle Statements (Aussagen) relevant, die er verteidigen kann. Da die Aussagen c und d nicht angegriffen werden, stützt er sein Ergebnis auf eben diese Aussagen. Für ihn ist der Beteiligte 2 der Unfallverursacher. Schließlich besagt die Aussage c, dass der Beteiligte 2 lediglich einen festen Grünpfeil ohne Lichtsignal hatte und die Aussage d kann aufgrund von Befangenheit nicht gewertet werden.

Ein leichtgläubiger Agent wählt hingegen die bevorzugten Extensionen $v_4 := \{a \mapsto f, b \mapsto f, c \mapsto w, d \mapsto f\}$ bzw. $v_5 := \{a \mapsto f, b \mapsto w, c \mapsto w, d \mapsto f\}$ aus. Alles was der Agent irgendwie verteidigen kann, ist für ihn plausibel. Für ihn haben die beiden Beteiligten gleichermaßen Schuld (Interpretation v_4) oder nur der Beteiligte 1 (Interpretation v_5). Diese Interpretationen sind zugleich Modelle des ADFs, da den Aussagen nur Boolsche Werte zugeordnet sind (vgl. Gleichung 2.4). ⌟

3 Problemanalyse

Unvollständige Information stellt KI-Systeme vor Probleme. Die Systeme bearbeiten den Ausschnitt einer „Welt“ in Form von einer ihnen gegebenen Wissensbasis. In der Auswahl dieses Ausschnitts liegt bereits eine gewisse Unvollständigkeit begründet (vgl. *Qualifikationsproblem*), auf die hier nicht konkreter eingegangen wird. Unvollständige Information kann viele Ursachen wie bspw. Programmfehler, fehlerhafte Aktualisierungen von Informationen, inkonsistente zugrundeliegende Daten uvm. haben.
Im konkreten Kontext einer Argumentation können Argumente hinzukommen, überholt sein oder sich ändern. Diese *Argumentationsdynamik* erfordert ein Framework, das mit unvollständiger Information umgehen kann. Des Weiteren können Informationen unsicher sein, weil z.B. das Vorhandensein eines Arguments unklar ist.
In diesem Abschnitt werden die Schwierigkeiten betrachtet, die auftreten, wenn ein ADF mit unvollständiger Information arbeiten soll.

Zwischen den Komponenten eines ADFs herrscht gemäß Definition 2.3.1 eine direkte Abhängigkeit. Eine Folge ist, dass Unvollständigkeit an einer Komponente eine Auswirkung an einer anderen verursacht. Unvollständige Information kann hierbei aus unterschiedlichen Richtungen betrachtet werden:

1. Elemente fehlen ganz oder sind an mindestens einer Komponente unvollständig. In diesem Fall sind die Informationen eines ADFs nicht vollständig und müssen diesem zur Konstruktion des Argumentationsgraphen *hinzugefügt* werden. Die fehlenden Informationen müssen sich aus den gegebenen Informationen wieder herstellen lassen (ableiten).

2. Aufgrund der zugrundeliegenden Daten kann die Existenz von Elementen nicht garantiert werden. Das ADF liegt grundsätzlich vollständig vor, jedoch sind die vorhandenen Daten nicht ausreichend, um eine Existenz von Argumenten oder Angriffen tatsächlich zu garantieren. Die fehlende Information führt zu einer Unsicherheit, sodass die entsprechenden Elemente besonders behandelt werden müssen.

Ersteres wird in Abschnitt 3.1 und letzteres in Abschnitt 3.2 untersucht.

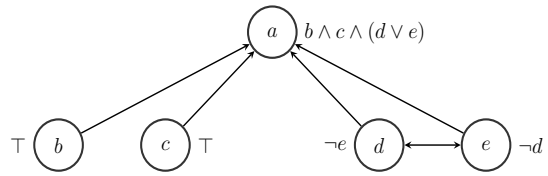
3.1 Unvollständigkeit an Komponenten

Unvollständige Information führt dazu, dass ein Argumentationsgraph nicht konstruiert werden kann. In diesem Abschnitt wird unvollständige Information aus dem Blickwinkel betrachtet, dass Elemente einer Komponente unvollständig sind. Um die Schwierigkeiten möglichst detailliert zu untersuchen, sei folgendes Beispiel gegeben:

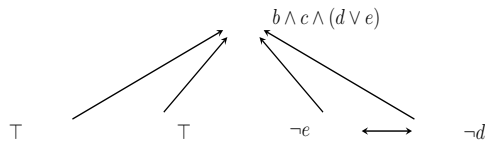
Beispiel 3.1.

$$\begin{aligned} DF =&(\{a,b,c,d,e\},\\ &\{(b,a),(c,a),(d,a),(e,a),(d,e),(e,d)\},\\ &\{\mathcal{C}_a = b \wedge c \wedge (d \vee e), \mathcal{C}_b = \top, \mathcal{C}_c = \top, \mathcal{C}_d = \neg e, \mathcal{C}_e = \neg d\}) \end{aligned}$$ ⌟

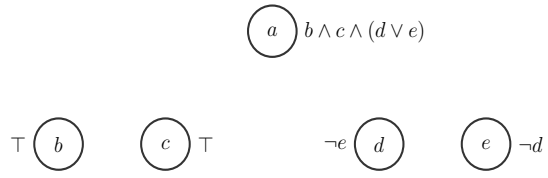
Der korrespondierende Argumentationsgraph ist in Abbildung 3.1a dargestellt. Gemäß der Definition 2.3.1 ist ein ADF ein Tripel $DF = (S, L, \mathcal{C})$ mit den Komponenten *Statements*, *Links* und *Akzeptanzfunktion*. Unvollständige Information kann an den einzelnen Komponenten oder in deren Kombination auftreten. Die Abbildung 3.1 gibt einen Überblick darüber, welche Auswirkung Unvollständigkeit auf die Konstruktion des Argumentationsgraphen im Allgemeinen hat.



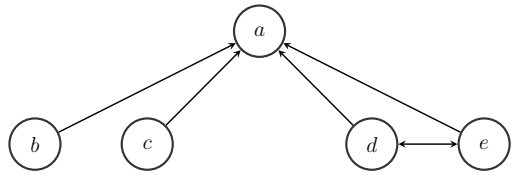
(a) vollständiger Argumentationsgraph



(b) unvollständige Statements



(c) unvollständige Links



(d) unvollständige Akzeptanzbedingungen

Abbildung 3.1: Ein ADF mit Unvollständigkeiten an verschiedenen Komponenten, verursacht durch unvollständige Information.

Die Unvollständigkeit an einer einzigen Komponente führt jeweils zu einer eigenständigen Fallkonstellation (Grundfall). Die Kombination aus verschiedenen Unvollständigkeiten an mehreren Komponenten stellt einen Sonderfall dar. Im Folgenden werden insbesondere die drei Grundfälle betrachtet.

Anmerkung. Die maschinelle Darstellung von ADFs erfolgt durch Datenstrukturen wie beispielsweise Arrays und Listen. Die Frameworks werden technisch als Graphen dargestellt, deshalb wird auf algorithmischer Ebene das entsprechende graphentheoretische Vokabular verwendet. Dabei werden Knoten als Synonym für Statements und (gerichtete) Kanten bzw. Pfeile als Synonym für Relationen genutzt.

3.1.1 Statements

Statements werden im Argumentationsgraph als Knoten abgebildet. Ein Knoten ist mindestens der Anfangs- bzw. Endpunkt einer Kante, oder bei einer Schleife[19] auch beides zugleich. Außerdem sind Knoten ohne Kante möglich, wodurch das entsprechende Statement keinen Bezug (Relation) zur übrigen Argumention hat.[20] Deshalb wird dieser theoretische Fall nicht weiter betrachtet. Sind Elemente aus der Menge der Statement S nicht vorhanden, kann der Knoten im Graphen nicht abgebildet werden. Dadurch entsteht eine Lücke, die entweder offensichtlich ist oder unbemerkt[20] bleibt. Fakt ist, dass dadurch eine Unvollständigkeit vorliegt, die zu einer nicht korrekten Darstellung des Graphen führt. Streng genommen dürfen Kanten nicht ohne Knoten beginnen bzw. enden, insofern dient die Abbildung 3.1b hauptsächlich einer vorteilhaften Darstellung des Problems. Gemäß der Definition 2.3.1 sind die Statements $s \in S$ sowohl die Voraussetzung für die Links L als auch für die Akzeptanzbedingungen $\mathcal{C}_s$. Hier zeigt sich die Abhängigkeit zwischen den Komponenten eines ADFs besonders deutlich.

Fraglich ist, ob fehlende Information aus den gegebenen Komponenten abgeleitet werden kann. Zu Abbildung 3.1b korrespondiert das folgende ADF:

[19] $(s, s) \in L, s \in S$; siehe auch Knoten c in Abbildung A.1
[20] bspw. Knoten c des nicht zusammenhängenden Graphen in A.1

$$
\begin{aligned}
DF = (\ & \emptyset, \\
& \{(b,a),(c,a),(d,a),(e,a),(d,e),(e,d)\}, \\
& \{\mathcal{C}_a = b \wedge c \wedge (d \vee e), \mathcal{C}_b = \top, \mathcal{C}_c = \top, \mathcal{C}_d = \neg e, \mathcal{C}_e = \neg d\})
\end{aligned}
$$

Es kommen zwei Möglichkeiten in Betracht, um die fehlenden Statements künstlich zu generieren:

1. aus den vorhandenen Informationen der Links oder
2. aus den Akzeptanzbedingungen.

In beiden Fällen kann dies mittels eines *Algorithmus* gelöst werden. Ein entsprechender Algorithmus iteriert durch die Menge L bzw. $\mathcal{C}$ und betrachtet jedes Element, um die unvollständige Information abzuleiten.

Zu **1.)** Im Fall der Links $(x, y) \in L$ wird jedes Statement eines Elements ausgewählt und der Menge S hinzugefügt, falls es noch nicht in dieser enthalten ist. Ein mögliches Vorgehen beschreibt der Algorithmus A.1 auf Seite 95.

Zu **2.)** Im Fall der Komponente der Akzeptanzfunktion wird nacheinander jedes Element $\mathcal{C}_s \in \mathcal{C}$ ausgewählt und die einzelnen Statements werden der Menge S hinzufügt. Der Algorithmus A.2 auf Seite 96 arbeitet analog zum vorherigen Ansatz. Unter der Annahme, dass den jeweiligen Akzeptanzbedingungen die dazugehörigen Statements bekannt sind, ist es ausreichend, wenn der Algorithmus lediglich die Indizes betrachtet. Die entsprechenden Statements können somit direkt angegeben werden.
Die getroffene Annahme stellt jedoch eine starke Vereinfachung dar. Gilt diese nämlich nicht, so müssen die Akzeptanzbedingungen den Statements erst zuordnet werden, wodurch sich folgende Situation ergibt:

Beispiel 3.1.1. (Zuordnung von Akzeptanzbedingungen)
$DF = (\emptyset, \{(b,a),(c,a),(b,c),(c,b)\}, \{\mathcal{C}_X = \neg b \vee c, \mathcal{C}_Y = \neg c, \mathcal{C}_Z = \neg b\})$ ⌟

Die Akzeptanzbedingungen $\mathcal{C}_X, \mathcal{C}_Y, \mathcal{C}_Z$ müssen den Statements (Knoten) erst zugeordnet werden.

Behauptung. Allgemein gilt bei einem schleifenfreien Argumentationsgraphen mit n-Knoten, dass es maximal $n * (n-1)$ verschiedene Links gibt.

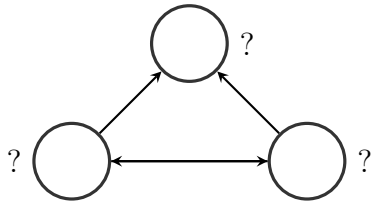


Abbildung 3.2: Ein ADF ohne Statements und entsprechender Zuteilung der Akzeptanzbedingungen

Beweis.
Man betrachte die vollständigen Graphen $K_1, \ldots, K_n$. Die Anzahl der Kanten des vollständigen Graphen K_n ist gleich $\binom{n}{2} = \frac{n(n-1)}{2}$.
Bei den Argumentationsgraphen handelt es sich um gerichtete Graphen, sodass beide Richtungen betrachtet werden. Die Anzahl der Links ist $\frac{n(n-1)}{2} * 2 = n * (n-1)$.

□

In Beispiel 3.1.1 sind drei Knoten vorhanden, wodurch es höchstens $n = 3 : 3*(3-1) = 6$ mögliche Links gibt (siehe Abbildung 3.2).

Des Weiteren müssen die Akzeptanzbedingungen den drei Statements zugeordnet werden.

$$I\ X = \neg b \vee c$$
$$II\ Y = \neg c$$
$$III\ Z = \neg b$$

Dabei stehen X, Y, Z für substituierbare Statements, die entweder aus der Menge S oder aus einer beliebigen, aber bestimmten Menge S^{π} entnommen werden. Das System muss entsprechend der Links und Akzeptanzbedingungen gelöst werden.

Offensichtlich sind Statements die Voraussetzung der ADFs, sowohl für die Links als auch für die Akzeptanzfunktion. Unvollständige Information bei den Statements kann nur unter bestimmten Annahmen künstlich ergänzt werden. Ohne initial gegebene Statements können weder Links noch Funktionen aufgrund ihrer Abhängigkeit zueinander abgeleitet werden.

3.1.2 Links

Die Repräsentation von Links erfolgt mittels gerichteter Kanten (Pfeilen) im Argumentationsgraph. Die Abbildung 3.1c gibt den Graphen ohne Kanten wieder. Die fehlenden Informationen der Links sind unproblematisch, da sie implizit über die Akzeptanzfunktion wiedergegeben werden können. Die Information der Links ist insofern redundant.

In der Arbeit [10] von Brewka et al. wird gezeigt, dass ADFs auch als Tupel $DF = (S, \mathcal{C})$ definiert werden können, wenn sich die nicht gegebenen Links stets aus der Relation zwischen Vorgängerknoten (*parents*) und entsprechender Akzeptanzbedingung des Nachfolgerknotens (*child*) ableiten lassen.

Links $(a, b) \in L$ sind implizit gegeben gdw. a [bzw. $\neg a$] in $\mathcal{C}_b$ enthalten ist [10, S. 804].

Insofern ist unvollständige Information über die Relationen zwischen den Statements ein trivialer Fall, der nicht weiter betrachtet wird.

3.1.3 Akzeptanzbedingungen

Die Menge der Akzeptanzfunktion enthält die Akzeptanzbedingung jedes einzelnen Statements, diese werden als aussagenlogische Formeln neben jedem Knoten dargestellt. In Abbildung 3.1d fehlen diese Bedingungen. Das Gerüst des Argumentationsgraphen kann dennoch aus Knoten und Kanten konstruiert werden. Durch die Akzeptanzbedingung wird festgelegt, ob ein Statement akzeptiert ist oder nicht. Ebenso enthält die Bedingung implizite Informationen (vgl. Links 3.1.2). Eine Unvollständigkeit an der Akzeptanzfunktion ist ein unbefriedigender Zustand, denn die Akzeptanzbedingung eines Statements wird für die Auswertung benötigt. Ohne diese für ein ADF essenziell Voraussetzung kann die Auswertung der Akzeptanz eines Statements nicht erfolgen. Die Akzeptanzfunktion stellt somit die entscheidungsrelevante Komponente dar.

Zu dem unvollständigen Argumentationsgraphen aus Abbildung 3.1d korrespondiert das folgende ADF:

$$DF = (\{a, b, c, d, e\}, \{(b, a), (c, a), (d, a), (e, a), (d, e), (e, d)\}, \emptyset)$$

Diese Version des Beispiels stellt einen von zwei Extremfällen dar.
In diesem Fall ist keine einzige Akzeptanzbedingung gegeben. Um die fehlende Information ableiten zu können, wird ein Algorithmus benötigt, der anhand von Statements und Links künstlich Akzeptanzbedingungen $\mathcal{C}^\pi$ erzeugt.

b	c	d	e	Akzeptanzbedingung
-	-	-	-	$\mathcal{C}_a = \neg b \pm \neg c \pm (\neg d \pm \neg e)$
-	-	-	+	$\mathcal{C}_a = \neg b \pm \neg c \pm (\neg d \pm e)$
-	-	+	-	$\mathcal{C}_a = \neg b \pm \neg c \pm (d \pm \neg e)$
-	-	+	+	$\mathcal{C}_a = \neg b \pm \neg c \pm (d \pm e)$
-	+	-	-	$\mathcal{C}_a = \neg b \pm c \pm (\neg d \pm \neg e)$
-	+	-	+	$\mathcal{C}_a = \neg b \pm c \pm (\neg d \pm e)$
-	+	+	-	$\mathcal{C}_a = \neg b \pm c \pm (d \pm \neg e)$
-	+	+	+	$\mathcal{C}_a = \neg b \pm c \pm (d \pm e)$
+	-	-	-	$\mathcal{C}_a = b \pm \neg c \pm (\neg d \pm \neg e)$
-	-	-	+	$\mathcal{C}_a = b \pm \neg c \pm (\neg d \pm e)$
-	-	+	-	$\mathcal{C}_a = b \pm \neg c \pm (d \pm \neg e)$
-	-	+	+	$\mathcal{C}_a = b \pm \neg c \pm (d \pm e)$
+	+	-	-	$\mathcal{C}_a = b \pm c \pm (\neg d \pm \neg e)$
+	+	-	+	$\mathcal{C}_a = b \pm c \pm (\neg d \pm e)$
+	+	+	-	$\mathcal{C}_a = b \pm c \pm (d \pm \neg e)$
+	+	+	+	$\mathcal{C}_a = b \pm c \pm (d \pm e)$

Tabelle 3.1: Darstellung der Möglichkeiten für die künstliche Akzeptanzbedingung von $\mathcal{C}_a^\pi$. Dabei symbolisiert + Unterstützung und − Angriff sowie ± die Verknüpfungsjunktoren $(\wedge, \vee)$.

Die aus $\mathcal{C}^\pi$ zu erzeugenden Instanzen werden, analog zu den Wahrheitstafeln, auszugsweise für das Statement a in Tabelle 3.1 dargestellt. In der Akzeptanzbedingung $\mathcal{C}_s^\pi$ kann jedes mit einer Tilde notierte Statement entweder eine Unterstützung oder einen Angriff darstellen. Dies führt zu $2^{|\tilde{S}|}$ verschiedenen Instanzen.[21]
Hinzu kommt der veränderliche Junktor ±, der entweder für eine Konjunktion $(+, \wedge)$ oder für eine Disjunktion $(-, \vee)$ stehen kann.[22] Daraus ergibt sich, dass jede aussagenlogische Formel $2^{|Junktoren|}$ verschiedene Instanzen bilden kann. In Summe sind

[21] $|\ |$ bezeichnet allg. die Anzahl; $|S|$ ist somit die Anzahl von S.
[22] Theoretisch ist jeder Junktor aus Tab. 2.3 möglich, wodurch sich die Komplexität zusätzlich erhöhen würde, da erst Äquivalenzumformungen durchgeführt werden müssen, um eine bestimmte Form zu erhalten (vgl. Def. A.2.3).

mindestens $2^{|\tilde{S}|} + 2^{|Junktoren|} = 2^{(n+(n-1))}$ Instanzen einer künstlichen Akzeptanzbedingung eines einzigen Statements möglich, wobei n gleich der Anzahl der künstlich ergänzten Statements einer Akzeptanzbedingung ist.

Betrachtet wird nun der zweite Extremfall, in dem nur eine einzige Akzeptanzbedingung unvollständig ist. Es ist wieder das ADF aus Beispiel 3.1 gegeben, jedoch mit folgender (einziger) Änderung: $\mathcal{C}_a = b \wedge c \wedge d$. Die Akzeptanzfunktion von $\mathcal{C}_a$ wird künstlich um das Statement e, des letzten Elements der Links, zu $\mathcal{C}_a^\pi = b \wedge c \wedge d \pm \tilde{e}$ erweitert. Der Junktor zwischen d und e kann durch eine geschickte Prüfung auf eine Disjunktion festgelegt werden, sodass nur die Zeilen 2 und 4 der nachfolgenden Tabelle 3.2 als mögliche Instanziierungen übrig bleiben.

e	Akzeptanzbedingung
$+$	$b \wedge c \wedge d \wedge e$
$+$	$b \wedge c \wedge d \vee e$
$-$	$b \wedge c \wedge d \wedge \neg e$
$-$	$b \wedge c \wedge d \vee \neg e$

Tabelle 3.2: Darstellung aller Möglichkeiten für die künstliche Akzeptanzbedingung $\mathcal{C}_a^\pi$

Die künstlichen Akzeptanzbedingungen $\mathcal{C}_s^\pi$ erzeugen exponentiell viele mögliche Instanziierungen. Es zeigt sich, dass die Akzeptanzfunktion das Kernstück der ADFs aus Abschnitt 2.3 ist. Eine Unvollständigkeit kann nur bis zu einem gewissen Grad behoben werden.

Das Beispiel mit der maximalen Unvollständigkeit hat gezeigt, dass der Aufwand für die Rekonstruktion eines ADFs enorm schwierig bis unmöglich ist. Das Minimalbeispiel wiederum hat verdeutlicht, dass eine akzeptable „Rekonstruktionsmenge“ für die Ableitung der Information mit entsprechenden Algorithmen gefunden werden kann.

3.1.4 Kombination

Eine Kombination aus der Unvollständigkeit von Statements 3.1.1, Links 3.1.2 und Akzeptanzbedingungen 3.1.3 führt zu enorm komplexen Fällen. Das Maß an Unvollständigkeit ist dabei derart groß, dass die Rekonstruktion eines vollständigen Argumentationsgraphen sehr schnell unmöglich wird. Dazwischen gibt es Kombinationen,

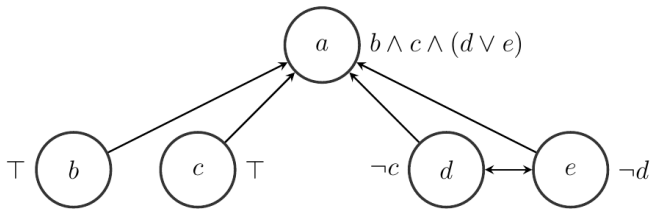
die dazu führen, dass der Graph Ungenauigkeiten respektive Unsicherheiten enthält. Die Kombination als Sonderfall wird im Rahmen dieser Arbeit nicht weiter betrachtet.

3.2 Unsicherheit an Komponenten

Unvollständige Information führt auch dazu, dass die Existenz von Argumenten nicht garantiert werden kann. Im vorherigen Abschnitt wurde Unvollständigkeit aus der Richtung untersucht, dass etwas fehlt und durch Ableitung aus gegebenen Informationen hinzugefügt werden muss. Betrachtet wird nun Unvollständigkeit aus der Richtung, dass ein Framework grundsätzlich vollständig vorliegt, jedoch das Vorhandensein einiger Elemente nicht garantiert werden kann. Zum Beispiel reicht die Datenlage nicht aus, um das Vorhandensein von Elementen an einer Komponente sicherzustellen. Ein Szenario dafür ist das Zusammenführen von Datenbanken mit inkonsistenten Werten über dasselbe Objekt.

Grundlage der ADFs sind die Statements, die nach Definition 2.3.1 Voraussetzung für die anderen Komponenten sind. Die Akzeptanzbedingungen geben diesen Statements und Relationen erst ihre Bedeutung. Aufgrund der Abhängigkeit der Komponenten untereinander ist es zielführend, die grundlegende Komponente der Statements genau zu betrachten – präzise ausgedrückt, wie ein ADF mit unvollständiger Information umgehen kann, wenn die Datenlage die Existenz eines Statements nicht sicherstellt.

Betrachtet wird das folgende ADF aus Beispiel 3.1. Dieses ist in Abbildung 3.1a dargestellt und nachfolgend noch einmal abgebildet. Nunmehr kann die Existenz des Statements d nicht garantiert werden.



Sowohl die Links als auch die Akzeptanzfunktion stehen in „Wechselwirkung“ zu den Statements. Ist die Existenz eines Statements unsicher, so hat dies Auswirkungen auf die anderen Komponenten eines ADFs. Im Rahmen dieser Arbeit werden im Folgenden drei Ansätze betrachtet: 3.2.1 Transparenz, 3.2.2 Vergessen und 3.2.3 Nicht-klassische Logik.

3.2.1 Transparenz

Aus der Abhängigkeit eines Knotens zu seinen Elternknoten folgt die (mittelbare) Abhängigkeit eines Knotens zu seinen Vorgängern[23]. Dabei kommt es auf die „Position“ des Statements im Graphen bzw. in der Akzeptanzbedingung an. Ein Statement ist ein:

i Blattknoten gdw. es keine Elternknoten und mindestens einen Nachfolger gibt.
$indeg(v) = 0$ und $outdeg(v) \geq 1$

ii Kindknoten gdw. es mindestens einen Vorgänger und keine Nachfolger gibt.
$indeg(v) \geq 1$ und $outdeg(v) = 0$

iii Mittelknoten gdw. es sowohl Nachfolger als auch Vorgänger gibt.
$indeg(v) \geq 1$ und $outdeg(v) \geq 1$

Die Abbildung 3.3 wird genutzt, um die beschriebenen Positionen zu visualisieren.

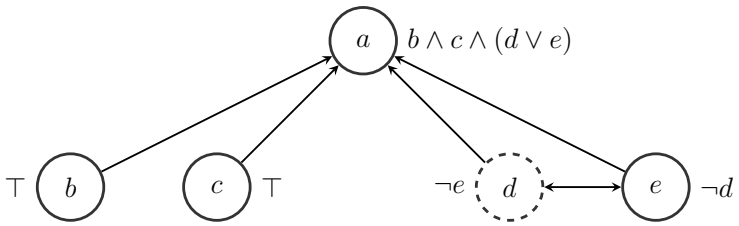


Abbildung 3.3: Die Existenz des Statements d ist unsicher, gekennzeichnet wird dies durch den gestrichelten Knoten.

Das Statement d soll gegenüber den anderen Statements transparent werden, d.h., dass alle Vorgänger von d nun direkt auf die Nachfolger von d wirken. In diesem Fall

[23] Vorgänger eines Knotens $v := an(v)$ siehe Definition A.1.2

ist d ein Blattknoten und verschwindet für alle anderen Statements, Links und Akzeptanzbedingungen.
Für den Fall des Blattknotens (i) muss das Vorkommen des zu entfernenden Statements aus den Akzeptanzbedingungen anderer Statements geschnitten werden. Das Statement d ist der Elternknoten zu a, woraus folgt, dass d aus $\mathcal{C}_a$ entfernt werden muss. In einer aussagenlogischen Formel kann ein Statement, jeweils für sich betrachtet, mit folgenden Einbettungen vorkommen:

1. mit *keinem* Verknüpfungsjunktor
   → Die atomare Formel $\mathbb{A}$ kann entfernt werden.
2. mit *einem* Verknüpfungsjunktor
   a) Position vorne: $\mathcal{C} = x \pm \ _{...}$ oder
   b) Position hinten: $\mathcal{C} = \ _{...} \pm x$.
   → In beiden Fällen kann das Statement zusammen mit seinem Junktor entfernt werden.
3. bei einem Einschluss zwischen *zwei* Verknüpfungsjunktoren $\mathcal{C} =_{...} \pm x \pm_{...}$.
   → In diesem Fall wird der Operator $\ddot{\imath}$ aus Definition 3.2.1 benutzt, um die aussagenlogischen Formelteile wieder zusammenzufügen.

Definition 3.2.1.
Der Operator $\ddot{\imath}$ verbindet zwei aussagenlogische Formelteile wie folgt:

$$\ddot{\imath} \quad = \quad \begin{cases} \wedge, & \text{falls } \ddot{\imath}(\wedge, \wedge) \\ \vee, & \text{sonst.} \end{cases}$$

Unter Beachtung der Präzedenz der Junktoren.

Hier hat $d \in \mathcal{C}_a$ zwei verschiedene Verknüpfungsjunktoren $\wedge$ d $\vee$, sodass unter Verwendung des $\ddot{\imath}$-Operators $\mathcal{C}_a = b \wedge c \vee e$ übrig bleibt.[24]

Sei d nun ein Mittelknoten, indem das ADF aus dem laufenden Beispiel 3.1 um $\{d_1, d_2\} \in S$, $\{(d_1, d), (d_2, d)\} \in L$ und $\mathcal{C}_d = d_1 \vee d_2$ sowie $\mathcal{C}_{d_1} = \top, \mathcal{C}_{d_2} = \top$ erweitert wird. Wenn der Knoten d transparent wird, dann wird jede Akzeptanzbedingung, in

[24] Die Klammerung der Akzeptanzbedingung wird mittels Distributivität aus Tab. A.2 aufgelöst.

der d vorgekommen ist, durch die Eltern von d ersetzt. Dies betrifft die Statements a und e. Daraus ergibt sich:

$$\mathcal{C}_a = b \wedge c \wedge ((d_1 \vee d_2) \vee e)$$
$$\mathcal{C}_e = \neg(d_1 \vee d_2)$$

Im Fall des Kindknotens wird das Statement a betrachtet. Da es keine Statements gibt, auf die a wirkt, kann a komplett entfernt werden.

Analog zur semantischen Äquivalenz aus Definition 2.2.8 wird folgende Definition für die ADFs eingeführt:

Definition 3.2.2 (semantisch äquivalente ADFs)**.**
Seien DF_1 und DF_2 zwei ADFs. Dann sind DF_1 und DF_2 semantisch äquivalent, falls für alle Interpretationen $v \in \mathcal{V}_3$ gilt:

$$\Gamma_{DF_1}(v) \equiv \Gamma_{DF_2}(v).$$

Zwei ADFs sind semantisch äquivalent, wenn ihre Semantiken unter denselben Interpretationen übereinstimmen.

Beispiel. Der Ansatz zur Transparenz wird mit einem Gegenbeispiel widerlegt. Das ADF aus Beispiel 2.3.2 sei nochmals gegeben:

$$DF_1 = (\{a, b, c\}, \{(a, c), (a, b), (b, c), (b, a)\}, \{\mathcal{C}_a = \neg b, \mathcal{C}_b = \neg a, \mathcal{C}_c = \neg a \vee b\}).$$

Die Existenz des Statements a sei unsicher, sodass a wie beschrieben transparent wird. Daraus ergibt sich:

$$DF_2 = (\{b, c\}, \{(b, c)\}, \{\mathcal{C}_b = \neg b, \mathcal{C}_c = b \wedge \neg b\}).$$

Damit der beschriebene Ansatz gilt, müssen beide ADFs semantisch äquivalent nach Definition 3.2.2 sein. Die Extension für die grundierte Semantik ist gleich $E_{DF_1} = E_{DF_2} = \emptyset$. Für die vollständige Semantik ist $E_{DF_1} = \{a\}, \{b, c\}$, während DF_2 unter derselben Interpretation keine Extension besitzt. Dies ist ein Widerspruch $\lightning$.

Ein Statement transparent werden zu lassen verfälscht den ursprünglichen Argumentationsgraphen. Dieser Ansatz wird somit verworfen.

3.2.2 Vergessen

Ein anderer Ansatz besteht darin, ein Statement, dessen Existenz unsicher ist, zu vergessen. Dabei muss das Statement aus der Wissensbasis entfernt werden. Dies kann entweder auf syntaktischer oder semantischer Ebene erfolgen. Ein Statement wird mittels Aussagenvariable repräsentiert. Den Aussagenvariablen eines ADFs wird ein Wert zugewiesen (Interpretation). Soll ein Statement vergessen werden (*forgetting variables*), so ist darauf zu achten, dass die Wissensbasis respektive der Argumentationsgraph nicht verfälscht wird.

Die Komponente der Akzeptanzfunktion eines ADFs enthält alle Akzeptanzbedingungen. Diese Bedingungen werden durch aussagenlogische Formeln dargestellt. Die Menge dieser aussagenlogischen Formeln wird mit φ bezeichnet. Jede aussagenlogische Formel kann in eine Normalform[25] transferiert werden. Häufig bringt es Vorteile mit sich, wenn nicht mit willkürlichen aussagenlogischen Formeln, sondern mit bestimmten Normalformen gearbeitet wird. Sind die φ in konjunktiver Normalform (KNF), wird dies als φ_{KNF} notiert.

Ein passender Ansatz wird in der Arbeit [14] von Delgrande beschrieben. Dieser kann insofern übernommen werden, da es sich bei den Akzeptanzbedingungen der ADFs (in dieser Arbeit) um aussagenlogische Formeln handelt. Folgendes Theorem wird aus [14, S. 1184-1185] übernommen:

Theorem 3.2.1. Sei φ eine Menge aussagenlogischer Formeln und p ein Atom.

1. $\mathcal{F}(\varphi, p) \leftrightarrow \varphi[p/\top] \vee \varphi[p/\bot]$ für endliche φ.
2. $\mathcal{F}(\varphi, p) \leftrightarrow \varphi_{KNF_{\downarrow p}} \cup \mathrm{Res}(\varphi_{KNF}, p)$.

Die Formel 1 des Theorems 3.2.1 entspricht dem Ansatz von Boole (1854) und kann für die ADFs übernommen werden, da es sich gemäß Definition 2.3.1 um endliche aussagenlogische Formeln handelt. Um eine Variable (Statement, Aussagenvariable)

[25] vgl. Def. A.2.3 auf Seite 92.

vergessen zu können, muss das Vorkommen der Variable, in einer logischen Formel mit $\{\top, \bot\}$ substituiert werden. Das „Vergessen“ eines Literals l in einer aussagenlogischen Formel φ ist schließlich definiert als:

$$\text{Vergessen}(\varphi, l) = \varphi[l/\top] \vee \varphi[\neg l \wedge \varphi] \qquad \text{[14, Definition 9].}$$

Beispiel 3.2.
Betrachtet wird wieder das ADF aus Beispiel 3.1. Die Akzeptanzbedingungen liegen bereits in KNF vor, es gilt φ_{KNF}. Bei $\{\top, \bot\}$ handelt es sich um Konstanten, deshalb werden sie aus φ entfernt.

$$\varphi_{KNF} = \{b \wedge c \wedge (d \vee e), \neg d, \neg e\}.$$

$$\begin{aligned}
\text{Für } \mathcal{C}_a : \varphi[d/\top] \vee \varphi[d/\bot] &= \{(b \wedge c \wedge (\top \vee e)) \vee (b \wedge c \wedge (\bot \vee e))\} \equiv b \wedge c \\
\text{Vergessen}(\varphi, d) &= \varphi[d/\top] \vee \varphi[\neg d \wedge \varphi] \equiv (b \wedge c \wedge (\top \vee e)) \vee \\
&(\neg d \wedge (b \wedge c \wedge (d \vee e))) \equiv b \wedge c \\
\text{Für } \mathcal{C}_e : \varphi[d/\top] \vee \varphi[d/\bot] &= \{\neg\top \vee \neg\bot\} \equiv \top \\
\text{Vergessen}(\varphi, \neg d) &= \varphi[\neg d/\top] \vee \varphi[\neg\neg d \wedge \varphi] \equiv \top \vee (d \wedge \neg d) \equiv \top
\end{aligned}$$

Beide Ergebnisse sind logisch äquivalent. ⌟

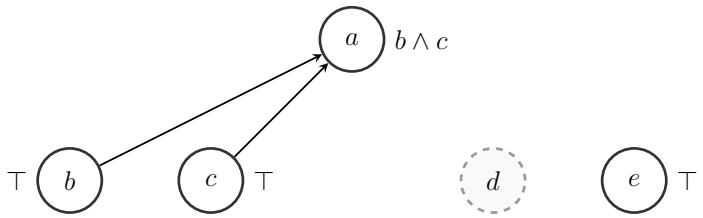


Abbildung 3.4: Das Statement d wurde „vergessen“

Die Existenz des Statements d ist unsicher. Wird d vergessen, so ergibt sich folgendes ADF: $DF_{\not d} = (\{a, b, c, e\}, \{(b, a), (c, a)\}, \{\mathcal{C}_a = b \wedge c, \mathcal{C}_b = \mathcal{C}_c = \mathcal{C}_e = \top\})$. Die Argumentationsgraphen aus Abbildung 3.3 und 3.4 sollten semantisch äquivalent nach Definition 3.2.2 sein. Betrachtet werden die folgenden zwei Interpretationen:

$$v_1 = \{a \mapsto t, b \mapsto t, c \mapsto t, d \mapsto f, e \mapsto t\}$$
$$v_2 = \{a \mapsto t, b \mapsto t, c \mapsto t, d \mapsto t, e \mapsto f\}.$$

Tatsächlich gilt für v_1, dass $\Gamma_{DF_\not d}(v_1)$ und $\Gamma_{DF}(v_1)$ des ursprünglichen ADFs dieselbe Extension haben. Unter der Interpretation v_2 unterscheiden sich die ADFs, da $\Gamma_{DF}(v_2)$ ein Fixpunkt ist, jedoch $\Gamma_{DF_\not d}(v_2)$ nicht. Somit scheidet auch dieser Ansatz aus.

3.2.3 Nicht-klassische Logik

Ist die Existenz eines Statements nicht garantiert, so kann der Wahrheitswert nicht ausschließlich wie bisher nur *wahr* oder *falsch* sein. Die Wahrheitswerte 2.2.3 der klassischen Logik reichen nicht mehr aus, um ein zweifelhaftes Vorhandensein darstellen zu können. Das Prinzip der Zweiwertigkeit wird zugunsten einer mehrwertigen Logik aufgegeben. Die dreiwertige (ternäre) Logik erweitert die klassischen Wahrheitswerte um den Wert *unbekannt*, *unentscheidbar* bzw. *unsicher* (kurz u bzw. U).
Damit ein vollständiges Gitter, statt eines Halbgitters (vgl. Ausführungen in Abschnitt 2.3.1), erzeugt werden kann, wird eine vierwertige Logik eingeführt. Diese ergänzt die dreiwertige Logik um den Wert *inkonsistent* (kurz i bzw. I).

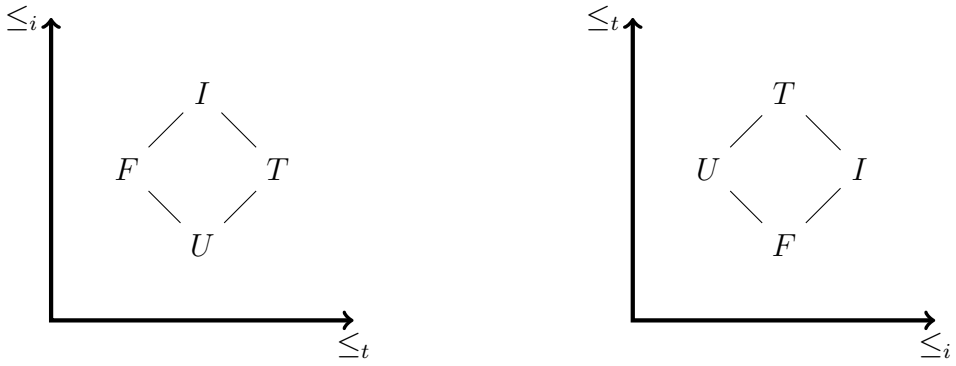
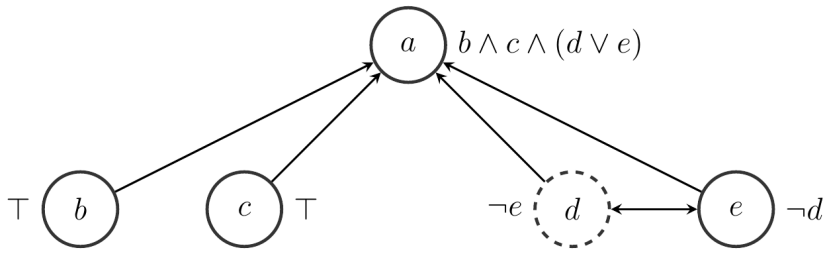


Abbildung 3.5: Bi-Gitter, kombiniert aus Informationsgehalt und Wahrheitswert: $\mathcal{G} = (\{T, F, U, I\}, \leq_i, \leq_t)$

Das ADF aus Beispiel 3.1 auf Seite 34 sei nochmals gegeben, wobei die Existenz des Statements d unsicher ist. Diesem Statement wird der nicht-klassische Wahrheitswert $d \mapsto u$ zugewiesen. Gesucht wird nun ein Modell für das ADF. Dieses wurde in Abschnitt 2.3 mit Gleichung 2.4 wie folgt eingeführt:

Ein ADF besitzt ein zweiwertiges Modell gdw. $v(s) = v(\mathcal{C}_c)$ für jedes $s \in S$ gilt.
Ein Lösungsansatz wurde ebenfalls im obigen Abschnitt mit der AFT in Kombination mit dem Γ-Operator skizziert. Eine approximierte zweiwertige Interpretation für d ist möglich, indem ein Paar von Interpretationen (v_1, v_2) verwendet wird, sodass $v_1 \leq d \leq v_2$ gilt (vgl. Approximationsgleichung 2.2). Dabei repräsentiert v_1 eine größte untere Schranke (Infimum) und v_2 eine kleinste obere Schranke (Supremum) in einem Gitter. Um ein Gitter gemäß Gleichung 2.2 zu erzeugen, wird ein Bi-Gitter genutzt (vgl. Abbildung 3.5). Die Werte, die d in einer zweiwertigen Logik annehmen kann, sind $v_1 = falsch$ und $v_2 = wahr$. Dies geschieht, ohne dass der ursprüngliche Argumentationsgraph verändert werden muss.



Eine Interpretation $v \in \mathcal{V}_3$ mit

a	b	c	d	e
w	f	w	**u**	f

kann mit folgenden Interpretationen $\mathcal{V}_2$ approximiert werden

	a	b	c	d	e
w_1	w	f	w	**f**	f
w_2	w	f	w	**w**	f

Die Auswertung erfolgt über den Γ-Operator aus Definition 2.3.4 anhand eines Bi-Gitters. Die Approximationsmenge für d ist $\{w_1, w_2\}$ mit $w_1 : d \mapsto f$ und $w_1 : d \mapsto w$. Der Argumentationsgraph aus Abbildung 3.3 bleibt für die Approximation unverändert und somit vollständig erhalten.

Das Statement d kann somit, trotz unvollständiger Information, approximiert und ausgewertet werden. Dieser Ansatz wird später konkret an Beispielen in Abschnitt 5.2 untersucht.

4 Adaptiv abstrakt dialektisches Framework

Die Argumentation ist eine Grundform des logisch menschlichen Schlussfolgerns (*commonsense reasoning*). Trotz unvollständiger Information und *Argumentationsdynamik* können Menschen im Alltag problemlos argumentieren. Ziel der KI ist es, dieses (menschliche) Verhalten zufriedenstellend zu simulieren.
Die in den Grundlagen (Kapitel 2) vorgestellten Ansätze der formalen Argumentation bieten grundsätzlich Möglichkeiten dafür, dass ein maschinelles System mit Argumentation umgehen kann. Problematisch wird eine zutreffende Auswertung, wenn Information unvollständig ist. In diesem Kapitel wird ein Ansatz vorgestellt, um die problematische Rigidität der ADFs zu beseitigen.

4.1 Anpassungsfähigkeit der ADFs

Die Motivation für anpassungsfähigere ADFs liegt in der Dynamik einer Argumentation begründet. Für KI-Systeme ist ein Mangel an Information stets problematisch, während Menschen relativ gut mit „Unwissenheit“ umgehen können. Bei KI-Systemen führt unvollständige Information zu schlechterer Qualität möglicher Schlussfolgerungen, trivialen Ergebnissen oder sogar zu gar keinem Ergebnis. Deshalb wird eine Idee vorgestellt, die das Argumentationsframework flexibler macht. Die Grundidee ist es, dass dem ADF keine konkreten Akzeptanzbedingungen mehr vorgegeben werden, sondern eine allgemeinere (Ziel-) Vorgabe.

Menschliche Argumentation folgt bestimmten (Argumentations-) Mustern (*pattern*), die dazu führen können, dass eine These akzeptiert oder nicht akzeptiert wird. Diese Muster können je nach Anwendungsgebiet unterschiedlich ausgeprägt sein. Beispielsweise gibt es sogenannte Ausschlusskriterien (umgangssprachlich K.O.-Kriterien), die eine These unabhängig aller weiteren Argumente nichtig machen. Analog zum de-

klarativen Programmierparadigma[26] steht das gewünschte Ergebnis im Vordergrund. Überträgt man diesen Gedanken auf das Framework, so wird durch die Akzeptanzbedingung keine konkrete Prämisse mehr vorgegeben, sondern über eine allgemeinere Vorgabe das zu erreichende Ziel. Dadurch werden die ADFs weniger rigide.

Um diese Idee zu illustrieren, wird eine Argumentation zwischen zwei Parteien betrachtet. In dem entstehenden Argumentationsprozess werden Pro- und Kontraargumente gegeneinander abgewogen.[27]

Beispiel 4.1 (Kino)**.**
Die beiden Personen Alice (A) und Bob (B) diskutieren darüber was sie heute noch unternehmen wollen. Alice schlägt vor ins Freilichtkino zu gehen. Daraufhin entsteht eine Argumentation, die über den Vorschlag entscheiden soll. Im Verlauf dieser Argumentation treten verschiedene Argumente bzw. Statements auf, die wie folgt abgekürzt werden: Freilichtkino (k), Wetter (w), Film (f), Begleitung (b), Eintrittspreis (p), Entfernung (e), Rabatt (r).

In diesem Beispiel einer Argumentation kommen zuerst zwei Proargumente (gutes Wetter, interessanter Film) und ein Kontraargument (Entfernung zum Veranstaltungsort) vor, um die These *ins Freilichtkino gehen* zu stützen.[28] Beide A und B können ihre vorgebrachten Argumente abwägen, um zu einem Schluss zu kommen.

A: Lass uns ins Freilichtkino gehen, weil das Wetter gut und der Film interessant ist.
B: Das Kino ist aber ziemlich weit weg.

Eine Argumentation wird meistens nicht naiv geführt, d.h., dass sich gegenüberstehende Parteien bereits vor dem Austausch von Argumenten darüber bewusst sind, welches Ziel sie mit ihrer Argumentation verfolgen. Dadurch entsteht eine Vorgabe (gewünschtes Ergebnis), die angibt unter welchen Voraussetzungen ein Argument akzeptiert ist. Unter der Zielvorgabe, dass es mehr Argumente für als gegen die These gibt, fällt der (positive) Schluss ins Freilichtkino zu gehen.

[26] Algorithmen wird vermittelt „*was*“ gelöst werden soll, statt konkret das „*wie*“ vorzugeben.
[27] Proargumente sind (stützende) Argumente, die `für` etwas stehen und Kontraargumente sind (angreifende) Argumente, die da`gegen` stehen.
[28] Ein Argument bzw. Statement v mit $indeg(v) > 0$ ist auch eine These (vgl. Def. A.1.3).

Ein Argumentationsprozess verläuft dynamisch, d.h., dass weitere Argumente hinzukommen oder wegfallen können (Argumentationsdynamik).

A: Wir können zusammen ins Kino gehen.

An dieser Stelle kommt ein weiteres Proargument (nette Begleitung) hinzu. Anhand der Zielvorgabe kann eine Akzeptanzbedingung erstellt und später ausgewertet werden. War die These *ins Freilichtkino gehen* bereits vor der Hinzunahme eines neuen Arguments akzeptiert, so ist keine Anpassung (*update*) des Frameworks durchzuführen. Die Schlussfolgerung ändert sich in diesem Fall nicht. ⌟

Durch das Beispiel Kino wurde noch einmal die Motivation für anpassungsfähigere ADFs verdeutlicht. In diesem Fall hatte die neue, bisher unbekannte Information keinerlei Auswirkung auf die Schlussfolgerung des Frameworks. Durch die Zielvorgabe sind unvollständige Informationen bei den Akzeptanzbedingungen besser zu integrieren ohne spezifische Anpassungen vornehmen zu müssen.

Das in Beispiel 4.1 beschriebene Argumentationsframework wird im folgenden formal notiert. Es handelt sich dabei grundsätzlich um ein ADF. Ein Argument ist ein abstraktes Statement s, das in direkter Relation zu anderen Statements stehen kann. Eine solche Relation wird Link genannt und existiert von einem Statement s zu seinen Vorgängern, die als Eltern bezeichnet werden. Bei dieser Adaption des Frameworks müssen Links besonders ausgezeichnet werden, i.d.R. als unterstützende, angreifende oder sonstige Relationen. Die Akzeptanzbedingung eines jeden Statements ist entweder initial vorhanden oder lässt sich bei Bedarf ableiten. Als zusätzliche Komponente, gegenüber der Definition 2.3.1 der ADFs, wird die Zielvorgabefunktion ζ eingeführt. Jedes Statement s, dessen Eingangsgrad größer als eins ist, benötigt von Anfang an eine Zielvorgabe. Anhand dieser Vorgabe und den Eltern $par(s)$ kann für ein Statement s eine Akzeptanzbedingung $\mathcal{C}_s^\pi$ erzeugt werden. Dies führt zu folgender Definition:

Definition 4.1.1 (ζADF)**.**
Ein abstrakt dialektisches Framework mit Zielvorgabe (ζADF) ist ein Quadrupel $ZF = (S, L, \mathcal{C}, \zeta)$, wobei

$S \subseteq \mathcal{U}$ eine endliche Menge an abstrakten Statements ist,

$L \subseteq S \times S$ eine endliche Menge an Links ist, die aus unterstützenden, angreifenden und sonstigen Links besteht,

$\mathcal{C} = \{\mathcal{C}_s\}_{s \in S} \cup \{\mathcal{C}_s^{\pi}\}_{s \in S}$ ist eine Menge von Akzeptanzbedingungen und künstlichen Akzeptanzbedingungen für jedes Statement $s \in S$ und

$\zeta = \{\zeta_s\}_{s \in S}$ ist die Menge an Zielvorgaben für jedes Statement $s \in S$. Verpflichtend für $\forall s \in S$, wenn $indeg(s) > 1$.

Alle unterstützenden Links werden mit L^+, alle angreifenden Links mit L^- und alle sonstigen Links mit $L^{\pm}$ bezeichnet. Die Zusammensetzung der Links ist:

$$L = \begin{cases} L^+ & \text{wenn unterstützend,} \\ L^- & \text{wenn angreifend,} \\ L^{\pm} & \text{sonst.} \end{cases}$$

Notiert wird ein unterstützender Link als $(r, s)^+$, ein angreifender Link als $(r, s)^-$ und ein sonstiger Link als $(r, s)^{\pm}$. Ein Link $(r, s) \in L$ ist grundsätzlich

- unterstützend $(r, s)^+$ gdw. für $C_s(par(s) \setminus r) = true$ und $C_r = true$ gilt $C_s(par(s) \cup \{r\}) = true$,
- angreifend $(r, s)^-$ gdw. für $C_s(par(s) \setminus r) = true$ und $C_r = true$ gilt $C_s(par(s) \cup \{r\}) = false$.

Die Zielvorgaben entsprechen menschenähnlichen Argumentationsmustern, die auszugsweise in Tabelle 4.1 dargestellt sind. Dieser nicht abschließende Auszug zeigt, dass die bisher zugrundeliegende Aussagenlogik (AL) nicht mehr ausreicht, um alle Muster ausdrücken zu können. Deshalb wird partiell die Prädikatenlogik 1. Stufe (PL1) als Erweiterung verwendet.

Die hier verwendete Form der PL1 erweitert die AL um Quantoren und Funktionssymbole. Die wesentlichen Bestandteile der Argumentationsmuster sind die sprachlichen Konstrukte „für alle“ ($\forall$), „es existiert“ ($\exists$) sowie die Verwendung von Funktionen ($\otimes, ko$) und Relationen ($>, \geq$).

Zielmuster	Bedeutung
$\forall+$	Alle positiven Links müssen aktiv sein.
$\exists+$	Mindestens ein positiver Link muss aktiv sein.
$\nexists-, \forall+$	Kein negativer und alle positiven Links müssen aktiv sein.
$\nexists-; \exists+$	Kein negativer oder mindestens ein positiver Link muss aktiv sein.
$+ > -$	Mehr positive als negative Links müssen aktiv sein.
$- \geq +$	Mindestens gleich viele negative wie positive Links müssen aktiv sein.
$\otimes(u, v)$	XOR-Funktion: entweder u oder v.
$ko(s)$	Ausschlusskriterium (Funktion): unterstützendes s muss zu $true$ bzw. angreifendes s zu $false$ ausgewertet werden.

Tabelle 4.1: Ein Auszug an möglichen Zielvorgaben, die anhand von Argumentationsmustern gebildet werden

4.2 Technische Darstellung

Der Argumentationsgraph eines Frameworks nach Definition 4.1.1 entspricht größtenteils der bisher bekannten Darstellungsweise. Die Statements werden als Knoten und die Links als Kanten mit + für unterstützende, − für angreifende oder ± für sonstige Links dargestellt. Statt der Akzeptanzbedingung steht grundsätzlich die Zielvorgabe für Statements neben den Knoten.

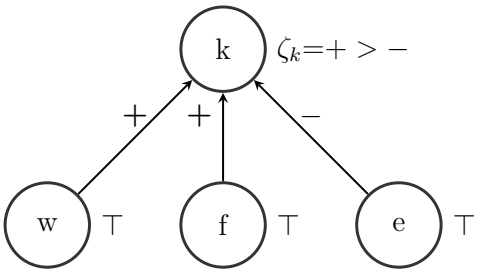


Abbildung 4.1: Darstellung des Argumentationsgraphen aus dem Beispiel Kino

Das Beispiel 4.1 ist in seiner Grundform als Argumentationsgraph eines ζADFs in Abbildung 4.1 dargestellt. Das dazu korrespondierende ζADF ist:
$ZF = (\{k, w, f, e\}, \{(w, k)^+, (f, k)^+, (e, k)^-\}, \{\mathcal{C}_w = \mathcal{C}_f = \mathcal{C}_e = \top\}, \{\zeta_k = + > -\})$.

Anhand der Zielvorgabe und den direkten Vorgängern $par(\mathrm{k})$ des Statements k kann die Akzeptanzbedingung $\mathcal{C}^{\pi}_{\mathrm{k}}$ erzeugt werden. Die Zielvorgabe $\zeta_k = + > -$ gibt an,

dass mehr Argumente für das Statement als dagegen sprechen müssen. Für die Akzeptanzbedingung des Statements Kino folgt:

$$\mathcal{C}^{\pi}_{\mathrm{k}} = ((w \vee f) \wedge \neg e) \vee ((w \wedge f) \wedge \underbrace{(e \vee \neg e)}_{\text{Tautologie}})$$

Informationen können jederzeit hinzugefügt oder entfernt werden, wodurch ggf. ein Update des Argumentationsframeworks notwendig wird (Anpassung der Akzeptanzbedingung). Dabei ergibt sich die Akzeptanzbedingung direkt aus der Zielvorgabefunktion ζ und den Eltern eines Statements $par(s)$. Dadurch ist es nicht zwingend notwendig, dass die Menge der Akzeptanzbedingungen von Anfang an vollständig vorhanden ist oder bei jeder Veränderung der Argumente (manuell) angepasst werden muss.

4.3 Eigenschaften

Damit weitere Eigenschaften der ζADFs aufgezeigt werden können, wird das Beispiel Kino 4.1 erweitert. Interessant ist es zu betrachten, wie sich das Framework unter Hinzunahme von Argumenten bei verschiedenen Zielvorgaben verhält.

Beispiel. (Forstsetzung)
Es kommen weitere Argumente hinzu:

B: Der Eintrittspreis ist mir zu teuer.
A: Durch eine Rabattaktion halbiert sich der Preis. ⌟

Die Abbildung 4.2 erweitert den ursprünglichen Argumentationsgraphen wie folgt:

$$\begin{aligned} ZF = (&\{k, w, f, b, e, p, r\}, \\ &\{(w,k)^+, (f,k)^+, (b,k)^+, (e,k)^-, (p,k)^-, (r,p)^-\}, \\ &\{\mathcal{C}_w = \top, \mathcal{C}_f = \top, \mathcal{C}_b = \top, \mathcal{C}_e \top, \mathcal{C}_p = \neg r, \mathcal{C}_r = \top\}, \\ &\{\zeta_k = + > -\}) \end{aligned}$$

Jeder Knoten $indeg(v) > 1$ benötigt gem. Definition 4.1.1 eine Zielvorgabe. Die Akzeptanzbedingung für p ist $\neg r$, es handelt sich um einen angreifenden Link $(r,p)^-$.

Sprachlich formuliert: wenn es einen Rabatt r gibt, dann ist der Eintrittspreis p nicht zu hoch. Die Akzeptanzbedingung für k (*ins Freilichtkino gehen*) wird durch folgende komplexe aussagenlogische Formel ausgedrückt:

$$\begin{aligned}\mathcal{C}^{\pi}_{\mathrm{k}} = &((w \vee f \vee b) \wedge (\neg p \wedge \neg e)) \vee \\ &(((w \wedge f) \vee (w \wedge b) \vee (f \wedge b)) \wedge ((\neg p \vee e) \vee (p \vee \neg e))) \vee \\ &((w \wedge f \wedge b) \wedge \text{Tautologie})\end{aligned}$$

Verändern sich beispielsweise die direkten Vorgängerstatements des Statements k, so gibt es zwei Möglichkeiten:

1. das Framework muss wegen der Anpassung der Akzeptanzbedingung aktualisiert werden oder
2. das Framework kann unverändert bleiben, ohne dass sich das Ergebnis bzw. die Schlussfolgerung verändert.

In diesem Fall muss das Framework nicht angepasst werden, da die Akzeptanzbedingung $\mathcal{C}^{\pi}_{k}$ bereits vor Hinzunahme des Statements p zu $true$ ausgewertet wurde, also akzeptiert war: $\mathcal{C}^{\pi}_{k}(par(k) \setminus p) = true \cup (p,k)^{-} = false \rightarrow \mathcal{C}^{\pi}_{k}(par(k) \cup \{p\}) = true$

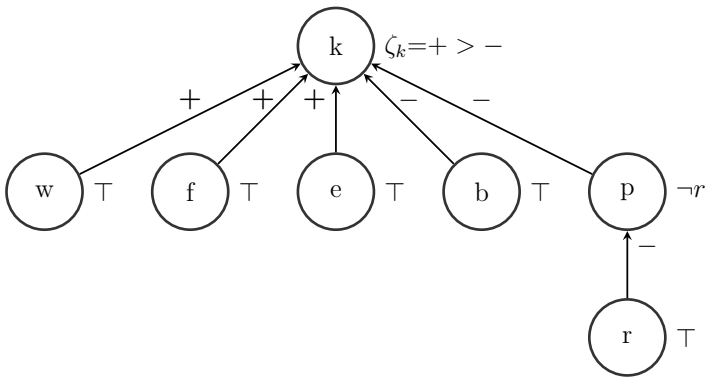


Abbildung 4.2: Anpassung des Argumentationsgraphen an den Verlauf der geführten Argumentation (Argumentationsdynamik).

Verändert sich nun die Zielvorgabefunktion der These wie folgt: $\zeta_k = ko(p)$, hängt die Entscheidung sprachlich formuliert allein vom Eintrittspreis p ab. Dabei handelt es sich um eine angreifende Relation $(p,k)^{-}$. Wird $\mathcal{C}_p$ zu $false$ ausgewertet, dann kann $\mathcal{C}^{\pi}_{k}$, deren einzige Akzeptanzbedingung von p abhängt, zu $true$ ausgewertet werden.

In diesem Fall existiert ein Rabatt, wodurch p tatsächlich zu $false$ und k zu $true$ ausgewertet wird.

Die Zielvorgabefunktion lässt sich beliebig verändern, wodurch stets eine künstliche Akzeptanzbedingung erzeugt wird. Verändern sich im Verlauf einer Argumentation die Prämissen einer These, so ist das Framework adaptiv genug, um mit bisher unbekannter Information umgehen zu können.

5 Umgang mit unvollständiger Information

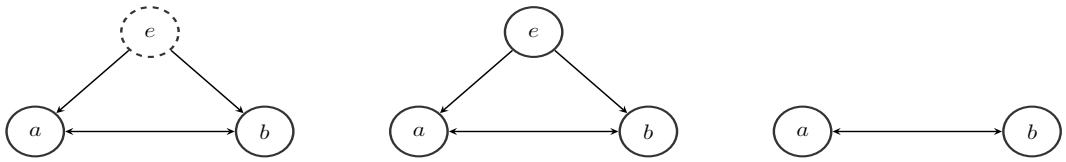
Information ist grundsätzlich unsicher bzw. mit einer gewissen Unsicherheit behaftet. Sicher ist sie nur in speziellen Anwendungsdomänen, in denen Information absolut vorliegt. Konkret auf die ADFs bezogen wird unvollständige Information so betrachtet, dass ein Statement unsicher ist. Es gibt mit den *Incomplete Argumentation Frameworks* (iAF) aus der Arbeit [15] von Baumeister et al. einen Ansatz, wie AFs mit unvollständiger Information umgehen können. Zuerst wird in diesem Kapitel untersucht, ob sich der Ansatz der iAFs auf die ADFs übertragen lässt. Danach wird ein eigener Ansatz vorgestellt, der ADFs befähigt mit unvollständiger Information umzugehen.

5.1 Translation des iAF Ansatzes

Kann das Konzept der iAFs analog auf die ADFs angewendet werden?
Kurz zusammengefasst arbeitet ein iAF mit unterschiedlichen Abschlüssen (*completion*) eines Argumentationsgraphen. Dabei wird ein Abschluss erzeugt, der die unsichere Information beinhaltet und ein anderer Abschluss, in dem diese nicht mehr vorkommt. Die Abbildung 5.1 soll dies illustrieren, das Beispiel ist entlehnt aus [16, Beispiel 1 ff.] übernommen.

In dieser Abbildung 5.1 sind drei Argumentationsgraphen dargestellt: Der ursprüngliche Graph 5.1a mit dem unsicheren Argument e und die Abschlüsse 5.1b und 5.1c. Die Auswertung dieser Abschlüsse (unterschiedlichen Graphen) führt auch zu unterschiedlichen (bevorzugten) Extensionen, nämlich $E_{5.1b} = \{e\}$ und $E_{5.1c} = \{a\}, \{b\}$. Ziel der iAFs ist es, aus den verschiedenen Abschlüssen und Extensionen einen Konsens zu bilden.

Stark vereinfacht ausgedrückt, werden auf der Grundlage von [16] für die iAFs zusätzlich die drei Eigenschaften *deterministisch*, *total* und *funktional* eingeführt. Dabei ist



(a) iAF mit unsicherem Argument e (b) Abschluss Λ_1 mit e ohne dessen Unsicherheit (c) Abschluss Λ_2 ohne Argument e

Abbildung 5.1: Ein iAF mit dem unsicheren Statement e. Die Abschlüsse sind wie Annahmen zu verstehen, bei denen e einmal vorhanden ist und einmal entfernt wurde.

ein Argument deterministisch, wenn sein Status immer derselbe bleibt, d.h., dass eine Entscheidung unabhängig von einer unsicheren Datenbasis getroffen werden kann. Ein Argument ist total, wenn es unter jeder Semantik eine Schlussfolgerung für sich zulässt. Und in der Kombination aus beiden ist ein Argument funktional, wenn es deterministisch und total ist. In der Abbildung 5.1 sind die Argumente a und b funktional, eine Entscheidung über diese Argumente kann unabhängig von e getroffen werden.

Um das AF aus Abbildung 5.1a in ein ADF zu transformieren wird folgende Formel aus [12, Definition 3.15] verwendet:

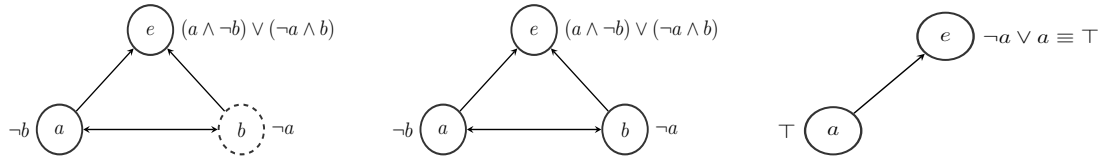
$$\mathcal{C}_a = \bigwedge_{\substack{b \in A, \\ (b,a) \in R}} = \neg b$$

Das entsprechende ADF ist:

$$DF = (\{a, b, e\}, \{(e, a), (e, b), (a, b), (b, a)\}, \\ \{\mathcal{C}_e = \top, \mathcal{C}_a = \neg e \wedge \neg b, \mathcal{C}_b = \neg e \wedge \neg a\})$$

Auch hier können analog zum obigen Verfahren zwei Abschlüsse mit ebenfalls identischen Extensionen gebildet werden. Eine erste Annahme legt nahe, dass sich iAFs auf ADFs übertragen lassen. Folgendes kleines Beispiel in Abbildung 5.2 widerlegt diese Annahme jedoch recht schnell.

Die Abschlüsse Λ_1 und Λ_2 ergeben bei nun unsicherem b zwei semantisch unterschiedliche Graphen. Die Komponenten eines ADFs weisen untereinander Abhängigkeiten auf. Dieses Problem ist in Abschnitt 3.2 herausgearbeitet. Ein weiteres Problem ist,



(a) ADF mit unsicherem Argument b

(b) fiktiver Abschluss Λ_1 ohne Unsicherheit

(c) fiktiver Abschluss Λ_2 ohne b

Abbildung 5.2: Ein ADF kann nicht analog wie ein iAF gehandhabt werden. Das Entfernen von Aussagenvariablen aus den Akzeptanzbedingungen führt zu ungewollten sinnverfälschenden Graphen.

dass es keine Übergangsfunktion gibt, die übereinstimmende Ergebnisse erzeugt, ohne das Information zur Sprache hinzugefügt wird (*language preserving*).
Die iAFs sind nicht ohne zusätzliche Information auf die ADFs übertragbar.

5.2 Funktionsweise der ADFs mit unvollständiger Information

> These: *ADFs können mit unvollständiger Information unter Verwendung der Approximation Fixpoint Theory mittels einer dreiwertigen Logik umgehen, indem alle dreiwertigen Interpretationen in einem Bi-Gitter nach Informations- und Wahrheitswert geordnet und mit einem speziellen Operator ausgewertet werden.*

Ist ein Statement in der dreiwertigen Logik *unbekannt* bzw. *unsicher*, so wird es mit dem nicht-klassischen Wahrheitswert u belegt. Dies führt zur folgenden Definition.

Definition 5.2.1 (nicht-klassische Wahrheitswerte)**.**
Die Wahrheitswerte einer nicht-klassischen Logik sind mehrwertig. Die Menge der nicht-klassischen Wahrheitswerte einer ternären Logik sind $\{true, false, uncertain\}$. Alternativ wird auch $\{t, f, u\}$, $\{T, F, U\}$ $\{wahr, falsch, unsicher\}$, $\{w, f, u\}$ benutzt.

In Abbildung 5.2a ist das Statement b unsicher und wird entsprechend mit $b \mapsto u$ belegt. Dabei bleibt der ursprüngliche Argumentationsgraph erhalten. Die Belegung ist in nachfolgender Tabelle 5.1 dargestellt:

e	a	b	$\mathcal{C}_e$	$\mathcal{C}_a$	$\mathcal{C}_b$
f	f	u	u	u	w
f	w	u	u	u	f
w	f	u	u	u	w
w	w	u	u	u	f

Tabelle 5.1: Auszug der Interpretationen des Graphen aus Abbildung 5.2a, wobei b unsicher ist (linke Seite) und die Auswertung der Akzeptanzbedingungen (rechte Seite).

Für die Auswertung werden in einem initialen Schritt alle möglichen Interpretationen $\mathcal{V}_3$ der Statements $a, b, e \in S$ des ADFs erstellt und in einem Gitter angeordnet.
Bei einem vollständigen Gitter müssen die Bedingungen aus Gleichung 2.2 erfüllt sein. Für eine Belegung mit den nicht-klassischen (ternären) Wahrheitswerten nach Definition 5.2.1 wird die Ordnung nach dem Informationswert $\leq_i$ eingeführt, sodass $u \leq_i t$ und $u \leq_i f$ ist. Das Infimum dieses Gitters ist stets durch die Interpretation $v \in \mathcal{V}_3 : S \mapsto u$ gegeben. Für ein Supremum wird der $\leq_i$-maximale Wert *inkonsistent* (kurz i) benötigt, sodass zusätzlich $t \leq_i i$ und $f \leq_i i$ gilt (vgl. Abbildung 3.5 auf Seite 47). Der Wert i „beinhaltet“ gemäß seiner $\leq_i$-Ordnung die Werte t und f, da i sowohl *wahr* als auch *falsch* sein kann.
Um weiterhin eine dreiwertige Logik verwenden zu können, wird der Wert i weggelassen, indem er durch u „ersetzt“ wird.[29] Ohne Supremum ergibt sich ein (unteres) Halbgitter. Um dennoch eine entsprechende Ordnung mit Supremum zu erhalten, wird ein weiteres Gitter mit der Ordnung nach dem Wahrheitswert $\leq_t$ über das Halbgitter gelegt. Das dadurch entstandene Bi-Gitter $\mathcal{G} = (S, \leq_i, \leq_t)$ erfüllt die Eigenschaften eines Gitters. Jedes Element des Gitters hat einen Nachbarn.

Definition 5.2.2 (Nachbarn)**.**
Sei $\mathcal{G} = (S, P)$ ein Gitter, wobei S eine Menge von Elemente und P eine Ordnung ist. Des Weiteren seien $e_1, e_2 \in S,$ wobei $e_1 < e_2$ und $P = \leq$. Dann sind e_1 und e_2 Nachbarn gdw. kein anderes Element $e \in S$ geordnet nach P zwischen diesen beiden vorkommt.

[29] Treffen die Werte t und f aufeinander ergibt sich der Wert i. Durch den Konsens Operator aus Definition 2.3.3 ergibt diese Konstellation den Wert u.

Beispielsweise ist $\mathcal{G} = (\mathbb{N}, <)$ gegeben, dann sind 1 und 2 Nachbarn, während 3 und 5 keine Nachbarn sind, weil 4 zwischen diesen vorkommt.
Die $\leq_i$-höheren Nachbarn eines Elements des Gitters eines ADFs sind die Approximation eben dieses Elements. In allen Gittern werden benachbarte Elemente durch Linie dargestellt.[30]

Die Auswertung der Interpretationen $\mathcal{V}_3$ mit dem Γ-Operator aus Definition 2.3.4 führt zu folgenden möglichen Ergebnissen (Auswertungsergebnisse):

1) eine Schlussfolgerung kann unabhängig von der Unsicherheit eines Statements getroffen werden, wenn der Γ-Operator ausschließlich klassische Wahrheitswerte $\{t, f\}$ erzeugt oder

2a) eine Schlussfolgerung ist möglich, wenn $s \mapsto u : \Gamma(\mathcal{C}_s) = u$, d.h. wenn nur der Status des unsicheren Statements nach Auswertung der Akzeptanzbedingung unsicher bleibt – die Unsicherheit des Statements s sich nicht auf andere Statements überträgt oder

2b) eine Schlussfolgerung kann mit den Nachbarn (vgl. Definition 5.2.2) des Elements (Interpretation) des Gitters approximiert werden, sodass die Interpretation mit $s \mapsto u$ durch die benachbarten Interpretationen, bei denen s einen Booleschen Wert hat, angenähert wird.

Zu beachten ist, dass ein Fixpunkt des Γ-Operators eine Extension darstellt, die einer bestimmten Semantik entspricht. Die Extension muss im Kontext der Domäne ausgewertet werden, um eine fundierte Entscheidung treffen zu können.

Die Funktionsweise des Umgangs von ADFs mit unvollständiger Information wird an zwei Beispielen demonstriert. Das erste Beispiel 5.2.1 ist an das Party-Beispiel aus [16] angelehnt. Dadurch wird für den allgemeinen Fall einer Argumentation gezeigt, dass ein ADF mit unvollständiger Information umgehen kann. Das zweite Beispiel 5.2.2 bezieht sich auf eine konkrete Anwendungsdomäne. Dabei soll den Lesern ein mögliches Anwendungsgebiet aufgezeigt werden.

[30] Gitter: Abbildung 2.5, Abbildung 5.5 und Abbildung A.3.

5.2.1 Beispiel Party

Es soll eine Party veranstaltet werden, zu der die Gäste Alice (a), Bob (b) und Charly (c) eingeladen sind. Aufgrund einer gemeinsamen Vergangenheit hegen sie gegenseitige Animositäten. Dies führt dazu, dass Alice nicht zur Party erscheint, wenn Bob zur Party geht. Bob wiederum kommt nicht zur Party, falls Alice und Charly kämen. Und Charly ist sich unsicher, ob sie aufgrund der o.g. Konstellation überhaupt an der Party teilnimmt. Diese Statements und Akzeptanzbedingungen werden durch folgendes ADF dargestellt:

$$DF = (\{a, b, c\}, \{(a, b), (b, a), (c, b), (c, c)\},$$
$$\{\mathcal{C}_a = \neg b, \mathcal{C}_b = \neg a \vee \neg c, \mathcal{C}_c = c\})$$

Der dazugehörige Argumentationsgraph ist in Abbildung 5.3 wiedergegeben.

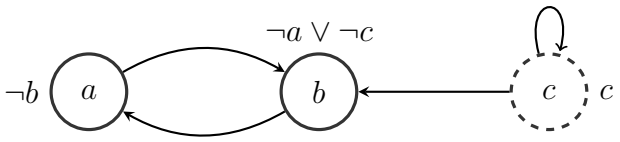


Abbildung 5.3: Der Argumentationsgraph bildet das bestehende Konfliktpotential unter den Partygästen ab

Es sei das Statement c unsicher (dargestellt als gestrichelter Knoten). Die Akzeptanzbedingung von c ist „c“ selbst, wodurch dargestellt wird, dass das Kommen von Charly nur von der eigenen Entscheidung abhängt.
Ein Auszug aller dreiwertigen Interpretationen und der jeweiligen ausgewerteten Akzeptanzbedingung ist in Tabelle 5.2 dargestellt. Die Zeilen, in denen c *unsicher* ist, haben keinen Einfluss auf die Akzeptanz von a und b. Diese Zeilen (Interpretation v_1 und v_2) stellen auf der rechten Seite der Tabelle 5.2 vollständige Extensionen $E_{comp_1} = \{b\}$ und $E_{comp_2} = \{a\}$ dar.

Die vollständigen Extensionen entsprechen dem obigen Auswertungsergebnis unter *Nr. 2a*. Die Interpretationen v_1 und v_2 enthalten das unsichere Statement c und können gemäß *Nr. 2b* approximiert werden. Konkret werden die Nachbarn der Interpretationen v_1 und v_2 des aufgestellten Gitters des ADFs gesucht, wodurch sich folgende Approximationen ergeben:

$$\overbrace{\{a \mapsto f, b \mapsto w, c \mapsto f\}}^{v_3} \leq_t \overbrace{\{a \mapsto f, b \mapsto w, \mathbf{c \mapsto u}\}}^{v_1} \leq_t \overbrace{\{a \mapsto f, b \mapsto w, c \mapsto w\}}^{v_4} \text{ und}$$

$$\underbrace{\{a \mapsto w, b \mapsto f, c \mapsto f\}}_{v_5} \leq_t \underbrace{\{a \mapsto w, b \mapsto f, \mathbf{c \mapsto u}\}}_{v_2} \leq_t \underbrace{\{a \mapsto w, b \mapsto f, c \mapsto w\}}_{v_6}.$$

Bei den Approximationen v_3 bis v_6 aus dem mittleren Teil der Tabelle 5.2 handelt es sich folglich um direkte Nachbarn nach Definition 5.2.2 von v_1 bzw. v_2 im Bi-Gitter $\mathcal{G}$, die die Voraussetzung 2.3 zur Approximation eines Elements im Gitter erfüllen.

	a	b	c	$\mathcal{C}_a$	$\mathcal{C}_b$	$\mathcal{C}_c$
v_1	f	w	**u**	f	w	u
v_2	w	f	**u**	w	f	u
v_3	f	w	f	f	w	f
v_4	f	w	w	f	w	w
v_5	w	f	f	w	w	f
v_6	w	f	w	w	f	w
v_7	u	f	f	w	w	f
v_8	u	f	w	w	f	w
v_9	u	w	f	f	w	f
v_{10}	u	w	w	f	f	w

Tabelle 5.2: Auszug der Interpretationen $\mathcal{V}_3$ mit den jew. ausgewerteten Akzeptanzbedingungen für die Statements a, b, c

Die vollständige Semantik mit unsicherem Statement gibt einem Partyplaner an, dass er (unabhängig von c) mit mindestens einer Person (Bob oder Alice) rechnen muss. Erscheint ihm diese Lösung nicht „exakt“ genug, so kann der unsichere Wert des Statements c approximiert werden. Die sich ergebenen fortgeführten approximierten Interpretationen führen zu bevorzugten Extensionen nach Definition 2.3.5 Nr. 3. Der Partyplaner kann demnach mit maximal zwei Personen rechnen, entweder Bob und Charly (Interpretation v_4: $E_{pref} = \{b, c\}$) oder Alice und Charly (Interpretation v_6: $E_{pref} = \{a, c\}$). Der Vollständigkeit halber ist Interpretation v_3: $E_{pref} = \{b\}$ auch eine mögliche Extension, bei der nur Bob erscheint.

Nun sei, ohne weitere Anpassungen, statt c das Statement a des obigen ADFs unsicher. Der entsprechende Auszug aller $\mathcal{V}_3$ Interpretationen ist im letzten Drittel der Tabelle 5.2 dargestellt. Obwohl das Statement a unsicher ist, können alle Akzeptanzbedingungen mit klassischen Wahrheitswerten ausgewertet werden, d.h. dass die Unsicherheit

des Statements keine Auswirkung auf den Argumentationsgraphen bzw. die Extensionen hat (vgl. Auswertungsergebnis *Nr. 1*).
Diese Extensionen ($E_{v_8} = \{a, c\}, E_{v_9} = \{b\}$) werden den zulässigen Semantiken (vgl. Def. 2.3.5) zugeordnet. Das Prinzip dahinter ist für den Partyplaner einfach: Für jedes Statement, das akzeptiert oder nicht akzeptiert wird, muss es für dessen Status eine Erklärung geben. Der Planer erkennt, dass minimal ein Gast Bob (v_9) oder maximal zwei Gäste Alice und Charly (v_8) zur Party kommen werden.

Dieses Beispiel verdeutlicht, dass eine Aufteilung des ursprünglichen Argumentationsgraphen (unsicheres Statement c) in einen Abschluss ohne unsicheres Statement und einen Abschluss, in dem das Statement als sicher betrachtet wird, nicht notwendig ist.[31] Dabei entsprechen, in Analogie an die iAFs, der Abschluss Λ_1 dem ursprünglichen ADF aus Abbildung 5.3 und der weitere Abschluss dem folgenden ADF:

$$\Lambda_2 = \{a, b\}, \{(a, b), (b, a)\}, \{\mathcal{C}_a = \neg b, \mathcal{C}_b = \neg a\}.$$

Beide ziehen jeweils eine zweiwertige Interpretation ohne unsicheres Statement nach sich. Die bevorzugten Extensionen sind im Fall von Λ_1 identisch mit dem ursprünglichen Graph, bei Λ_2 ändern sich im Vergleich zum obigen Ergebnis die Extensionen. Dies ist eine Folge des angepassten Argumentationsgraphen Λ_2, bei dem das unsichere Statement c nicht mehr vorhanden ist. Die Interpretationen v_1 und v_2 aus Tabelle 5.2 entsprechen ohne Statement c (ohne die Spalten: c und $\mathcal{C}_c$) dem Abschluss Λ_2. Dadurch verändern sich die vormals vorhandenen vollständigen zu bevorzugten Extensionen. Die Schlussfolgerungen, die aus den Semantiken der Abschlüsse Λ_1 und Λ_2 gezogen werden können, entsprechen in diesem Fall ausnahmsweise jenen des ursprünglichen ADFs.

Der Vorteil der ADFs im Umgang mit unvollständiger Information ist, dass durch die dreiwertige Belegung und Auswertung mittels Γ-Operators verschiedene Schritte der Umwandlung und Auswertung eingespart werden können. Insbesondere weil die Transformation des ursprünglichen Graphen, aufgrund der Abhängigkeit und Wechselwirkung der Komponenten untereinander, nicht immer möglich ist. Des Weiteren erfolgt die Betrachtung des Graphen mit unvollständiger Information durch die dreiwertigen Interpretationen im aufgebauten Gitter vollumfänglich und implizit.

[31] Mit Abschluss ist i.S.v. Abschnitt 5.1 ein neuer Graph gemeint.

5.2.2 Beispiel Rechtswesen

Die formale Argumentation als KI-System eignet sich insbesondere im Rechtswesen als Evaluations- und Rechtsfolgeprozess. Diese Anwendungsdomäne ist aufgrund ihres Aufbaus prädestiniert und stellt gleichzeitig ein geeignetes Beispiel für ein mögliches Anwendungsgebiet bereit.

Gesetze werden in Deutschland abstrakt gestaltet, sodass sich Sachverhalte unter die entsprechenden Paragrafen der Gesetze subsumieren lassen. Die Gesetze stellen dabei konkrete Regelwerke für eine Domäne (Zivil-, Straf-, Steuerrecht usw.) dar. Die Subsumtion realer Sachverhalte unter die Tatbestandsvoraussetzungen des jeweiligen Paragrafen benötigt Informationen. Eine Rechtsfolge tritt ein, wenn die nötigen Voraussetzung vorliegen. Dieses Zusammenspiel aus Voraussetzung(en) und (Rechts-) Folge kann durch ein Argumentationsframework nachgebildet werden, wie folgende Modellierung eines Ausschnitts aus dem Einkommensteuergesetz (EStG) zeigt.

Die Tatbestandsvoraussetzungen, ob in Deutschland eine Person einkommensteuerpflichtig ist, regelt §1 Abs. 1 Satz 1 EStG:

> „*Natürliche Personen, die im Inland einen Wohnsitz oder ihren gewöhnlichen Aufenthalt haben, sind unbeschränkt einkommensteuerpflichtig.*“

Gesetzestexte lassen sich häufig als Regel in der Form von Prämissen (Tatbestandsvoraussetzungen) und Konklusion (Rechtsfolge) darstellen. Obiger Gesetzestext wird mittels aussagenlogischer Formeln wie folgt ausgedrückt werden:

$$\begin{aligned}\text{nat. Person} \;\wedge\; \text{Inland} \;\wedge\; (\text{Wohnsitz} \;\vee\; \text{gew. Aufenthalt})\\ \rightarrow\; \text{unbes. einkommensteuerpflichtig}\end{aligned}$$

Übertragen auf ein ADF gibt es fünf Statements, die in direkter bzw. indirekter Relation zueinander stehen. Die Akzeptanzbedingungen werden als $\mathcal{C}$, indiziert mit dem jeweiligen Statement, dargestellt. Die Tatbestandsvoraussetzungen müssen hier akzeptiert sein, damit die Rechtsfolge eintritt.

Die These (Rechtsfolge) e benötigt die Statements (Tatbestandsvoraussetzungen) a, b und c oder d. Der resultierende Graph des ADFs ist in Abbildung 5.4 auf der linken Seite dargestellt. Die Konstruktion eines ADFs aus dem Gesetz ist damit umgesetzt.

Aussagenvariable	**syntaktisches Objekt**	**Akzeptanzbedingung**
a	natürliche Person	$\mathcal{C}_a = \top$
b	Inland	$\mathcal{C}_b = \top$
c	Wohnsitz	$\mathcal{C}_c = \neg d$
d	gewöhnlicher Aufenthalt	$\mathcal{C}_d = \neg c$
e	unbes. einkommensteuerpflichtig	$\mathcal{C}_e = a \wedge b \wedge (c \vee d)$

Ein sich ergebendes Problem sind unvollständige Informationen, wie folgende fiktive Situation zeigt:

Die Verwaltung der Einkommensteuer ist Aufgabe der einzelnen Bundesländer. Der Umzug eines Steuerpflichtigen und der damit einhergehende Transfer von Behördendaten, beispielsweise von Bayern nach Nordrhein-Westfalen, kann dazu führen, dass über den Wohnort und gewöhnlichen Aufenthalt keine Informationen mehr verfügbar sind. Diese Situation ist in Abbildung 5.4 auf der rechten Seite durch die gestrichelten Knoten dargestellt, die andeuten, dass die Information unbekannt bzw. unsicher ist.

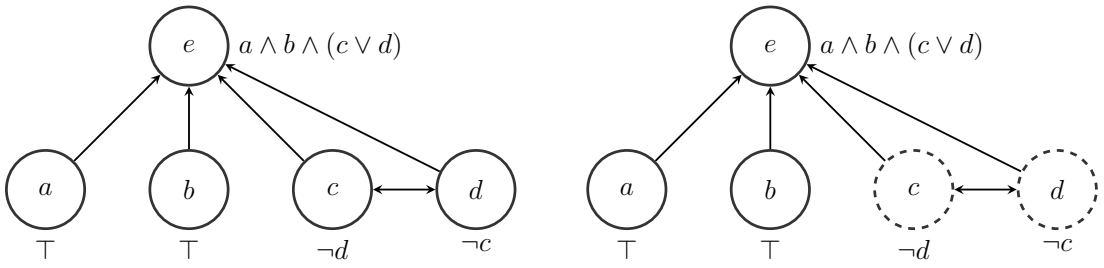


Abbildung 5.4: Der resultierende Graph eines ADFs, um die „Einkommensteuerpflicht“ gem. §1 Abs.1 S.1 EStG darzustellen. Die linke Seite zeigt den vollständigen und die rechte Seite den Graph mit unvollständiger Information.

Eine Betrachtung des Graphen ohne die unsicheren Statements führt zu einem falschen Ergebnis, weil zwingend notwendige Voraussetzungen entfernt würden. Die ADFs müssen folglich mit Unsicherheiten dieser Art umgehen können, um ein praktikables Framework der Domäne zu sein. Wird das obige Framework auf die bereits beschriebene Funktionsweise ausgewertet, so ergeben sich die in Tabelle 5.3 auf der nächsten Seite dargestellten Fixpunkte.

Der $\leq_i$-kleinste Fixpunkt entspricht der grundierten Semantik, die Extension dabei ist $\{a, b\}$. Es soll ein passendes Modell für den §1 Abs.1 Satz 1 EStG gefunden werden,

deshalb wird das Statement e mit dem Wert w belegt (vgl. ab Zeile 3 in Tabelle 5.3). Die Statements c und d sind weiterhin unsicher (mit u belegt), die Auswertung dieser Belegung ergibt keinen Fixpunkt. Die unsicheren Statements werden approximiert, in dem ihre Nachbarn des aufgespannten Gitters gesucht werden, um ein passendes Modell zu finden (siehe unteren Teil der Tabelle 5.3). Die sich ergebenen Modelle

$$w_1 = \{a \mapsto w, b \mapsto w, c \mapsto f, d \mapsto w, e \mapsto w\} \text{ und}$$
$$w_2 = \{a \mapsto w, b \mapsto w, c \mapsto w, d \mapsto f, e \mapsto w\}$$

sind mögliche „Lösungen des ADFs“. Das System vermittelt einem Agenten, dass mit den obigen Approximationen eine Lösung gefunden werden kann. Ein Agent würde in diesem Fall weitere Nachforschungen anstellen.
Das Ergebnis dieser Situation ist offensichtlich: Um die Steuerpflicht abschließend klären zu können, wird die Information über den Wohnsitz respektive den gewöhnlichen Aufenthalt benötigt.

$\leq_i$	a	b	c	d	e	$\mathcal{C}_a$	$\mathcal{C}_b$	$\mathcal{C}_c$	$\mathcal{C}_d$	$\mathcal{C}_e$
2	**w**	**w**	**u**	**u**	**u**	**w**	**w**	**u**	**u**	**u**
3	w	w	u	u	w	w	w	u	u	u
4	w	w	u	f	w	w	w	w	u	f
	w	w	u	w	w	w	w	f	u	w
	w	w	f	u	w	w	w	u	w	f
	w	w	w	u	w	w	w	u	f	w
5	w	w	f	f	w	w	w	w	w	f
	w	**w**	**f**	**w**	**w**	**w**	**w**	**f**	**w**	**w**
	w	**w**	**w**	**f**	**w**	**w**	**w**	**w**	**f**	**w**
	w	w	w	w	w	w	w	f	f	w

Tabelle 5.3: Die erste Spalte gibt den Informationslevel an. Die fett gedruckten Zeilen sind Fixpunkte des ADFs.

Selbstverständlich können beliebig komplizierte Situationen gewählt werden, sodass die Auswertung weniger augenscheinlich ist. Dies erschwert jedoch den Vergleich zwischen den Aktionen eines Menschen und der Auswertung des maschinellen Systems, welche im Idealfall identisch sein sollten. Während der obige Ausschnitt die Machbarkeit der Konstruktion von ADFs aus Gesetzestextes verdeutlicht hat, zeigt der nächste Ausschnitt die Stärke im Umgang mit unvollständiger Information.

Adaption: Betriebsaufgabe

Es soll ein Gewerbebetrieb beendet werden (Betriebsaufgabe). Dazu meldet der Betriebsinhaber den Betrieb ab und erstellt eine Aufgabebilanz für seine Buchführung. In der Bilanz ist das Betriebsvermögen mit den Buchwerten zum Aufgabezeitpunkt aufgeführt. Die Differenz zwischen Buchwert und gemeinem Wert wird als stille Reserve bezeichnet und muss versteuert werden. Insbesondere bei Grundstücken ergeben sich durch Wertsteigerungen der letzten Jahrzehnte hohe stille Reserven. Der Finanzverwaltung ist die Existenz eines (Betriebs-) Grundstücks aus vielerlei Gründen nicht immer bekannt. Das folgende ADF stellt diese fiktive Situation dar, wobei a die Rechtsfolge und b die Prämisse der Betriebsaufgabe sowie c die stille Reserve in Form eines Betriebsgrundstücks ist:

$$DF = (\{a, b, c\}, \{(b, a), (c, a)\}, \\ \{\mathcal{C}_a = b \vee (b \wedge c), \mathcal{C}_b = \top, \mathcal{C}_c = c\})$$

Die ADFs erlauben eine Auswertung, bei der eine Belegung mit dem Wert u vorkommen kann. Dabei kann ein Fixpunkt gefunden werden, der ein unsicheres Statement enthält, wie nachstehende Tabelle zeigt. Dies ist gegenüber anderen Frameworks ein Vorteil, da keine Umformungen vorgenommen werden müssen.

a	b	c	$\mathcal{C}_a$	$\mathcal{C}_b$	$\mathcal{C}_c$
w	w	**u**	w	w	**u**
w	w	f	w	w	f
w	w	w	w	w	w

Der Fixpunkt aus Zeile 1 der Tabelle entspricht der grundierten Semantik (Auswertungsergebnis *Nr. 2a*). Ein Agent glaubt hierbei nur was er auch verteidigen kann. Dies ist jedoch kein Modell des ADFs nach Gleichung 2.4, weil das Statement c unsicher ist. Der Agent könnte den Sachverhalt abschließen, was eine akzeptable Lösung ist, oder ein passendes (zweiwertiges) Modell durch Approximation von c finden. Im letzteren Fall könnten stille Reserven aufgedeckt werden, die pro-fiskalisch zu einer höheren Besteuerung führen. Ähnlich würde ein erfahrener Sachbearbeiter der Finanzverwaltung bei einem gleichgelagerten Sachverhalt agieren.

5.3 Eigenschaften und Komplexität

In diesem Abschnitt werden die Eigenschaften der ADFs im Umgang mit unvollständiger Information zusammengefasst. Es wird die Funktionsweise beschrieben und gezeigt, wie eine Auswertung mit unsicheren Statements via Approximation erfolgt. Abschließend wird die Komplexitätsfrage der Schlussfolgerungsprobleme der ADFs geklärt. Hierbei wird betrachtet, wie effizient die ADFs leichtgläubige (*credulous*) und skeptische (*skeptical*) Schlussfolgerungen lösen können.

5.3.1 Eigenschaften

Es existieren unterschiedliche Ansätze, um unvollständige Information darzustellen. Die ADFs nutzen zum Umgang mit fraglichen Existenzen von Statements eine dreiwertige Logik.[32] Die klassischen Wahrheitswerte aus Definition 2.2.3 werden um den Wert *unsicher* (kurz u) ergänzt. Dadurch kann ein Statement mit einem nicht-klassischen Wahrheitswert nach Definition 5.2.1 belegt werden, insbesondere wenn das Vorhandsein des Statements unsicher ist.

Es treten dadurch zwei Arten von Interpretationen auf:

- I bzw. $\mathcal{V}_2$, bei denen nur Belegungen mit klassischen Wahrheitswerten enthalten sind und
- $\mathcal{V}_3$, bei denen auch Zuordnungen mit dem nicht-klassischen Wahrheitswert u vorkommen.

Eine charakteristische Funktion der ADFs muss entsprechend angepasst werden, um mit einer dreiwertigen Logik umgehen zu können. Zuerst wird der Konsens-Operator Π aus Definition 2.3.3 eingeführt. Dieser wird vom Γ-Operator aus Definition 2.3.4 verwendet, um zwei- und dreiwertige Interpretationen auszuwerten.

Vorab werden alle Statements der Menge S eines ADFs (vgl. Definition 2.3.1) mit allen möglichen Kombinationen der nicht-klassischen Wahrheitswerte belegt. Das entspricht allen $3^{|S|}$ Interpretationen aus $\mathcal{V}_3 \supseteq \mathcal{V}_2$.

[32] Den ADFs liegt eine possibilistische Logik zugrunde [17].

Sei nun ein einziges Statement gegeben $a \in S$, dann gibt es 3^1 mögliche Interpretationen $v \in \mathcal{V}_3$, nämlich $\{a \mapsto u\}, \{a \mapsto f\}, \{a \mapsto w\}$. Diese Interpretationen lassen sich nach ihrem Informationsgehalt $\leq_i$ ordnen, sodass sich folgende aufsteigende Ordnung ergibt: $a \mapsto u$, dann $a \mapsto f$ auf demselben Informationslevel (kurz: *lvl*) wie $a \mapsto t$. Folglich ergibt sich eine partiell geordnete Menge, die bei Erfüllung der Bedingung 2.2 ein Gitter darstellt. Konkret ergibt sich mit $\mathcal{G} = (S, \leq_i)$ ein Halbgitter.

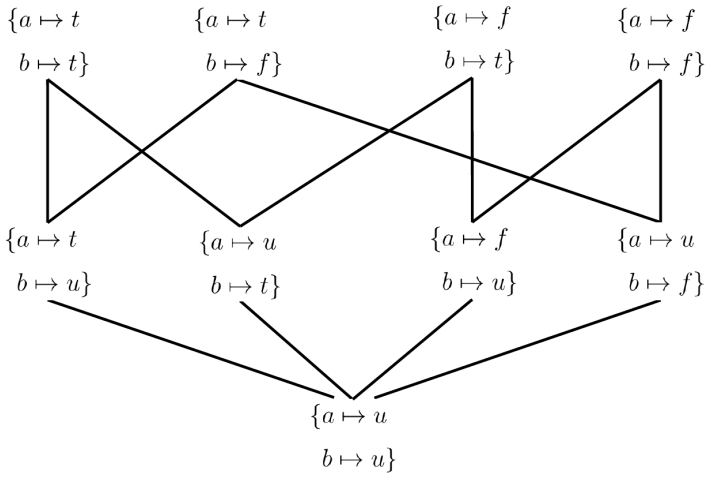


Abbildung 5.5: Das erzeugte Bi-Gitter eines ADFs mit zwei Statements. Die Nachbarn sind durch Linie miteinander verbunden. Für eine Approximation werden die Nachbarn auf dem $\leq_i$-höheren Level benötigt. Diese sind wiederum nach dem Wahrheitswert $\leq_t$ geordnet, sodass die Approximationsgleichung 2.3 erfüllt wird. alternativ A.2

Sei nun $S = \{a, b\}$, dann gibt es die 3^2 Interpretationen aus folgender Tabelle 5.4:

Informationsgehalt	Interpretation
lvl_0	$\{a \mapsto u, b \mapsto u\}$
lvl_1	$\{a \mapsto f, b \mapsto u\}$, $\{a \mapsto t, b \mapsto u\}$, $\{a \mapsto u, b \mapsto f\}$, $\{a \mapsto u, b \mapsto t\}$
lvl_2	$\{a \mapsto f, b \mapsto f\}$, $\{a \mapsto f, b \mapsto t\}$, $\{a \mapsto t, b \mapsto f\}$, $\{a \mapsto t, b \mapsto t\}$

Tabelle 5.4: Die Tabelle enthält alle Interpretationen einer ternären Logik. In der ersten Spalte ist das Informationslevel und in der zweiten Spalte sind die dazugehörigen Interpretationen eines Gitters dargestellt.

Auf dem niedrigsten Informationslevel (lvl_0) gibt es stets nur eine einzige Interpretation, diejenige bei der allen Statements der Wert u zugewiesen wird. Auf dem höchsten Informationslevel ($lvl_{|S|}$) gibt es $2^{|S|}$ Interpretationen, dabei erfolgt die Belegung wie in der klassischen Logik. In Abhängigkeit von der Anzahl der Statements gibt es unterschiedlich viele Informationslevel mit verschieden vielen Interpretationen zwischen lvl_0 und $lvl_{|S|}$. Die Formel 5.1 gibt den Aufbau der Informationslevel und deren Anzahl an Interpretation wieder. Der Laufindex i steht für den Level.

$$1 + \sum_{i=1}^{i=|S|} \binom{|S|}{i} * 2^i \tag{5.1}$$

Dies kann exemplarisch mit Tabelle 5.4 nachvollzogen werden. Visuell dient Abbildung 5.5 zur Veranschaulichung des Bi-Gitters, bestehend aus zwei Statements.

Das Beispiel Party 5.2.1 auf Seite 62 hat drei Statements $S = \{a, b, c\}$, also $3^3 = 27$ Interpretationen. Diese sind in $|S| + 1$ Informationslevel aufgeteilt, dabei ist: $lvl_0 = 1$, $lvl_1 = 6$, $lvl_2 = 12$, $lvl_3 = 8$.

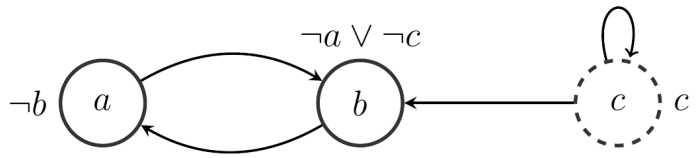


Abbildung Whd-5.1: Der Graph aus Beispiel 5.2.1

Durch die Ordnung der Menge nach ihrem Informationsgehalt $\leq_i$ entsteht ein Halbgitter. In diesem Gitter ist das Infimum, die größte untere Schranke, spätestens das Element auf dem niedrigsten Informationslevel lvl_0. Der Γ-Operator erhält die Interpretationen $v \in \mathcal{V}_3$ in aufsteigender $\leq_i$-Reihenfolge zur Auswertung. Die Auswertung erfolgt anhand der Akzeptanzbedingung der Statements $\{\mathcal{C}_s\}_{s \in S}$. Werden Belegungen mit unterschiedlichen Wahrheitswerten ausgewertet, wird der Konsens-Operator verwendet, der für zwei Elemente das Infimum aus dem Halbgitter ergibt.[33]

Stimmen Interpretation und Auswertung mittels Γ-Operator überein, ist ein *Fixpunkt* gefunden. Fixpunkte können den Semantiken aus Definition 2.3.5 zugeordnet werden,

[33] Hierbei ergeben t und f nicht den Wert i, sondern den Wert u, weshalb der Einsatz einer dreiwertigen Logik ausreichend ist.

außerdem lassen sich die entsprechenden Extensionen entnehmen.
Konkret ist im Beispiel Party 5.2.1 das Statement c unsicher. Dennoch werden zuerst alle Interpretation $\mathcal{V}_3$ gebildet und nacheinander ausgewertet.[34] Dieses Vorgehen ist notwendig, um die Semantiken der ADFs korrekt zu ermitteln.

Die Semantiken und ihre Extensionen eines ADFs geben Lösungsmengen vor. Sogenannte *Agenten* simulieren nach [8] Verhaltensweisen, die sich bestimmten Praktiken zuordnen lassen, um zu einer Schlussfolgerung zu gelangen:
Beispielsweise ist ein skeptischer Agent so formuliert, dass er der grundierten Semantik entspricht. Schließlich wird die Lösungsmenge gesucht, die am wenigsten fragwürdig ist. Das heißt, dass wirklich nur die Statements akzeptiert werden, deren Akzeptanz auch unumgänglich ist.
Ein leichtgläubiger Agent ist derart formuliert, dass auf ihn die bevorzugte Semantik zutrifft. Es wird eine Lösungsmenge gesucht, die so viele Statements akzeptiert, wie es vernünftig ist. Das heißt, dass auch Alternativen in Betracht gezogen werden, was zu unterschiedlichen Extensionen führen kann. Zum Beispiel kann ein gegenseitiger (sich ausschließender) Angriff dadurch vernünftig gelöst werden, dass in einer Extension das eine Statement vorkommt und in einer anderen Extension folglich das widersprüchliche Statement, jedoch nie beide zusammen.

Ist das Vorhandensein eines Statements unsicher, kann trotzdem eine Auswertung erfolgen. Im Beispiel 5.2.1 gilt das Statement c als unsicher. Der $\leq_i$-kleinste Fixpunkt wird mit der Interpretation des niedrigsten Informationslevels gefunden. Ein skeptischer Agent, der die grundierte Semantik nutzt, hat stets die Extension $E_{grou} = \emptyset$ als Lösungsmenge zur Verfügung.
Abseits der trivialen Lösung (leere Menge) gibt es zwei Interpretationen mit $c \mapsto u$, die Fixpunkte erzeugen (vgl. Interpretation v_1 und v_2 aus Tabelle 5.2 auf Seite 63). Die Extensionen sind $E_{comp_1} = \{b\}$ und $E_{comp_2} = \{a\}$. Die vollständige Semantik kann als Präzisierung der zulässigen Semantik verstanden werden. Für jedes akzeptierte Statement muss es eine Begründung geben und darüber hinaus darf ein Statement nur unsicher bleiben, wenn es wirklich keine Gründe für eine andere Belegung $\{w, f\}$ gibt. Das heißt, dass ein Statement weder den Status *akzeptiert* noch *nicht akzeptiert* nur hat, wenn es tatsächlich keinen Grund gibt, ihm einen anderen Status zu zuordnen.

[34] Der Aufbau eines Gitters kann exemplarisch an Abbildung 5.5 betrachtet werden, ein vollständiges Gitter mit Auswertung ist in Abbildung A.3 auf Seite 99 dargestellt.

Für einen Agent, der Unsicherheit(en) akzeptiert, ist die Lösungsmenge plausibel. Ist nichts über c bekannt, dann ist hier entweder a oder b eine Lösung. Für einen Agent, der es genauer wissen will (*investigating*), ist diese Lösung unzureichend. Bisher wurde nichts über c ausgesagt, obwohl c approximiert werden kann.

Das (Halb-) Gitter wurde bislang mit der Ordnung nach dem Informationsgehalt $\leq_i$ betrachtet, dabei ist eine zusätzliche zweite Ordnung nach dem Wahrheitswert $\leq_t$ möglich. Dadurch erweitert sich das Gitter zu einem Bi-Gitter $\mathcal{G}$. Enthält ein Fixpunkt ein Statement, dessen Wert *unsicher* ist, so kann die Interpretation, die dieses unsichere Statement enthält, durch seine Nachbarn im Gitter $\mathcal{G}$ approximiert werden. Benachbarte Elemente einer bestimmten Interpretation sind nach Definition 5.2.2 diejenigen Elemente des Gitters, die nach der vorgegebenen Ordnung direkt auf die gegebene Interpretation folgen. Die benachbarten Interpretationen einer Interpretation mit unsicherem Statement liegen stets mindestens ein Informationslevel höher im Gitter. Das Bi-Gitter $\mathcal{G}$ ermöglicht mit der Ordnung nach dem Wahrheitswert $\leq_t$ eine Approximation nach Gleichung 2.3. Die Belegung $c \mapsto u$ wird bspw. durch die Nachbarn des Gitters $c \mapsto w$ (obere Schranke) und $c \mapsto f$ (untere Schranke) angenähert. Dies wird entsprechend für E_{comp_1} und E_{comp_2} aus dem Beispiel 5.2.1 durchgeführt (vgl. Interpretationen v_3 bis v_6 in Tabelle 5.2). Tatsächlich erwachsen aus den Approximationen drei Fixpunkte mit folgenden Extensionen: $E_{pref_1} = \{b\}$, $E_{pref_2} = \{b, c\}$ und $E_{pref_3} = \{a, c\}$. Diese stellen die Lösungsmenge für einen leichtgläubigen Agent dar.

Ist das Statement c unsicher, dann können die Lösungen wie folgt interpretiert werden: Bei unsicherem c gilt entweder $E_{comp_1} = \{b\}$ oder $E_{comp_2} = \{a\}$. Ist c approximiert, dann gilt $E_{pref_1} = \{b\}$, $E_{pref_2} = \{b, c\}$ oder $E_{pref_3} = \{a, c\}$. Das Statement a kommt in diesen möglichen Extensionen zweimal vor, das Statement b dreimal und das Statement c wiederum zweimal. Die Chance, dass das Statement b akzeptiert wird, liegt bei 60% und bei den Statements a, c bei 40%. Genauso kann das leichtgläubige oder skeptische Schlussfolgern der Argumentation zur Entscheidungsfindung eingesetzt werden.

Eine andere Anfrage an ein entsprechendes maschinelles System könnte ähnlich zu Beispiel Rechtsweisen 5.2.2 auf Seite 65 sein (in Abbildung Whd-5.2 erneut dargestellt). Die Wissensbasis des ADFs enthält den vollständig konstruierten Argumentationsgraph. Unklar ist, ob mit den bereits vorliegenden (unvollständigen) Informationen eine Schlussfolgerung gezogen werden kann. Das System bekommt die Interpretati-

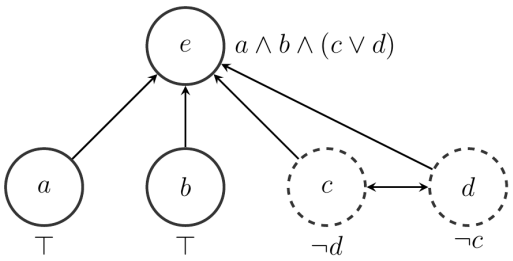


Abbildung Whd-5.2: Der Argumentationsgraph des Beispiel 5.2.2

on entsprechend der vorhanden Daten und der Anfrage zur Akzeptanz einer These (hier: Statement e) als Eingabe. In diesem Fall ergibt dies die folgende Interpretation: $\{a \mapsto w, b \mapsto w, c \mapsto u, d \mapsto u, e \mapsto w\}$.

Anders als zuvor ergibt die Auswertung keine vollständige Extension mit einem unsicheren Statement. Dadurch ist ersichtlich, dass eine Lösungsmenge ohne Approximation nicht gefunden werden kann. In der Tabelle 5.3 auf Seite 67 ist die durchgeführte Näherung abgebildet. Die Fixpunkte entsprechen der bevorzugten Semantik. Die Extensionen $E_{pref_1} = \{a, b, d, e\}$ und $E_{pref_2} = \{a, b, c, e\}$ sind Lösungen zur Anfrage. Ein mögliches Fazit ist, dass ohne die unsicheren Statements c und d keine Entscheidung getroffen werden kann.

Zum Abschluss des Abschnitts wird die Adaption des Beispiels 5.2.2 (Betriebsaufgabe) betrachtet. Die Eingabe an das ADF ist folgende Interpretation $\{a \mapsto w, b \mapsto w, c \mapsto u\}$. Tatsächlich wird ein Fixpunkt der vollständigen Semantik $E_{comp} = \{a, b\}$ gefunden. Ein investigativer Agent nimmt dies als Anhaltspunkt, um den Wert von Statement c zu approximieren. Dadurch ergibt sich die Lösungsmenge aus $E_{pref_1} = \{a, b\}$ und $E_{pref_2} = \{a, b, c\}$. Daraus folgt, dass eine Aufklärung zur Existenz des Statements c sinnvoll ist.

Die wesentliche Eigenschaft der ADFs im Umgang mit unvollständiger Information ist, dass die ADFs keine zusätzlichen Mittel benutzen müssen. Der ursprüngliche Argumentationsgraph bleibt erhalten, insbesondere wird keine exponentielle Anzahl weiterer Graphen mit verschiedenen Abschlüssen benötigt. An der ursprünglichen Wissensbasis des Frameworks werden für die Auswertung keine Manipulationen vorgenommen. Dennoch können Statements, deren Existenz unsicher ist, wie beschrieben ausgewertet werden. Ein dabei gefundener Fixpunkt (i.d.R. vollständige Semantik) stellt eine Lösung dar, d.h, dass unsichere Statements in Fixpunkten vorkommen können, ohne

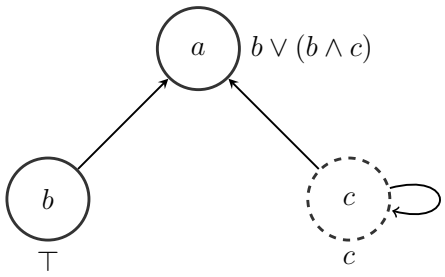


Abbildung 5.6: Die Rechtsfolge der Betriebsaufgabe (a) tritt bei Beendigung des Gewerbebetriebs (b) ein. Zudem können weitere ggf. unsichere Prämissen, bspw. ein Betriebsgrundstück (c), die Rechtsfolge zusätzlich prägen.

dass sie die Lösungsfindung stören. Das unsichere Statement eines Fixpunkts ist für die entsprechende Extension ohne Bedeutung. Diese erste Lösung kann in einem nächsten Schritt approximiert werden, indem der Belegung des unsicheren Statements die Booleschen Werte zugewiesen werden, welche genau den Nachbarn des aufgestellten Gitters entsprechen. Enthält die erneute Auswertung wieder einen Fixpunkt, dann wird die bisherige Lösungsmenge (ohne das unsichere Statement) um die neue Menge mit dem approximierten Statement erweitert. Die entstandene Lösungsmenge kann je nach Anwendungsdomäne genutzt werden, um eine Entscheidung zu treffen.
Es existiert immer ein Fixpunkt der grundierten Semantik, da der Γ-Operator $\leq_i$-monoton wachsend ist [10, S. 805]. Jedes ADF verfügt des Weiteren über mindestens eine zulässige, vollständige und bevorzugte Extension [12, S. 256]. Das triviale Ergebnis ist dabei die leere Menge.

5.3.2 Komplexität

Die Komplexität der Schlussfolgerungsprobleme gibt Aufschluss darüber, wie effizient eine Schlussfolgerungsaufgabe bei den ADFs gelöst werden kann. Dies ist für die ADFs ausführlich von Dvorák und Dunne in [18] ausgeführt. Die wesentlichen Ergebnisse daraus werden im Folgenden kurz wiedergegeben.

Die Konstruktion eines Argumentationsgraphen und dessen Auswertung mittels Γ-Operator erzeugt verschiedene Extensionen, die den Argumentationssemantiken nach Definition 2.3.5: $\varsigma = \{\text{vollständig}, \text{zulässig}, \text{bevorzugt}, \text{grundiert}\}$ zugewiesen werden.

Diese Menge an Extensionen wird benutzt, um eine Schlussfolgerung über die Statements zu treffen. Dabei können Statements relevant sein, die in einer einzigen oder in allen Extensionen vorkommen. Ersteres wird als leichtgläubiger Schluss (*credulous reasoning*) und letzteres als skeptischer Schluss (*skeptical reasoning*) bezeichnet. Die Entscheidungsprobleme können präzise wie folgt zusammengefasst werden:

- Ein *leichtgläubiger Schluss* liegt vor, wenn ein vorgegebenes Statement in wenigstens einer Extension vorkommt.
  Eingabe: Ein ADF $DF = (S, L, \mathcal{C})$ und ein Statement $a \in S$.
  Anfrage: Gibt es eine Interpretation $I \in \varsigma(DF)$ mit $I(a) = t$?

- Ein *skeptischer Schluss* liegt vor, wenn ein Statement in allen Extensionen vorkommt.
  Eingabe: Ein ADF $DF = (S, L, \mathcal{C})$ und ein Statement $a \in S$.
  Anfrage: Ist $I(a) = t$ für jede Interpretation $I \in \varsigma(DF)$?

Neben den beiden genannten Schlussfolgerungsproblemen existieren auch weitere Probleme, wie z.B. die Suche nach der Existenz einer (nicht trivialen) Lösung, die in dieser Arbeit nicht weiter betrachtet wird. Ergänzend zu obiger Aufzählung kommt die Überprüfung (*Verification*) hinzu:

- Verifikation ist erfolgreich, wenn die gegebene Interpretation einer Semantik (Extension) des ADFs entspricht.
  Eingabe: Ein ADF $DF = (S, L, \mathcal{C})$ und eine Interpretation I.
  Anfrage: Ist eine geg. Interpretation I auch eine Extension des ADFs; $I \in \varsigma(DF)$?

Um die Frage nach der Komplexität zu beantworten, werden die obere und die untere Grenze der Laufzeit eines Problems gesucht. Eine obere Grenze ist gefunden, wenn es einen Algorithmus gibt, der das Problem innerhalb einer bestimmten Komplexitätsklasse löst oder wenn sich das Problem auf ein anderes Problem reduzieren lässt, das bereits nachgewiesen in einer Klasse K liegt. Für eine untere Grenze wird ein anderes Problem P′ gesucht, das sich auf das aktuelle Problem reduzieren lässt. Das Problem P′ ist dabei in einer Komplexitätsklasse K-schwer (K-*hard*), d.h., dass das Problem P′ genauso schwer ist wie alle anderen Probleme in dieser Komplexitätsklasse. Stimmen die beiden Grenzen überein, so ist das Problem exakt der Komplexitätsklasse K-vollständig (*complete*) zuzuordnen.

Um einen Algorithmus für die obere Grenze zu finden, wird die Standardmethode „*guess and check*“ verwendet. Zuerst rät ein solcher Algorithmus nicht-deterministisch eine mögliche Extension und überprüft anschließend, ob es sich dabei tatsächlich um eine gewünschte Lösung handelt.
Analog wird für das leichtgläubige Schlussfolgern verfahren. Zuerst wird eine Menge an Statements zufällig ausgewählt, um danach zu prüfen, ob die Menge eine Extension der betrachteten Semantik ist. Die Antwort fällt positiv aus, wenn das betrachtete Statement in mindestens einer Auswahl zu *wahr* ausgewertet wird. Die Komplexität für die Verifikation einer Extension ist V, sodass ein Algorithmus für das leichtgläubige Schlussfolgern in NP^V liegt.
Für das skeptische Schlussfolgern ist es einfacher, das komplementäre Problem zu betrachten und dafür einen Algorithmus zu finden, in diesem Fall, dass ein Statement nicht unter die skeptische Schlussfolgerung fällt. Wieder wird zufällig eine Menge an Statements ausgewählt und anschließend überprüft, ob es sich um eine Extension handelt, die das betrachtete Statement nicht enthält. Die Antwort ist *ja*, wenn jede Auswahl zu $falsch$ ausgewertet wird. Der Algorithmus für das skeptische Schlussfolgern liegt in $coNP^V$, wobei V wieder die Komplexität der Verifikation bezeichnet.
Über die Reduzierung mittels *standard translation* lässt sich zeigen, dass diese beiden oberen Grenzen bereits optimal sind.

Ob ein Kandidat (*guess*) zutreffend ist, wird mit der Verifikation geprüft. Anhand der Arbeit [10] von Brewka, Strass et al. erfolgt die Betrachtung der Komplexität für die Verifikation.
Für die grundierte Semantik wird das Verifikationsproblem auf das DP-vollständige Problem *SAT-UNSAT* reduziert. Eine Instanz (ϕ, ψ) des SAT-UNSAT Problems besteht aus einer aussagenlogischen Formel ϕ, für die zu prüfen ist, ob sie erfüllbar ist und einer aussagenlogischen Formel ψ, für die zu prüfen ist, ob sie unerfüllbar ist.

> „Die Verifikation, ob eine dreiwertige Interpretation einer grundierten Semantik eines gegebenen ADFs entspricht, ist DP-vollständig“ [10, Theorem 6].

Auf diesem Ergebnis aufbauend, folgt direkt das Korollar zu der vollständigen Semantik.

> „Die Verifikation, ob eine dreiwertige Interpretation einer vollständigen Semantik eines gegebenen ADFs entspricht, ist DP-vollständig“ [10, Corollary 7].

Die Skizze des Beweises für die Verifikation der zulässigen Semantik kann aus [10, S. 807] entnommen werden. Dadurch ergibt sich folgende Aussage.

> „Die Verifikation, ob eine dreiwertige Interpretation einer zulässigen Semantik eines gegebenen ADFs entspricht, ist coNP-vollständig“ [10, Proposition 10].

Wird die Verifikation für eine zulässige Semantik mit der Standardmethode „*guess and check*“ kombiniert, ergibt sich für die Prüfung, ob bei gegebener dreiwertiger Interpretation eine bevorzugte Semantik vorliegt, die Einordnung in die Komplexitätsklasse $\prod_2^P$ -vollst. [12, S. 676].

	leichtgläubiger Schluss	skeptischer Schluss	Verifikation
grundiert	$coNP$-vollst.	$coNP$-vollst.	DP-vollst.
zulässig	$\sum_2^P$ -vollst.	trivial	$coNP$-vollst.
vollständig	$\sum_2^P$ -vollst.	$coNP$-vollst.	DP-vollst.
bevorzugt	$\sum_2^P$ -vollst.	$\prod_3^P$ -vollst.	$\prod_2^P$ -vollst.

Tabelle 5.5: Komplexität der Schlussfolgerungsaufgaben der ADFs, übernommen aus [18, Table 10]

„Die Komplexitätsergebnisse der Tabelle 5.5 sind für nicht triviale Schlussfolgerungen ein Level in der Polynom Hierarchie (PH) höher als bei den AFs. Ursächlich ist der Γ-Operator, der anders als die charakteristische Funktion nicht in Polynomialzeit auswertet, sondern in NP bzw. $coNP$.“ [18, S. 675]
Die dargestellten Verifikationsergebnisse sind Bestandteil des leichtgläubigen und des skeptischen Schlussfolgerungsproblems.

Fraglich ist, ob die Approximation eines unsicheren Statements zusätzliche Zeit beansprucht und sich die Komplexität somit weiter erhöht.
Über die Auswertung des Γ-Operators werden unsichere Statements bereits mit berücksichtigt. In einem initialen Schritt wird ein Bi-Gitter $\mathcal{G}$ mit allen Interpretation $\mathcal{V}_3$ eines ADFs erzeugt. Dieses beinhaltet auch unsichere Statements, sodass kein extra Schritt zur Approximation durchgeführt werden muss. Folglich benötigt die in dieser Arbeit beschriebene Approximation der unsicheren Statements keine zusätzlich Zeit. Die Komplexität der Schlussfolgerungsprobleme ist wie bei den originären ADFs.

Die aufgestellte *These* kann gehalten werden. Ein ADF ist derart beschaffen, dass der Umgang mit unvollständiger Information implizit gegeben ist. Initial findet eine Kombinatorik aus allen Statements $s \in S$ statt, sodass alle dreiwertigen Interpretationen $\mathcal{V}_3$ betrachtet werden. Konkret wird jedes Statement auch im Fall einer möglichen Unsicherheit ($s \mapsto u$) ausgewertet. Diese Auswertung erfolgt mit dem Γ-Operator (Def. 2.3.4), der nach Brewka et ali. in [12, Corollary 3.6] als „*ultimative Approximation*“ des Operators G (Def. 2.3.2) bezeichnet wird. Die dort beschriebene Approximation gelingt, wenn der Γ-Operator die geg. Interpretation v erweitert, sodass die zweiwertigen Interpretationen von v berücksichtigt werden.

Dieses Kapitel hat insbesondere die Funktionsweise der ADFs im Umgang mit unvollständiger Information in Form von unsicheren Statements untersucht. Dazu wurde das nach dem Informationswert geordnet Gitter mit der Ordnung nach dem Wahrheitswert kombiniert. Die in diesem Bi-Gitter angeordneten Interpretationen (Elemente) schaffen die Möglichkeit zur Approximation eines unsicheren Statements nach Gleichung 2.3. Die Unsicherheit eines Statements kann verschiedene Auswirkungen auf die Auswertung haben. Dies wurde mit den *Auswertungsergebnissen* und Beispielen (u.a. Party 5.2.1, Rechtswesen 5.2.2) gezeigt. In Abhängigkeit des genutzten Agenten können auch Extensionen „mit Unsicherheiten“ plausible Lösungsmengen darstellen. Ist das Vorhandensein eines Statements s nicht garantiert, so kann die entsprechende Interpretation v mit $s \mapsto u$ approximiert werden. Dazu werden zuerst die $\leq_i$-höheren benachbarten und anschließend die nach der Ordnung $\leq_t v$ einschließenden Interpretationen des Bi-Gitters verwendet. Die dadurch entstehende Approximationsmenge weist je nach Anzahl der unsicheren Statements und gewünschten Annäherung unterschiedlich viele Elemente (Interpretationen) auf.

Die ADFs bieten ein gutes Konzept zum Umgang mit unvollständiger Information an. Dadurch eröffnen sich viele interessante Anwendungsgebiete (siehe später Diskussion 6.2). In diesem Abschnitt wurde der Vorteil der ADFs in Bezug auf unsichere Statements herausgearbeitet und dargestellt.

6 Zusammenfassung

In dieser Arbeit wurde der Umgang der formalen Argumentation, konkret in der Form des abstrakt dialektischen Argumentierens, mit unvollständiger Information untersucht. Die Nachbildung einer Argumentation für ein maschinelles System erfolgte mit dem abstrakt dialektischen Framework (ADF). Die formale Grundlage bilden, neben den ADFs selbst, die abstrakten Argumentation Frameworks (AFs) von Dung und die Aussagenlogik (AL). Mit den ADFs ist die Konstruktion einer Argumentation möglich, deren Argumente bzw. Statements in unterschiedlichen Relationen zueinander stehen können. Die Akzeptanz eines Statements hängt dabei ausschließlich von dessen Akzeptanzbedingung ab. Für jedes einzelne Statement definiert diese Bedingung die Relation zwischen eben diesem Statement und seinen direkten Vorgängern bzw. Eltern. Dadurch entsteht die für das Framework namensgebende Dialektik.

Ist die Existenz von Statements oder Angriffen nicht sicher gegeben, führt dies im Kontext der formalen Argumentation zu Problemen. Konkret betrifft dies die Konstruktion des Argumentationsgraphen eines ADFs bei unvollständiger Information. Für diesen Fall wurde analysiert, welche Auswirkung das Fehlen von Elementen an den Komponenten (Statement, Link, Akzeptanzfunktion) eines Graphen hat. Ebenso wurde betrachtet, wie sich die Unsicherheit eines Statements auf das Framework auswirkt. Die Ergebnisse werden im Folgenden zusammengefasst vorgestellt.

6.1 Ergebnisse

Eine zentrale Feststellung dieser Arbeit ist, dass die Komponenten der ADFs in Wechselwirkung zueinander stehen. Aufgrund der Dialektik ist es nicht möglich ein Statement oder einen Angriff ohne weiteres zu entfernen bzw. hinzuzufügen. Die Veränderung an einer Komponente zieht grundsätzlich Anpassungen an anderen Komponenten nach sich.

Daraus folgt direkt die Feststellung, dass iAFs nicht ohne weitere Informationen auf ADFs übertragbar sind (*language preserving*).

Dungs AFs liegt die Idee zugrunde, dass sich Argumente stets gegenseitig angreifen bis eine Menge an Argumenten *zuletzt* übrig bleibt. Ist die Existenz eines Arguments unsicher, so kann es unter einem Abschluss (*completion*) aus dem Argumentationsgraphen entfernt werden.
Bei den ADFs wird die Beziehung der Statements zueinander erst über die Akzeptanzbedingungen festgelegt. Ein unsicheres Statement kann nicht einfach aus dem Graph entfernt werden, weil neue Abschlüsse des ursprünglichen Graphen dann nicht mehr semantisch äquivalent sind.

Konstruktion des Argumentationsgraphen bei unvollständiger Information

Um bei unvollständiger Information ein ADF konstruieren zu können, ist es entscheidend, an welcher Komponente die Unvollständigkeit auftritt. Beispielsweise sind Informationen bzgl. der Komponenten der Links redundant vorhanden und können bei Unvollständigkeit abgeleitet werden. Eine Unvollständigkeit an den anderen beiden Komponenten oder in Kombination macht die Konstruktion eines Graphen immer komplexer bis zu einem Punkt, an dem keine sinnvolle Lösung mehr gefunden werden kann.

Mit diesem Gedanken wurde die Komponente der Akzeptanzfunktion genauer betrachtet. Die einzelnen Akzeptanzbedingungen besitzen den größten Informationsgehalt zum Ableiten anderer Information und sind für die Auswertung des ADFs von zentraler Bedeutung. Bei unvollständiger Information werden aufgrund der Dialektik stets die Akzeptanzbedingungen benötigt. Erforderlich sind sie, um selbst angepasst zu werden, damit die Akzeptanz eines Statements zutreffend angegeben werden kann, oder um Informationen via Algorithmen aus ihnen abzuleiten. Die Abhängigkeit der Statements zu ihren Eltern bzw. Vorgängern sorgt dafür, dass die ADFs rigide sind.

Ein vorgeschlagener Lösungsansatz ist es, die ADFs derart zu modifizieren, dass sie flexibler werden, ohne jedoch ihre Eigenschaften zu verändern. Dazu werden sogenannte Zielvorgabefunktionen (ζ-Funktionen) für die Statements eingeführt und die bisher (informations-) redundanten Links klassifiziert. Die Zielvorgabe besteht aus einer partiellen Form der Prädikatenlogik der 1. Stufe, in der nur Quantoren, Funktions- und 0-stellige Prädikatensymbole (Aussagenvariablen) vorkommen. Die Links werden dabei fortan in die Klassen unterstützend, angreifend oder sonstige eingeteilt. Anhand

dieser Modifikation kann über die ζ-Funktion für jedes Statement eine an die aktuellen Informationen angepasste Akzeptanzbedingung hergeleitet werden. Die ζADFs können somit schneller auf sich veränderte Situationen einer Argumentation reagieren.

Der Umgang mit unvollständiger Information wird durch ADFs geboten

Die ADFs stellen eine Struktur für die formale Argumentation zur Verfügung, die auch mit unvollständiger Information umgehen kann. Voraussetzung dafür ist, dass das Framework, wie in dieser Arbeit beschrieben, verwendet wird.

Die Komponente der Akzeptanzfunktion enthält ausschließlich aussagenlogische Formeln. Einer Aussagenvariable kann neben den Booleschen Werten auch der Wert *unsicher* zugewiesen werden. Dadurch eröffnet sich eine Möglichkeit, um unvollständige Information direkt formal über eine Belegung auszudrücken. Konkret kann ein Statement somit nicht nur den Wert *wahr* oder *falsch*, sondern auch *unsicher* annehmen. Ein entsprechender Operator wertet die dreiwertige Logik aus und gibt dadurch die Akzeptanz eines Statements an.

Dazu werden alle vorkommenden Aussagenvariablen mit allen möglichen Werten *wahr*, *falsch* und *unsicher* belegt. Dadurch ergeben sich exponentiell viele Interpretationen (präzise: $3^{|Statements|}$). Diese Menge an dreiwertigen Interpretationen kann nach ihrem Informationsgehalt und ihrem Wahrheitswert geordnet werden. Die Ordnung bei ersterem ist, dass die Werte *wahr* und *falsch* größer als der Wert *unsicher* sind und bei letzterem ist *wahr* größer als *unsicher* und *unsicher* ist größer als *falsch*. Diese partiell geordnete Menge ergibt ein mathematisches (Bi-) Gitter. Die einzelnen Elemente (Interpretationen) des Gitters besitzen benachbarte Elemente entsprechend ihrer Ordnung.

Jede Interpretation, d.h. jedes Element des Gitters, dient als Eingabe des Γ-Operators, der diese auswertet. Ist die Eingabe gleich der Ausgabe, so handelt es sich um einen Fixpunkt. Die Fixpunkte des Operators lassen sich den Semantiken wie *vollständig*, *zulässig*, *bevorzugt* und *grundiert* zuordnen. Ein solcher Fixpunkt bzw. die Auswertung einer Interpretation entspricht einer Extension des ADFs.

Diese Arbeit hat gezeigt, dass dies auch mit Interpretationen möglich ist, bei denen Statements *unsicher* sind. Das heißt, dass auch bei Statements, deren Vorhanden-

sein nicht zwingend gegeben ist, eine Auswertung erfolgen kann. Entweder erfolgt die Auswertung direkt inklusive der Unsicherheit oder die unsicheren Statements werden approximiert. Ermöglicht wird diese Approximation durch das Bi-Gitter, das die Interpretationen nach Informationsgehalt und Wahrheitswert ordnet. Eine Interpretation mit unsicherem Statement kann über seine Nachbarn approximiert werden.
Ein maschinelles System kann in diesem Zusammenhang mit unvollständiger Information umgehen. Eine solche Unvollständigkeit erzeugt keinen Abbruch oder eine triviale Lösung, sondern simuliert eine menschenähnliche Schlussfolgerung. Die entstandenen Extensionen (ausgewerteten Ergebnisse) können interpretiert werden, um eine fundierte Entscheidung zu treffen.

Die Eigenschaften der ADFs bleiben dabei erhalten. Insbesondere muss keine exponentielle Anzahl an Graphen verschiedener Abschlüsse gebildet und separat ausgewertet werden. Auch die bisherige Komplexität der Schlussfolgerungsprobleme bleibt bestehen.

6.2 Diskussion

Die ADFs lassen sich dem Forschungsbereich der *symbolischen KI* zurechnen. Durch spezifische, symbolbasierte Regeln wird versucht, menschliches Schlussfolgern zu simulieren. Die künstliche Intelligenz wird dabei durch formale und logische Regeln dargestellt. Die Inferenzprozesse basieren auf der ihr zur Verfügung stehenden Wissensbasis. Unvollständige Information ist für ein KI-System dieser Art besonders problematisch, da dessen regelbasierte Methodik klare und unmissverständliche Werte erfordert.

Die dargestellten Ergebnisse dieser Arbeit haben einen Ansatz präsentiert, um ADFs flexibler zu gestalten und gezeigt, dass ADFs mit unvollständiger Information umgehen können. Für letzteres bietet sich eine separate Fallstudie an, um die Stärke des Frameworks hervorzuheben. In diesem Abschnitt wird eine Diskussion des Ergebnisses zum Umgang mit unvollständiger Information angeregt. Für die Fallstudie wird eine geeignete Anwendungsdomäne benötigt.

Der Autor schlägt vor, das Rechtswesen, speziell des deutschen Steuerrechts, zu verwenden. Diese Anwendungsdomäne besitzt eine Reihe von Vorzügen, um die Einsatzfähigkeiten der ADFs zu zeigen.

Die KI-Forschung wird derzeit häufig im Zusammenhang mit E-Government und Digitalisierung genannt. In den dabei auftretenden Anwendungsgebieten sind die Fähigkeiten der ADFs zur menschenähnlichen Schlussfolgerung relevant. Die Digitalisierung der Verwaltung ist von besonderer Wichtigkeit, da Deutschland z.B. EU-Richtlinien umsetzen muss und die Verwaltung einen höheren Automatisierungsgrad benötigt, um ihre Aufgaben auch künftig erfüllen zu können.[35]

Als prominentes Beispiel kann die Einkommensteuererklärung angeführt werden. Die Steuerpflichtigen erklären keinen Sachverhalt in Form eines geschriebenen Textes, der erst unter die entsprechenden Paragrafen des EStG subsumiert wird, sondern sie nutzen amtlich vorgeschriebene Formulare. Dadurch erfolgt ein Zuschnitt des Sachverhalts in ein für die Verwaltung geeignetes Format zur Weiterverarbeitung. Die zugrundeliegenden Daten besitzen eine Darstellung, die auch für die weitere Verarbeitung mittels symbolische KI vorteilhaft ist.

Gesetze lassen sich als Regeln darstellen, wobei die Tatbestandsvoraussetzung der Prämisse und die Rechtsfolge der Konklusion entspricht. Der Verwaltung sind vom Gesetzgeber sogenannte Verwaltungsrichtlinien vorgegeben, damit der Ermessensraum deutschlandweit gleich ausgeübt wird. Einfach gesagt, sind dies Handlungsanweisungen, die vorgeben, wie ein bestimmter Sachverhalt zu handhaben ist. Sollten durch die Gesetzesformulierung weitere Unklarheiten bestehen, entscheiden die Gerichte über die Auslegung via Rechtsprechung. Kurzum existieren für dieses Anwendungsgebiet eindeutige, klare und genaue Regeln, die es zu befolgen gilt.

Der Einsatz von symbolischer KI ist für die Automation der Steuerverwaltung prädestiniert. Sollen Einkommensteuererklärungen vollautomatisiert veranlagt werden, wird ein logisches System benötigt, das die Erklärung ähnlich wie ein Sachbearbeiter auswertet. Erforderlich sind hohe Genauigkeit und Konsistenz sowie die Nachvollziehbarkeit des Lösungsprozesses. Das Wissen wird hierbei explizit durch logische Regeln ausgedrückt und die Schlussfolgerungen finden auf Basis dieser vordefinierten Regeln statt.

Das Beispiel Rechtswesen 5.2.2 auf Seite 65 hat einen ersten Eindruck der Einsatzfähigkeit der ADFs auf dem vorgeschlagenen Anwendungsgebiet vermittelt. Eine entsprechende Fallstudie dient auch zur Evaluation der Ergebnisse dieser Arbeit.

[35] vgl. BMI: Das Onlinezugangsgesetz und Berichte [19], [20].

Besonders vorteilhaft am Einsatz der ADFs ist, dass eine Auswertung auch möglich ist, wenn das Vorhandensein eines Statements nicht sicher ist. Auf die Rechtsmaterie bezogen ist etwas unsicher, wenn das Vorliegen einer Tatbestandsvoraussetzung unklar ist. Die ADFs gehen mit dieser unvollständigen Information wie folgt um:

- eine Rechtsfolge tritt unabhängig von der unklaren Voraussetzung nicht ein oder
- eine Rechtsfolge kann trotz Unsicherheit eintreten. Dies kann bereits zusammen mit der Unsicherheit geschehen oder erst durch ihre Beseitigung mittels Approximation erfolgen. In jedem Fall ermöglicht die Approximation bei einer Unklarheit eine schärfere Betrachtungsweise.

Die Aussagekraft dieser Auswertungen sollte in der Fallstudie untersucht werden. Dadurch kann sichergestellt werden, dass die Auswertungen zu einer vorteilhaften Lösungsfindung beitragen können.

6.3 Fazit

In der KI führt das Vorhandensein von unvollständiger Information stets zu Problemen. Besonders die symbolische KI hat aufgrund ihrer regelbasierten Methodik Schwierigkeiten, mit Unvollständigkeit und Unsicherheit zufriedenstellend umzugehen. So leiden beispielsweise die Robustheit, die Adaptionsfähigkeit und die Qualität der Entscheidung(en) des KI-Systems unter unvollständiger Information. In der KI-Forschung und -Entwicklung ist es deshalb sehr relevant, wie Modelle mit unvollständiger Information umgehen können.

Diese Arbeit hat das Problem auf dem Gebiet der formalen Argumentation, konkret an den ADFs, untersucht. Dabei wurde die Auswirkung von unvollständiger Information auf die ADFs unter dem Blickwinkel von Unvollständigkeit und Unsicherheit auf die Komponenten analysiert. Eine zentrale Feststellung dieser Analyse ist, dass Elemente der Komponenten eines ADFs in Wechselwirkung zueinander stehen. Dadurch können bereits vorhandene Ansätze im Umgang mit unvollständiger Information für die AFs bzw. AL nicht in analoger Weise auf die ADFs angewendet werden. In dieser Arbeit wurden zwei Ansätze präsentiert, damit ADFs hinsichtlich unvollständiger Information dennoch einsetzbar sind.

Im Hinblick auf die Unvollständigkeit ist eine flexiblere Anpassung der ADFs erforderlich. Im Verlauf einer ausgeprägten Argumentation können sich Positionen und Argumente verändern (Argumentationsdynamik), auf die das Framework reagieren muss. Mit dem Ansatz der *adaptiven ADFs* wird über die Argumentationsmuster ein (Argumentations-) Ziel festgelegt. Eine Funktion passt die Akzeptanzbedingungen entsprechend der sich verändernden Argumention an. Dadurch kann sich das Framework an Dynamiken anpassen und zufriedenstellend reagieren. Offen geblieben ist, ob die partielle Prädikatenlogik 1. Stufe auch direkt für die Akzeptanzbedingungen eingesetzt werden kann.

Für das fragliche Vorhandensein eines Statements (Unsicherheit) der ADFs wurde ebenfalls ein Ansatz vorgestellt. Dieser ermöglicht dem KI-System den Umgang mit unvollständiger Information. Ein unsicheres Statement kann entweder direkt über einen Operator oder mittels Approximation ausgewertet werden. Für die Approximation wird ein Bi-Gitter, das alle dreiwertigen Interpretationen nach Informations- und Wahrheitswert ordnet, verwendet. Eine Interpretation, die ein unsicheres Statement enthält, kann entsprechend mit ihren Nachbarn angenähert werden. Dadurch gelingt dem Framework ein geeigneter Umgang mit unvollständiger Information. Bisher fehlt es an der diskutierten Fallstudie, die die Umsetzung des Frameworks anhand einer Anwendungsdomäne evaluiert.

Beide Ergebnisse ermöglichen einem KI-System den robusteren Umgang mit unvollständiger Information. Der erstere Ansatz macht die ADFs flexibler gegenüber Veränderungen, verursacht z.B. durch die Argumentationsdynamik. Der letztere Ansatz sorgt dafür, dass auch bei Unsicherheiten nachvollziehbare Entscheidungen getroffen werden können. Eine mögliche Anwendungsdomäne stellt das Rechtsgebiet dar.

Literatur

[1] P. M. Dung, „On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games,“ *Artif. Intell.*, Jg. 77, Nr. 2, S. 321–358, 1995. DOI: 10.1016/0004-3702(94)00041-X. Adresse: https://doi.org/10.1016/0004-3702(94)00041-X.

[2] G. Brewka und S. Woltran, „Abstract Dialectical Frameworks,“ in *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*, F. Lin, U. Sattler und M. Truszczynski, Hrsg., AAAI Press, 2010. Adresse: http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1294.

[3] L. Amgoud, „Argumentation for Decision Making,“ in *Argumentation in Artificial Intelligence*, G. R. Simari und I. Rahwan, Hrsg. Springer, 2009, S. 301–320. DOI: 10.1007/978-0-387-98197-0_15. Adresse: https://doi.org/10.1007/978-0-387-98197-0%5C_15.

[4] T. J. M. Bench-Capon, H. Prakken und G. Sartor, „Argumentation in Legal Reasoning,“ in *Argumentation in Artificial Intelligence*, G. R. Simari und I. Rahwan, Hrsg. Springer, 2009, S. 363–382. DOI: 10.1007/978-0-387-98197-0_18. Adresse: https://doi.org/10.1007/978-0-387-98197-0%5C_18.

[5] K. Atkinson, P. Baroni, M. Giacomin u. a., „Towards Artificial Argumentation,“ *AI Mag.*, Jg. 38, Nr. 3, S. 25–36, 2017. DOI: 10.1609/AIMAG.V38I3.2704. Adresse: https://doi.org/10.1609/aimag.v38i3.2704.

[6] T. J. M. Bench-Capon und P. E. Dunne, „Argumentation in artificial intelligence,“ *Artif. Intell.*, Jg. 171, Nr. 10-15, S. 619–641, 2007. DOI: 10.1016/J.ARTINT.2007.05.001. Adresse: https://doi.org/10.1016/j.artint.2007.05.001.

[7] C. Beierle und G. Kern-Isberner, *Methoden wissensbasierter Systeme - Grundlagen, Algorithmen, Anwendungen, 5. Auflage* (Computational intelligence). SpringerVieweg, 2014, ISBN: 978-3-8348-1896-6. DOI: 10.1007/978-3-8348-2300-7. Adresse: https://doi.org/10.1007/978-3-8348-2300-7.

[8] P. Baroni, M. Caminada, M. Giacomin u. a., „Abstract argumentation frameworks and their semantics," in *Handbook of Formal Argumentation*, P. Baroni, D. Gabbay, M. Giacomin und L. van der Torre, Hrsg., College Publications, 2018, Kap. 4, S. 159–236, ISBN: 978-1-84890-275-6.

[9] U. Schöning, *Logik für Informatiker, 5. Auflage* (Spektrum-Hochschultaschenbuch). Spektrum Akadem. Verl., 2000, ISBN: 978-3-8274-1005-4.

[10] G. Brewka, H. Strass, S. Ellmauthaler, J. P. Wallner und S. Woltran, „Abstract Dialectical Frameworks Revisited," in *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, F. Rossi, Hrsg., IJCAI/AAAI, 2013, S. 803–809. Adresse: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6551.

[11] G. Brewka, S. Polberg und S. Woltran, „Generalizations of Dung Frameworks and Their Role in Formal Argumentation," *IEEE Intell. Syst.*, Jg. 29, Nr. 1, S. 30–38, 2014. DOI: 10.1109/MIS.2013.122. Adresse: https://doi.org/10.1109/MIS.2013.122.

[12] G. Brewka, S. Ellmauthaler, H. Strass, J. P. Wallner und S. Woltran, in *Handbook of Formal Argumentation*, P. Baroni, D. Gabbay, M. Giacomin und L. van der Torre, Hrsg., College Publications, 2018, Kap. 5, S. 237–285.

[13] M. Denecker, V. Marek und M. Truszczyński, „Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning," *Logic-based artificial intelligence*, S. 127–144, 2000.

[14] J. P. Delgrande, „A Knowledge Level Account of Forgetting," *J. Artif. Intell. Res.*, Jg. 60, S. 1165–1213, 2017. DOI: 10.1613/JAIR.5530. Adresse: https://doi.org/10.1613/jair.5530.

[15] D. Baumeister, M. Järvisalo, D. Neugebauer, A. Niskanen und J. Rothe, „Acceptance in incomplete argumentation frameworks," *Artif. Intell.*, Jg. 295, S. 103 470, 2021. DOI: 10.1016/J.ARTINT.2021.103470. Adresse: https://doi.org/10.1016/j.artint.2021.103470.

[16] G. Alfano, S. Greco, F. Parisi und I. Trubitsyna, „Incomplete Argumentation Frameworks: Properties and Complexity," in *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on*

Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022, AAAI Press, 2022, S. 5451–5460. DOI: 10.1609/AAAI.V36I5.20483. Adresse: https://doi.org/10.1609/aaai.v36i5.20483.

[17] J. Heyninck, G. Kern-Isberner, T. Rienstra, K. Skiba und M. Thimm, „Possibilistic Logic Underlies Abstract Dialectical Frameworks," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, L. D. Raedt, Hrsg., ijcai.org, 2022, S. 2655–2661. DOI: 10.24963/IJCAI.2022/368. Adresse: https://doi.org/10.24963/ijcai.2022/368.

[18] W. Dvorák und P. E. Dunne, „Computational problems in formal argumentation and their complexity," in *Handbook of Formal Argumentation*, P. Baroni, D. Gabbay, M. Giacomin und L. van der Torre, Hrsg., College Publications, 2018, Kap. 13, S. 631–687, ISBN: 978-1-84890-275-6.

[19] B. Rumpe, J. Michael, O. Kautz u. a., „Digitalisierung der Gesetzgebung zur Steigerung der digitalen Souveränität des Staates," *Berichte des NEGZ*, 2021. Adresse: https://idst.tax/wp-content/uploads/2021/06/NEGZ-Kurzstudie-19-Digitalisierung-der-Gesetzgebung-2021.pdf.

[20] C. Rammer, I. Bertschek, B. Schuck, V. Demary und H. Goecke, „Einsatz von künstlicher Intelligenz in der deutschen Wirtschaft: Stand der KI Nutzung im Jahr 2019," 2020. Adresse: https://www.bmwk.de/Redaktion/DE/Publikationen/Wirtschaft/einsatz-von-ki-deutsche-wirtschaft.pdf?__blob=publicationFile&v=1.

[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest und C. Stein, *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009, ISBN: 978-0-262-03384-8. Adresse: http://mitpress.mit.edu/books/introduction-algorithms.

Glossar

Begriff	Definition/Erklärung
Approximation Fixpoint Theory	Beschreibt nach [13] einen Ansatz, um Semantiken mittels Operator systematisch anhand von Fixpunkten zu identifizieren.
Agent	Agenten werden unterschiedliche Funktionen zuteil, durch die ein bestimmtes menschliches Verhalten nachgebildet werden soll. Diese Nachbildung erfolgte anhand der beschriebenen Semantiken aus der Arbeit [8, Kap. 3].
Algorithmus	Ein Algorithmus ist eine feste endliche Sequenz an Befehlen, die von einem Prozessor abgearbeitet wird.
Argumentations-dynamik	Im Verlauf einer Argumentation können Argumente hinzukommen oder überholt sein. Eine angemessene Argumentation erfordert es, dass die teilnehmenden Parteien mit der Veränderung von Argumenten umgehen können.
Aussagenlogik	Spezialfall der Prädikatenlogik 1. Stufe, der sich ergibt, wenn keine Quantoren und nur Prädikatensymbole mit Stelligkeit 0 betrachtet werden.
Extension	Als Extension wird eine Menge an Argumenten bzw. Statements eines Argumentationsframeworks bezeichnet, die sich genau den Regeln einer Semantik zuordnen lassen.
Inferenz	ist die aus einem formalen System automatisiert erstellte Schlussfolgerung.
Qualifikations-problem	Ein KI-System betrachtet nur eine Inszenierung der tatsächlichen Welt, d.h., dass es für eine Schlussfolgerung nicht möglich ist, alle Vorbedingungen unter allen erdenklichen Umständen zu garantieren und vollständig anzuführen (*qualification problem*).

A Anhang

A.1 Graphentheorie

Definition A.1.1 (Gerichteter Graph; aus [21, B.4 Graphs])**.**
Ein gerichteter Graph ist ein Paar (V, E), bestehend aus einer endlichen Menge V, den Knoten sowie einer Teilmenge von geordneten Paaren $E \subseteq V \times V$, den gerichteten Kanten.

Definition A.1.2 ([21, B.4 Graphs], [7, Def. B.4, B.6, B.7])**.**
Eine Folge von paarweise verschiedenen Knoten $v_1, \ldots, v_k \in V$ in einem gerichteten Graph heißt Pfad, wenn gilt: $(v_i, v_{i+1}) \in E,\ \forall i \in \{1, \ldots, k-1\}$.
Gilt zusätzlich: $v_1 = v_k$, so heißt die Folge Kreis (Zyklus). Ein Graph, der keine Zyklen enthält, heißt azyklisch. In einem azyklischen gerichteten Graphen bezeichnet:

1. $pa(v) := \{w \in V \mid (w, v) \in E\}$: die Menge aller direkten Vorgänger (*parents*).
2. $an(v) := \{w \in V \mid$ Es gibt einen Pfad von w nach v$\}$: die Menge aller Vorgänger (*ancestors*).
3. $ch(v) := \{w \in V \mid (v, w) \in E\}$: die Menge aller direkten Nachfolger (*children*).
4. $de(v) := \{w \in V \mid$ Es gibt einen Pfad von v nach w$\}$: die Menge aller Nachfolger (*descendants*).

Definition A.1.3 ([21, B.4 Graphs])**.**
Der Grad (*degree*) eines Knotens v bei einem gerichteten Graphen ist

$indeg(v) = |\{u \in V : (u, v) \in E\}|$ für alle einfallenden Kanten,

$outdeg(v) = |\{u \in V : (v, u) \in E\}|$ für alle ausgehenden Kanten und

$deg(v) = indeg(v) + outdeg(v)$ für alle Kanten.

A.2 Ergänzung zur Logik

Definition A.2.1 ([7, Def. 3.12])**.**
Eine Formel F ist:

1. *erfüllbar* (konsistent), falls F mindestens ein Modell besitzt.
2. *unerfüllbar* (inkonsistent, Kontradiktion), falls F kein Modell besitzt.
3. *allgemeingültig* (Tautologie), falls jede Interpretation ein Modell von F ist.
4. *falsifizierbar*, falls es wenigstens eine Interpretation gibt, die kein Modell von F ist.

Definition A.2.2 (Literal; aus [9, S. 38])**.**
Sei A eine atomare Formel ($A \subseteq \mathbb{A}$). Dann sind A und seine Negation $\neg A$ *Literale*. Die beiden Literale A und $\neg A$ sind komplementär.

Definition A.2.3 (Normalformen; aus [9, S. 26–27])**.**
Gegeben seien Literale L_{ij}, für $1 \leq i \leq m$, $1 \leq j \leq n_i$.

1. Eine Formel F ist in konjunktiver Normalform (KNF), falls sie eine Konjunktion von Disjunktionen ist: $F = \bigvee_{i=1}^{m}(\bigwedge_{j=1}^{n_i} L_{ij})$
2. Eine Formel F ist in disjunktiver Normalform (DNF), falls sie eine Disjunktion von Konjunktionen ist: $F = \bigwedge_{i=1}^{m}(\bigvee_{j=1}^{n_i} L_{ij})$

Definition A.2.4 (Klausel; aus [9, S. 38])**.**
Eine Klausel ist eine Disjunktion von Literalen. Eine endliche Klauselmenge entspricht einer Formel F in KNF.

Definition A.2.5 (Resolvente; aus [9, S. 38–39])**.**
Seien K_1 und K_2 Klauseln mit $L \in K_1$ und $\neg L \in K_2$. Dann ist $R = (K_1 \setminus L) \vee (K_2 \setminus \neg L)$ eine Resolvente von K_1 und K_2 nach L.

A.3 Tabellen

a	b	c	d	e	$a \wedge b \wedge (c \vee d) \rightarrow e$
0	0	0	0	0	1
0	0	0	0	1	1
0	0	0	1	0	1
0	0	0	1	1	1
0	0	1	0	0	1
0	0	1	0	1	1
0	0	1	1	0	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	0	1	1
0	1	0	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	0	1	1
1	0	0	1	0	1
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	0	1	1
1	0	1	1	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	0	1	1
1	1	0	1	0	0
1	**1**	**0**	**1**	**1**	**1**
1	1	1	0	0	0
1	**1**	**1**	**0**	**1**	**1**
1	1	1	1	0	0
1	**1**	**1**	**1**	**1**	**1**

Tabelle A.1: Die vollständige Wahrheitstafel mit fett gedruckten Zeilen für die Schlussregel *Modus Ponens* $\hookleftarrow$

$F \wedge F \equiv F$ $F \vee F \equiv F$	Idempotenz
$F \wedge G \equiv G \wedge F$ $F \vee G \equiv G \vee F$	Kommutativität
$(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$ $(F \vee G) \vee H \equiv F \vee (G \vee H)$	Assoziativität
$F \wedge (F \vee G) \equiv F$ $F \vee (F \wedge G) \equiv F$	Absorption
$(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H))$ $(F \vee (G \wedge H)) \equiv ((F \vee G) \wedge (F \vee H))$	Distributivität
$\neg\neg F \equiv F$	Doppelnegation
$\neg(F \wedge G) \equiv \neg F \vee \neg G$ $\neg(F \vee G) \equiv \neg F \wedge \neg G$	deMorgansche Regeln
$F \equiv (F \vee G) \wedge (F \vee \neg G)$	Resolution
$F \vee G \equiv F,$ falls F eine Tautologie $F \wedge G \equiv G,$ falls F eine Tautologie	Tautologieregeln
$F \vee G \equiv G,$ falls F unerfüllbar $F \wedge G \equiv F,$ falls F unerfüllbar	Unerfüllbarkeitsregeln
$F \to G \equiv \neg G \to \neg F$	Kontraposition
$F \to G \equiv \neg F \vee G$	Implikation
$F \leftrightarrow G \equiv (F \to G) \wedge (G \to F)$	Koimplikation

Tabelle A.2: Äquivalenzen übernommen aus [9, S. 24] und ergänzt. ↩

A.4 Algorithmen

```
foreach edge (x, y) ∈ L; do
  if [ x ∉ S ]; then
     S = S + x
  fi
  if [ y ∉ S ]; then
     S = S + y
  fi
done
```

Quelltext A.1: Algorithmus 1 ↩

Die Funktionsweise des Algorithmus A.1 wird anhand des *ADFs* aus Beispiel 3.1 beschrieben.
Zu Beginn wird aus der Menge der Links (L) ein Element entnommen – in diesem Fall von links nach rechts, sodass (b, a) zuerst entnommen wird. Anschließend prüft der Algorithmus, ob die Statements b und a noch nicht in der Menge der Statements (S) vorhanden sind. Bei positiver Prüfung werden die Statements der Menge S hinzugefügt: $S = \{b, a\}$. Der Algorithmus wiederholt diesen Vorgang bis es keine Elemente mehr in L gibt. Das letzte entnommene Element ist (e, d). Die Statements e und d werden nicht zur Menge S hinzugefügt, da sie bereits in S vorhanden sind.
Der Algorithmus terminiert, wenn kein Element mehr in L vorhanden ist – qua Definition 2.3.1 ist L endlich, dadurch ist auch der Algorithmus endlich. An den ursprünglichen gegeben Elementen der Komponenten nimmt der Algorithmus keinerlei Änderung vor, d.h. er verfälscht die Daten zu keinem Zeitpunkt. Die Laufzeit des Algorithmus ist abhängig von der Größe der Eingabe, in diesem Fall der Anzahl der Links. Es handelt sich um einen einfachen Algorithmus, der einmal durch die Menge der Links iteriert. Nachdem der Algorithmus terminiert, enthält die Menge S folgende Elemente: $\{b, a, c, d, e\}$.
Der Algorithmus kann nur ein zutreffendes Ergebnis liefern, wenn jedes Statement in einem zshg. Graphen in mindestens einer anderen Relation steht. Beispielsweise erzeugt der Algorithmus A.1 für:

$$DF = (\emptyset, \{(a, b), (b, a)\}, \{\mathcal{C}_a = a \wedge b, \mathcal{C}_b = \top, \mathcal{C}_c = \top\})$$

ein offensichtlich falsches Ergebnis: $S = \{a, b\}$.

```
1 foreach condition C_s ∈ C; do
2   foreach node x ∈ C_s; do
3     if [ x ∉ S ]; then
4        S = S + x
5     fi
6   done
7 done
```

Quelltext A.2: Algorithmus 2 ↩

Die Funktionsweise des Algorithmus A.2 wird anhand des *ADFs* aus Beispiel 3.1 erläutert.
Zuerst wird das Element $\mathcal{C}_a = b \wedge c \wedge (d \vee e)$ aus der Menge der Akzeptanzfunktion entnommen. Dieses Element wird in all seine Bestandteile aufgegliedert, das entspricht genau den einzelnen Statements b, c, d, e. Anschließend prüft der Algorithmus, ob ein Statement noch nicht in der Menge der Statements (S) vorhanden ist. Bei erfolgreicher Prüfung wird das Statement der Menge S hinzugefügt – das erste Statement b wird der Menge hinzugefügt: $S = \{b\}$. Der Algorithmus verfährt analog mit jedem weiteren Statement der Elemente (Akzeptanzbedingungen) aus der Komponente der Akzeptanzfunktion. Ist das letzte Elemente aus der Akzeptanzfunktion entnommenen und die Statements geprüft, terminiert der Algorithmus.
Die Menge der Akzeptanzfunktion ist endlich, wodurch auch der Algorithmus nach endlich vielen Schritten sicher terminiert. Das Ergebnis $S = \{b, c, d, e\}$ ist jedoch unzutreffend. Es ist offensichtlich, dass die Akzeptanzbedingung eines Statements nicht zwingend sich selbst beinhalten muss. Unter der Annahme, dass den jeweiligen Akzeptanzbedingungen die dazugehörigen Statements bekannt sind, ist es ausreichend, den obigen Algorithmus so zu modifizieren, dass die for-Schleife aus Zeile 2 entfällt. Der Algorithmus betrachtet nun lediglich die Indizes der Akzeptanzbedingungen, die das entsprechende Statement direkt angeben. Für die Grundfälle ist dies tatsächlich ausreichend, da hier *nur* eine Komponente an unvollständiger Information leidet. Mit dem angepassten Algorithmus ist das Ergebnis $S = \{a, b, c, d, e\}$ zutreffend. Die Laufzeit hängt in diesem Fall von der Größe der Eingabe der Akzeptanzfunktion ab.

A.5 Abbildungen

$DF = (\{a, b, c\}, \{(a, b), (b, a), (c, c)\}, \{\mathcal{C}_a = b, \mathcal{C}_b = a, \mathcal{C}_c = c\})$

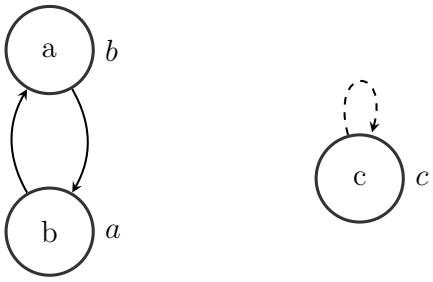
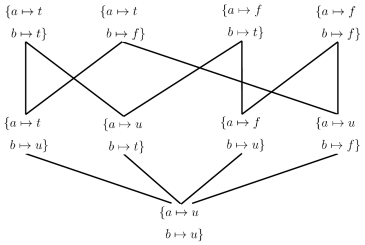
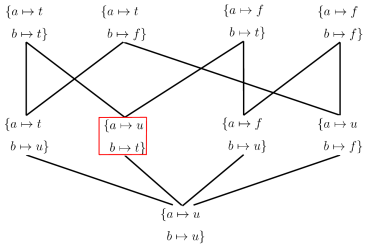


Abbildung A.1: Beispiel eines nicht zusammenhängenden Argumentationsgraphen mit Zyklus $(a, b), (b, a)$ und Schleife (c, c), wobei der Knoten c auch ohne Schleife vorkommen kann.

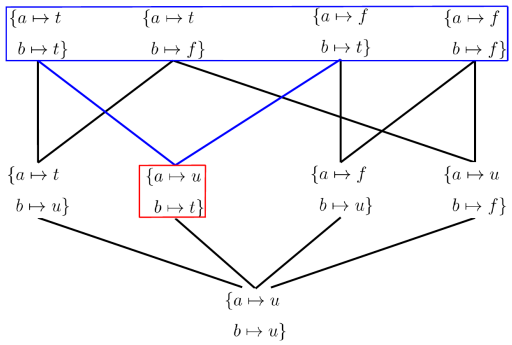
Das Bi-Gitter aus Abbildung 5.5 wird wiederholt durch die folgenden Abbildungen A.2a, A.2b, A.2c und A.2d dargestellt.



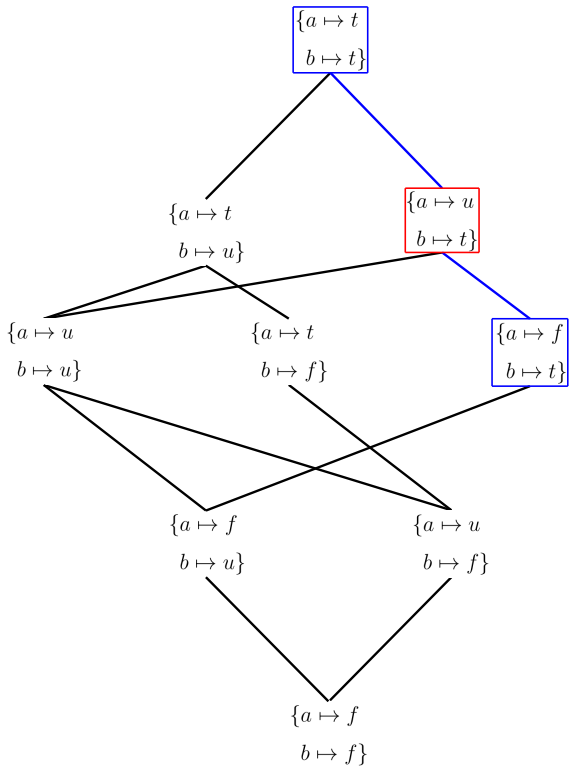
(a) Bi-Gitter mit zwei Statements nach $\leq_i$ und $\leq_t$ geordnet



(b) Eine Interpretation mit unsicherem Statement wird ausgewählt (rot)



(c) Auswahl der $\leq_i$-höheren Nachbarn (blaue Linien)



(d) Approximation nach 2.3 mit den $\leq_t$-Nachbarn (blaue Kästen)

Abbildung A.2: Für die Approximation werden zuerst die $\leq_i$-höheren Nachbarn ausgewählt und anschließend die $\leq_t$ Nachbarn, sodass die Gleichung 2.3 erfüllt ist. ↩

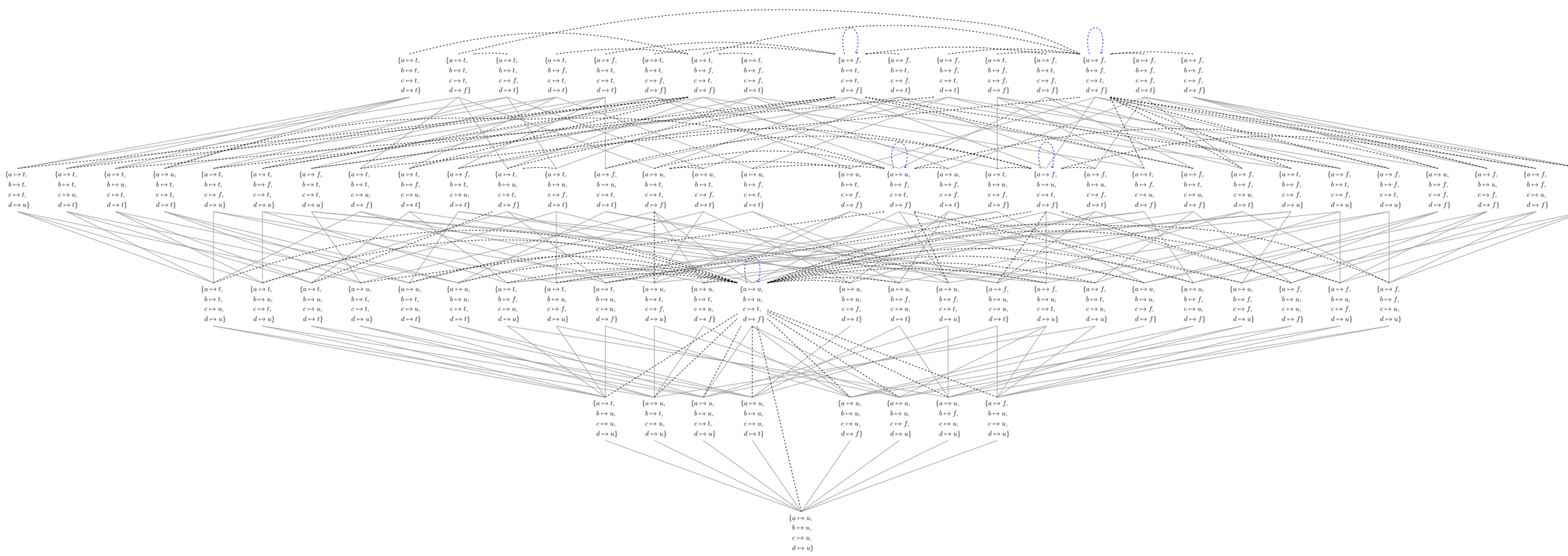


Abbildung A.3: Das Gitter $\mathcal{G}_3$. Die Übergänge zu den $\leq_i$-Nachbarn sind durch ganze Linien und die Transformationen des Γ-Operators als gestrichelte Pfeile dargestellt. Die Fixpunkte sind blau hervorgehoben. Die Abbildung ist skalierbar, wodurch ein Hineinzoomen ermöglicht wird. ↩