FernUniversität in Hagen
Fakultät für Mathematik und Informatik

AIG
Artificial Intelligence Group

Falsifikation eines Verifizierers Neuronaler Netze am Beispiel Marabous

Masterarbeit

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

vorgelegt von
Roland Krause

Erstgutachter:	Prof. Dr. Matthias Thimm, *Artificial Intelligence Group*
Zweitgutachter:	Prof. Dr. Jürgen Freudenberger, *Agentur für Innovation in der Cybersicherheit,* *Abteilung Schlüsseltechnologien*
Betreuer:	Prof. Dr. Matthias Thimm, Prof. Dr. Jürgen Freudenberger

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.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Zusammenfassung

Die Korrektheit von Marabou, einem Verifikationsprogramm von Eigenschaften Neuronaler Netze, wird untersucht. Neben einer kurzen Einführung in die Theorie Neuronaler Netze und ihrer Prädikatenlogischen Formalisierung, wird eine ausführliche Darstellung der Grundlagen dieser Formalisierung sowie der Spezifizierung der zu verifizierenden Eigenschaften gegeben, was auf die Introduktion von Marabou als Satisfiablity-Modulo-Theory-(SMT)-basierten Programm führt. Es wird der Reluplexalgorithmus vorgestellt, auf dem das Verfikationsverfahren basiert. Dass die Korrektheit dieses Algorithmus, durch die Verwendung von Fließkommazahlen anstelle der Reellen Zahlen gefährdet ist, findet sich exemplifiziert. Die Falsifikation Marabous wird auf der Grundlage eines neuartigen auf Hilbertmatrizen basierenden Verfahrens durchgeführt. Der von Jia und Rinard in [JR20] vorgestellte Exploit eines vollständigen Verifizierers Neuronaler Netze wird mit Bezug auf Marabou reproduziert.

Abstract

This thesis investigates the soundness of Marabou as a Satisfiability Modulo Theory based program for verifying properties of neural networks. An introduction to the concept of neural networks is given and the means are presented to transform the information incorporated in them into a formula of first order logic constrained to the theory of linear real arithmetic. The representation of properties to be verified is discussed. Expounding the modus operandi of the real valued Reluplex algorithm founds the suspicion of unsoundness resulting from Marabou's substituting floating point arithmetic for real arithmetic in the implementation of that very algorithm. By instilling Hilbert matrices into neural networks a novel approach for falsifying floating-point based verifiers is proved to be effective. The findings of [JR20] for exploiting verified neural networks via floating point numerical error are reproduced with respect to Marabou.

Inhaltsverzeichnis

1 Einleitung

> Es kommen die Zeiten des Betrugs [. . .] und der Edle wird in ihre Netze fallen.
>
> Johann Wolfgang von Goethe,
> *Götz von Berlichingen*

> The SAT problem is evidently a "killer app," because it is key to the solution of so many other problems. Consequently I can only hope that my lengthy treatment does not also kill off my faithful readers!
>
> Donald Ervin Knuth,
> *The Art of Computer Programming, Volume 4, Pre-Fascicle 6A, A Draft of Section 7.2.2.2: Satisfiability, 23. 09. 2015*

Die Zeiten des Betrugs sind bereits da. Zwar sind die Menschen – ob edel oder unedel – nicht vordringlich im Begriff in die Netze der Falschheit der „Nichtswürdigen", wie es in Goethes Götz gewendet wird, zu fallen, umso mehr indes in der Gefahr auf die, „nichtswürdig", falschen Begriffe von Netzen, Neuronalen Netzen, hereinzufallen. Die in Abbildung 1 dargestellten handschriftlichen Ziffernbilder zeigen offen-sichtlich – für menschliche, zumindest für von arabischen Ziffern geprägte, Begriffe – beide die Ziffer Null, in jedem Falle aber dieselbe Ziffer, da die Bilder sich in einer Weise, fast unmerklich, voneinander unterscheiden, die ihre – menschliche – Klassifikation als die Ziffer Null nicht beeinträchtigen kann.

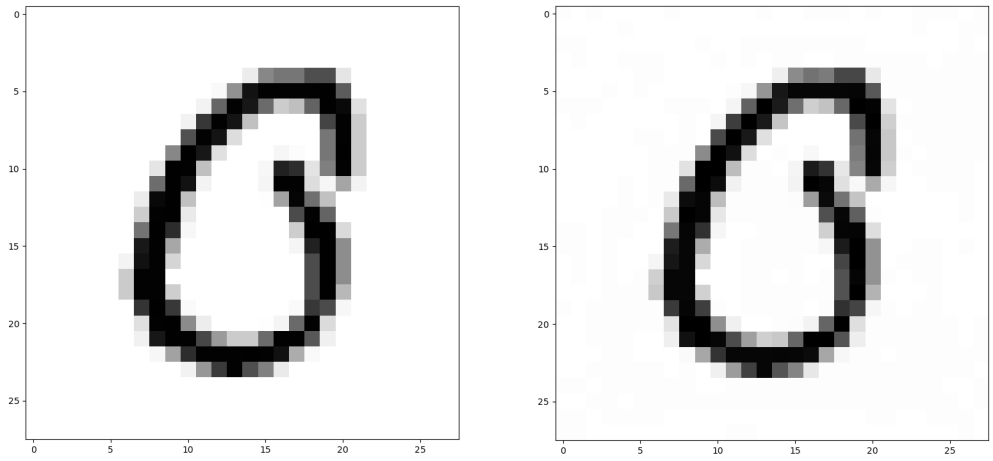


Abbildung 1: Verdeutlichung der Unterschiede menschlicher Begriffe und der Begriffe Neuronaler Netze. Dieses Beispiel ist bezogen auf eine konkrete Netz-Instanz N. Links: Dieses Bild wird von N als die Ziffer 0 klassifiziert. Rechts: Dieses Bild wird von N als die Ziffer 5 klassifiziert.

Anders jedoch stellt sich die Situation für ein, im Beispiel: konkretes, Neuronales Netz N dar. Das erste, links abgelistete, Bild wird von N klar als die Ziffer Null erkannt, im zweiten Bild hingegen erkennt das Netz N jedoch – „verblüffenderweise“ – die Ziffer Fünf. Diese verblüffende, englisch *intriguing*, Eigenschaft Neuronaler Netze ist zum ersten Mal, in systematischer Form, von einer Forschergruppe um Ian Goodfellow im Aufsatz *Intriguing Properties of Neural Networks*, [SZS+14], dokumentiert worden.

Die Möglichkeit des Nachweises der Abwesenheit dieser Eigenschaft – seine *Nicht-Robustheit* der Klassifikation bei winzigen pixelweisen Veränderungen eines Bildes – eines Neuronalen Netzes dürfte seitdem einer der Hauptmotivatoren für die Entwicklung von *Verifizierern* Neuronaler Netze gewesen sein. Zu diesen Verifizierern Neuronaler Netze – man sollte hier exakter von Verifizierern *von Eigenschaften* Neuronaler Netze, oder, ganz exakt, von Verifizierern von *Aussagen* bezüglich der Anwesenheit einer spezifischen Eigenschaft eines spezifischen Netzes, sprechen – zählt *Marabou*, welcher im Zentrum der vorliegenden Masterarbeit, im Kommenden auch nur „die Arbeit“, stehen wird. Die, längere, negative Antwort auf die Frage, ob dieser Verifizierer verifiziert werden kann, ist diese Arbeit.

1.1 Stil der Arbeit

Bevor an dieser Stelle die Struktur der Arbeit offenbart werden soll, möge die Adressierung einiger Peinlichkeiten des Autors erlaubt sein. Diese Arbeit ist übergriffig und eine Zumutung, wie im Folgenden dargelegt wird. Ab sofort nämlich, liebe Leserin, lieber Leser, sind wir „Per-Wir“, was insbesondere die Anmaßung impliziert, dass „wir“ künftig die Sachverhalte „behaupten“, „untersuchen“, „erschließen“ werden, ohne zu hinterfragen ob dies tatsächlich der Fall ist – weshalb sollte *ich*, als Leserin, als Leser, etwas behaupten, was nicht überzeugend ist, weshalb untersuchen, was unerquicklich zu sein verspricht und warum schließlich erschließen, was nicht schlüssig erscheint? Zur Verteidigung hat der Autor dieser Arbeit diesbezüglich wenig vorzubringen außer, vielleicht, diese Hinsicht: Die Nützlichkeit der vorgetragenen Gedanken ist zuweilen hinreichend groß um über diese Anmaßung hinwegzutrösten. Wer ertrüge den Ton des „Oberlehrers“ Seneca anders, als dadurch, dass durch diesen großartige Erkenntnisse transportiert werden, die auf anderem Wege nicht leicht zu erlangen sind. Seine „Mikroökonomik“ – hier insbesondere als der Lehre vom Erwerb (sowie Verlust) und der Verteilung von Gütern und Dienstleistungen im privaten –, im Werk *De Beneficiis* dargelegt, ist bis heute nicht, insbesondere durch die hiesigen wirtschaftswissenschaftlichen Fakultäten nicht, erreicht. Sokrates erfuhr für seine „Besserwisserei“, genauer: seinen Nachweis des „Nicht-Besserwissens“ seiner Gesprächspartner, sogar das Urteil des Todes durch seine Zeitgenossen – doch niemand, der heute um den Wert des meinungsbildenden Diskurses innerhalb einer als Republik verfassten Demokratie weiß, bestreitet den Wert der „maieutischen“ Sokratischen Methode: Der kommunikativen Interaktion durch – „Wissen gebährende“ – Argumentation.

Der Autor der vorliegenden Arbeit erlaubt sich des Weiteren, das generische Femininum neben das generische Maskulinum zu stellen, was im gegenwärtig tobenden Kulturkampf – wonach die Einführung neuer Zeichen in die Sprache Goethes, deren Integrität durch die Einführung des „ß“ als eigenem Buchstaben demnach wohl bereits damals angezählt gewesen sein dürfte, anscheinend als derartig invasiv begriffen wird, dass man, der deutschen Sprache selbst nicht mächtig („Ik gihorta ðat seggen, ðat sih urhettun ænon muotin, Hiltibrant enti Haðubrant untar heriun tuem.“, Beginn des Hildebrandslieds, einem, zugegeben, älteren deutschen Text mit einigen neuen alten Zeichen), Verbote erlassen zu müssen glaubt – hoffentlich nicht bereits eine zu große Angriffsfläche bietet.

Eine zweite Gegebenheit der vorliegenden Arbeit, für die sich dessen Autor leider zu einer Rechtfertigung genötigt sieht, betrifft das Medium, durch welches die wissenschaftlichen Gedanken dieser Arbeit transferiert werden sollen: die Sprache, oder, besser, die Art der Sprache. Es scheint sich, seitdem die großen Tagen der deutschsprachigen Mathematik – zu deren Status etwa Felix Klein, David Hilbert oder Kurt Gödel, um nur einen kleine, willkürliche, Auswahl zu geben, nicht unwesentlich beigetragen haben dürften, bis in die Dreißiger Jahren des letzten Jahrhunderts hinein – (die Informatik war zu diesem Zeitpunkt noch keine von der Mathematik verschiedene Einzelwissenschaft) der Vergangenheit zuzuzählen sind, das Paradigma durchgesetzt zu haben, Texte der Informatik müssten „in Assembler“ und nicht „in Hochsprache“ geschrieben werden. Im Einleitungstext zu Uwe Schönings – wohlgemerkt einem der, im Hinblick auf seine Verdienste um das sogenannte „*Satisfiability Problem*“, unbestritten renomiertesten, und eben deshalb, *Pars pro Toto*, als Beispiel zur Veranschaulichung ausgewählten, zeitgenössischen deutschsprachigen Informatikers, aber, als Hochschullehrer, eben auch Didakt – als Einführung gedachtes, Lehrbuch „*Logic for Computer Scientists*“, für Studierende der Informatik, nicht etwa der Mathematik mit Spezialisierung in Formaler Logik, liest man als allerersten Satz:

> Formal Logic investigates how assertions are combined and connected, how theorems formally can be deduced from certain axioms, and what kind of object a proof is. [Sch01]

An dieser, im Zitat bereits vierten Auflage der, „Einleitung“ ist fast nichts falsch – ein Beweis, englisch *proof*, ist, *pace* Professor Schöning, allerdings kein *Gegenstand*, englisch *object*, sondern *ein Satz*, eine Tautologie, der Art „Aus Axiom A folgt logisch B, woraus logisch folgt, dass ... und aus C folgt logisch D, das zu Zeigende.“ –, aber fast alles unverständlich: Was sind denn Behauptungen, englisch „assertions“? Etwa auch „Gegenstände“? Was ist ein Axiom, was ein Beweis? Aus der Perspektive der Didaktik – von griechisch διδάσκειν [didáskein], „lehren“, „unterrichten“, eigentlich aber: „mitteilen“ –, der Wissenschaft der erfolgreichen „Informationsübertragung“, welche bereits als solche einem Informatiker angelegentlich sein sollte, dürften nur robuste Persönlichkeiten einen unverminderten Drang verspüren, ab dieser Stelle weiterzulesen. Für eine Person, die nicht bereits „zum Kreis derjenigen Eingeweih-

ten“ gehört, welche mit den Begriffen der Formalen Logik vertraut sind – weshalb auch sonst sollte sie dieses Einführungsbuch zu dieser Thematik zu lesen in Erwägung ziehen –, liest sich obiger Satz nämlich wie

> x_1 investigates how x_2s are combined and connected, how x_3 can be Red from certain x_4s, and what kind of object an x_5 is.

wobei x_1, x_2, x_3, x_4, x_5, sprichwörtlich, Unbekannte sind und R eine nicht näher spezifizierte Relation namens „*formal deduction*“ zu sein scheint.

Die vorliegende Masterarbeit möchte im bewussten Gegensatz dazu dieses „Assemblersprachen“-Paradigma durchbrechen und „in Hochsprache“ *erklären*, was es mit den zur Untersuchung stehenden Gegenständen – und Sätzen – auf sich hat. Im angelsächsischen Raum ist dieser Paradigmenwechsel bereits in vollem Gange, wovon etwa Turing-Award-Gewinner Donald Knuth durch die unterhaltsame, humorvolle Art der Aufbereitung wissenschaftlicher Erkenntnisse vor dem Transport an die Leserschaft, reiches Anschauungsmaterial liefert. Dass dabei in einer vergleichbaren Weise pfleglich und respektvoll mit den Untersuchungsgegenständen umgegangen wird, wie etwa die Primatenforscherin Jane Goodall mit den ihren, ist Teil dieses neuen Paradigmas, dem sich die vorliegende Arbeit verpflichtet fühlt. Nichts erklären und mit Bezug auf die Geschichte der Untersuchungsgegenstände – „If I have seen further it is by standing on the shoulders of Giants“. Isaac Newton – diesen keinen Respekt, lateinisch „Rück-Sicht“, zu zollen, benötigt dabei natürlich, rein quantitativ, weniger Worte, und meist auch weniger Wörter, als diese Spezifika einer didaktischen „Hochsprache“ einzuhalten, was zum Umfang der vorliegenden Arbeit beigetragen haben dürfte. Den eigentlichen Umfang der Arbeit – welcher der Leserin, dem Leser im Folgenden zugemutet, besser: zugetraut, wird – jedoch macht deren Inhalt aus, auf den wir nun zu sprechen kommen möchten.

1.2 Aufbau der Arbeit

Zu den wichtigsten Gegenständen, mit denen in der vorliegenden Arbeit hantiert wird, gehören die sogenannten *Neuronalen Netze*, denen sich das Kapitel 2 widmet. Die Geschichte dieser Netze, ihr Aufbau und ihre prinzipielle Funkitionsweise finden dort ihre kurze Beschreibung. Insbesondere werden auch die bilderkennenden Netze, Klassifizierer genannt, welchen wir bereits im allerersten Abschnitt dieser Einleitung begegnet sind, vorgestellt.

Dass diese innere Struktur, ihre „Semantik“, Neuronaler Netze durch eine Formalisierung in der Prädikatenlogik Erster Stufe ausgedrückt werden kann, ist der Untersuchungsgegenstand von Kapitel 3, welches desweiteren die Möglichkeit einer kompakteren Darstellungsform Neuronaler Netze aufzeigt.

Kapitel 4 leistet eine konzise Einführung in die Prädikatenlogik in dem Maße, dass sie die Grundlagen der Verifikation *Marabou*s, als Verifizierer Neuronaler Netze, verständlich machen kann. Wer mit der Prädikatenlogik wohl vertraut ist, sei, um die Lektüre zu verkürzen, ermutigt, dieses Kapitel zu überspringen.

Eine ähnliche Aufforderung ergeht bezüglich des sich anschließenden Kapitels 5. Wer sich mit den Begriffen Erfüllbarkeit, englisch *satisfiability* und dem Konzept der Theorien sowie der „Erfüllbarkeit“ unter, lateinisch „*modulo*“, einer Theorie vertraut fühlt, überspringe dieses Kapitel, welches sich zunächst der Erfüllbarkeit von Sätzen widmet, um daraus den Begriff der Logischen Folgerung ableiten zu können, was auf den Theoriebegriff führt. Der Begriff der Theorie, insonderheit, der Theorie der Linearen Arithmetik Reeller Zahlen, ist an dieser Stelle von Bedeutung, da diese Theorie – als eine Menge von Sätzen – zugleich die Sprache, genauer: die Menge aller wahren Sätze dieser Sprache, *Marabou*s ist, die in diesem Kapitel ebenfalls vorgestellt wird.

Wer das Kapitel 4 und das Kapitel 5 übersprungen hat, sollte indes mit Kapitel 6 wieder einsteigen, welches der Klarifikation des Begriffes der Eigenschaften gewidmet ist, deren Verifikation mit Bezug auf ein gegebenes Neuronales Netz, durch *Marabou* vollzogen werden soll. Da bisher wenig Forschungsarbeit in Richtung einer diesbezüglichen Begriffsklärung der Eigenschaften von Neuronalen Netzen im Speziellen geleistet worden zu sein scheint, beginnt diese Untersuchung mit der Analyse einer der wenigen dennoch zu findenden Definitionen und leistet auf dieser Analyse aufbauend schließlich eine eigene Definition von Eigenschaften Neuronaler Netze. Zum Abschluss des Kapitels wird gemäß dieser Definition die Eigenschaft der, uns bereits im ersten Abschnitt dieser Einleitung begegneten, Robustheit formalisiert.

Das Kapitel 7 stellt den Algorithmus, *Reluplex*, vor, auf dessen Grundlage *Marabou* die Verifikation der im vorherigen Kapitel eingeführten Eigenschaften Neuronaler Netze durchführt. Wer mit der Funktionsweise des Simplex-Algorithmus vertraut ist, mag geneigt sein, direkt bei Kapitel 7.6 einzusteigen, könnte es indes nützlich finden, in vorherige Abschnitte dieses Kapitels bei Bedarf zurückzublättern.

Das Projekt der Falsifikation *Marabou*s, dem sich diese Arbeit verschrieben hat, verwendet als Hebel dieser Falsifikation die Annahme der Entwickler des *Marabou*-Programms, die Reellen Zahlen durch Fließkommazahlen und die Arithmetik der Reellen Zahlen durch die Fließkommazahlenarithmetik, welche im Verifikationsprozess Verwendung finden, ersetzen zu können. Den Problemen, die aus dieser Annahme zu entstehen vermögen, widmet sich das Kapitel 8.

Den Abschluss der Arbeit stellt das Kapitel 9 dar, welches der eigentlichen Falsifikation *Marabou*s dient. Nachdem der, durchaus erklärungsbedürftige, Begriff der Falsifikation eines Verifizierers, der sich nicht zuletzt auch im Titel der Arbeit wiederfindet, erläutert wurde, wird ein für diese Arbeit eigens entwickelter neuartiger Ansatz zur Falsifikation fließkommabasierter Verifizierer Neuronaler Netze erfolgreich auf *Marabou* appliziert, womit die Falsifikation *Marabou*s als abgeschlossen gilt. Der letzte Abschnitt reproduziert – lies: wiederholt diesen und gelangt zum selben Ergebnis – die Ergebnisse eines aus der Literatur entnommenen „Angriffs“ auf fließkommabasierte Verifizierer Neuronaler Netze, erfolgreich für den Verifizierer *Marabou*, was dadurch seiner erneuten Falsifikation gleichkommt.

1.3 Reverenzen

Wenngleich damit die Peinlichkeit einhergeht, neben dem Betreuer zugleich auch dem Gutachter Dank abzustatten, so nötigen die vielleicht nicht gänzlich gewöhnlichen Umstände den Verfasser der vorliegenden Arbeit zu eben jenem Schritt. Die Ermunterung von Prof. Dr. Thimm, die vorliegende Arbeit unter das Dach seiner *Artificial Intelligence Group* der FernUniversität Hagen zu stellen, zeugt von einer Aufgeschlossenheit, die – da das umfasste Lehrgebiet sich nicht zuletzt auch der Erforschung der sogenannten Formalen Argumentation widmet, und die *Artificial Intelligence Group* somit, durch die Analyse des Widerstreits der Argumente, der Sokratischen Methode, gerade für die Offenheit gegenüber geisteswissenschaftlichen Ansätzen, zu welchen auch die Falsifikation eines Verifizierers, *qua* Logischem Unterfangen, zu zählen ist, steht – zwar wenig überrascht aber, eben deshalb, großen Respekt und der dankenden Erwähnung verdient. Dem ist hinzuzufügen, dass aus dem Oberseminar von Prof. Dr. Thimm, in welchem dem Autor Gelegenheit zur Vorstellung einiger Thesen der Arbeit gegeben wurde, wertvolle Anregungen für die Arbeit entstanden sind. Zu danken ist hier insbesondere Kai Sauerwald, der, die Tragweite des Projekts sofort begreifend, interessantere zu verifizierende Eigenschaften als die Robustheitseigenschaft vom Autoren einforderte, wobei Letzterer sich jedoch nicht sicher ist, diese durch die vorliegende Arbeit liefern zu können. Das ihm gegebene Versprechen – eine drastische, aber treffende, Beschreibung dessen, was der Autor mit Bezug auf *Marabou* in der Arbeit vorhabe, nicht zu zitieren – ist hiermit eingelöst.

Diese „nicht gänzlich gewöhnlichen Umstände“ stammen aus der Kooperation mit der *Agentur für Innovation in der Cybersicherheit*, kurz *Cyberagentur*, im Rahmen derer diese Arbeit entstanden ist. Die Peinlichkeit des vorherigen Absatzes wiederholend, danke ich meinem von der Cyberagentur gestellten Betreuer Prof. Dr. Freudenberger, nicht zuletzt für seine unerschütterliche Zuversicht in die Fähigkeiten des Autors der Arbeit, das anspruchsvolle Projekt der Falsifikation *Marabou*s, zu Ende führen zu können. Mein besonderer Dank gilt schließlich Syrko Kulas von der *Cyberagentur*: Seine „proferierten“ kritischen Anmerkungen induzierten Überarbeitungen, welche der Verständlichkeit der Arbeit maßgeblich zugute gekommen sein dürften. Die Verantwortung der Unverständlichkeit aller übrigen Passagen zu tragen, verbleibt indes die Obliegenheit des Autors dieser Arbeit.

1.4 Offenlegung der Abhängigkeit der Arbeit

Der Autor der vorliegenden Arbeit sieht sich aus wissenschaftsethischen Gründen zu folgender Aussage verpflichtet. Hiermit ist offengelegt, dass die Entscheidungsfindung für das Thema dieser Arbeit aus der Kooperation mit der *Agentur für Innovation in der Cybersicherheit* hervorgegangen ist, und dass ein Beitrag zur Finanzierung dieser Arbeit von der *Agentur für Innovation in der Cybersicherheit* geleistet worden sein wird. Die Einschätzung der wissenschaftlichen Unabhängigkeit der vorliegenden Masterarbeit gilt unter diesen Bedingungen.

2 Neuronale Netze

> Since the advent of electronic computers [...], an increasing amount of attention has been focused on the feasibility of constructing a device possessing such human-like functions as perception, recognition, concept formation, and the ability to generalize from experience.
>
> Frank Rosenblatt,
> *The Perceptron – A Perceiving and Recognizing Automaton* [Ros57]

Um zu verstehen, wie die Verifikation von Eigenschaften Neuronaler Netze vonstatten gehen kann, müssen wir zunächst die in der Verifikation beteiligten Gegenstände formal einführen. Wir beginnen mit der Darstellung Neuronaler Netze, so wie sie in der Verifikation Verwendung finden werden.

Das Eingangszitat von Frank Rosenblatt, mag als Strukturgeber für dieses Kapitel gelten. Neuronale Netze sind ein Zusammenspiel von Apparaturen (englisch *devices*; Rosenblatt konzipierte das Perzeptron-System anders als zeitgenössische Neuronale Netze als Hardware, nicht als Software, [Ros57]), welche die menschliche Wahrnehmung emulieren und aus dem Wahrgenommenen (englisch *perceiving*; daher der Name *Perceptron*) Begriffe entwickeln, die über das Perzipierte hinaus generalisieren. Letzteres kann durchaus als intelligentes Verhalten interpretiert werden. Das Perzeptron, welches vor nunmehr über 65 Jahren konzipiert wurde, hat, in adaptierter Form, als Grundbaustein Neuronaler Netze bis heute seine Gültigkeit. Man vergleiche zum Folgenden auch das erste Kapitel aus Nielsons Buch über Neuronale Netze und *Deep Learning*, [Nie15].

2.1 Perzeptronen und Neuronen

Wir können uns – hier und im Folgenden auf der Grundlage von Rosenblatt [Ros57] – ein *Perzeptron* als ein System vorstellen, welches drei Hauptkomponenten umfasst. Betrachten wir dazu die Abbildung 2. Die erste Komponente bildet den Zugang zur Außenwelt über eine Nachbildung der Sensorik. Entsprechend wird sie von Rosenblatt als *S-System*, für englisch *Sensory System*, bezeichnet. Das *S-System* ist ein System von Reizeindrücken oder Stimuli, die über entsprechende Verbindungen in das sich anschließende *A-System* Eingang finden. Jeder der Stimuli ist durch eine Reelle Zahl repräsentiert. Dabei gilt: Eine positive Zahl repräsentiert einen Reiz oder Exzitation. Eine negative Zahl repräsentiert eine Reizminderung oder Inhibition, bezogen auf die gleichzeit ebenfalls mit dem Reizminderungssignal eintreffenden Reize. Eine Verbindung zwischen *S-* und *A-System* entscheidet, vermittels des ihr jeweils individuell beigeordneten reellzahligen Gewichtes, wie ein Reiz(-unterdrücker) aufbereitet wird, bevor er in das *A-System* eingespeist wird. Dies geschieht durch einfache Multiplikation des Stimulus-Wertes mit dem Gewicht der Verbindung. Demnach lässt ein Gewicht von 1 den sensorischen Impuls unverän-

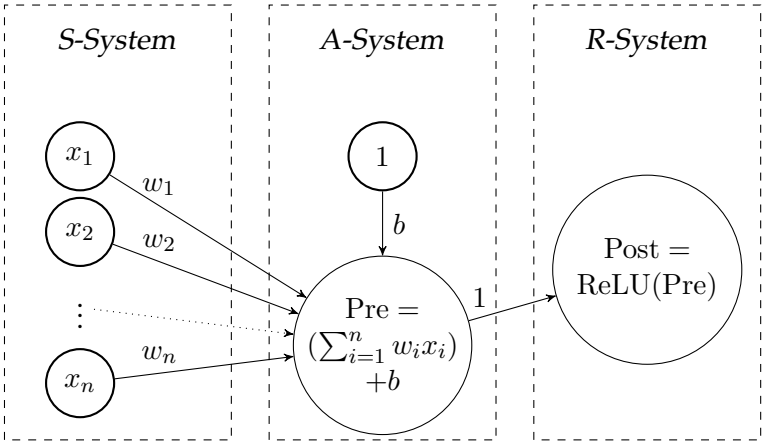


Abbildung 2: Schema eines Perzeptrons. Links: Schema des *S-Systems*, mit Übergängen zum *A-System*. Das *S-System* besteht aus den reellwertigen x_i, mit $i \in \{1, \ldots, n\}$ sowie $n > 0$. Der Übergang vom *S-* in das *A-System* ist für jedes der x_i mit dem zugehörigen reellwertigen Gewicht w_i markiert. Mitte: Schema des *A-Systems* mit Übergang zum *R-System*. Das *A-System* vermittelt zwischen dem *S-* und dem *R*-System, indem es dem *R*-System die jeweils mit w_i gewichtete Summe der x_i des *S*-Systems, additiv ergänzt um den perzeptroneigenen Bias, b, übergibt; dabei handelt es sich um den Wert unmittelbar vor der Anwendung der Aktivierungsfunktion und wird deshalb als „Pre“ (englisch *pre-activation value*) bezeichnet. Rechts: Schema des *R-Systems*. Seine Aufgabe besteht in der Anwendung der Aktivierungsfunktion auf den Pre-Wert. Als Aktivierungsfunktion wurde hier die *ReLU*-Funktion gewählt. Das Resultat ist dann der Post-Aktivierungswert, kurz „Post“.

dert, ein Wert größer als 1 verstärkt ihn, ein Wert zwischen 0 und 1 schwächt ihn ab und ein Wert kleiner als 0 „invertiert“ ihn zusätzlich zum Effekt des Absolutbetrags des Wertes: Aus einer Inhibition wird eine Exzitation und *vice versa*.

Das bereits erwähnte *A-System* (*Association Sytem*) wird von Rosenblatt eingeführt, um zwischen Eingabe, also dem vorverabeiteten Reiz, und der Ausgabe, also der Reaktion auf den Reiz, zu vermitteln. Der folgende Teil der Darstellung unterscheidet sich in den Details deutlich von Rosenblatt [Ros57] und gibt die heute gängige Ausdeutung künstlicher Neuronen wieder, wie etwa in Goodfellow et. al., [GBC16], dargestellt; die Kernidee Rosenblatts bleibt davon aber unberührt. Dabei verfährt das *A-Sytem* wie folgt. Es kumuliert die, wie besprochen über die Gewichts-Multiplikation der jeweiligen Verbindungen zwischen *S-* und *A-System*, vorverarbeiteten eintreffenden Reize und Reizminderungen vermittels schlichter Addition ihrer Werte. Außerdem gibt es im *A-System* einen, für jedes Perzeptron eigenen, Perzeptron-Schwellwert. Ist der kumulierte Eingangswert kleiner als der

Schwellwert, ist also die Differenz zwischen der Kumulation und dem Schwellwert negativ, so wurde die Schwelle nicht überschritten. Andernfalls ist die Differenz positiv und der Schwellwert wurde überschritten. Die Differenz wird in beiden Fällen an die dritte Komponente, das *R*-System, übergeben. Der mit -1 multiplizierte Schwellwert s, mit dem wir im Folgenden anstelle des Schwellwertes s stets operieren werden, wird *Bias* genannt und meist als b notiert: $b := -s$. Mangels geeignet erscheinender Übersetzungen, verwenden wir im Folgenden stets nur die englische (eigentlich französische: *biais*, „das Schräge", „Schiefe") Bezeichnung *Bias*, und verweisen an dieser Stelle auf die, möglicherweise bisweilen irritierende, Tatsache dass im Deutschen der Plural von Bias ebenfalls Bias lautet. Die Aufgabe des *A-Systems* lässt sich, zusammenfassend, also beschreiben als die Addition der Summe der vorverarbeiteten Reize und Reizminderungen mit dem Bias.

Die dritte und letzte Komponente des Rosenblatt-Perzeptrons bildet schließlich das *R-System*. Es steht für das sogenannte *Response System*, als dasjenige System, das basierend auf der Information des *A-Systems*, über die letztendliche Reaktion auf den Eingangsreiz entscheidet. Auch hier nehmen wir wieder eine Adaption in Richtung heute gängiger Darstellung, wie etwa auch Goodfellow et. al., [GBC16], an Rosenblatts Version vor. Das *A-System* übermittelt dem *R-System*, so hatten wir gesehen, entweder das negativwertige Signal wie viel positive Reiz-Eingabe noch fehlt, um den Schwellwert zu erreichen oder das positivwertige Signal, um wie viel der Schwellwert überschritten wurde. Auf Grundlage dieser Information entscheidet das *R-System* des Perzeptrons über die endgültig Antwort des Perzeptrons gegeben die Eingangs-Stimuli. Diese Entscheidung lässt sich als eine Funktion über die Information des *A-System* darstellen, wobei der Wert der Funktion, zur jeweiligen Eingabe, die endgültige Reaktion des Perzeptrons liefert. In Rosenblatts ursprünglicher Version des Perzeptrons, [Ros57], wurde diese Funktion als Alles-Oder-Nichts-Prinzip konzipiert. Wenn der Schwellwert des Perzeptrons überschritten wird, dann ist der Wert der Funktion „Alles": Ein Hebel („*typebar*", [Ros57]) wird in Gang gesetzt, eine Kontroll-Leuchte wird eingeschaltet oder Ähnliches. Wird der Schwellwert nicht überschritten, passiert nichts. In heutiger Terminologie wird diese Funktion als „Aktivierungsfunktion" bezeichnet und das Alles-Oder-Nichts-Prinzip aufgegeben; für eine grobe Übersicht über die Fülle heute gängier Aktivierungsfunktionen verweisen wir auf Goodfellow et. al., [GBC16].

Für die vorliegende Arbeit interessiert vordergründig eine Art der Aktivierungsfunktion, oder auch kurz Aktivierung, die sogenannte *Rectified Linear Unit*, kurz *ReLU*. Wir beschränken uns im Wesentlichen auf diese Aktivierungsfunktion, weil sie eine der wenigen gängigen nichtlinearen Aktivierungsfunktionen ist, die sich exakt als aus stückweise Linearen Funktionen zusammengesetzte Funktion beschreiben lässt. Da der *Marabou*-Verifizierer nur auf der Basis von stückweise linearisierbaren Funktionen operieren kann und die *ReLU*-Funktion darüberhinaus eine Aktivierungsfunktion ist, deren Verwendung in gegenwärtigen Neuronalen Netzen, seit ihrer Einführung als praktikable Alternative zu transzendenten Aktivierungsfunktionen, [NH10], nicht unüblich ist, scheint diese Beschränkung nicht unbegründet.

Im Folgenden sei die *ReLU*-Funktion als Aktivierung im Kontext des Perzeptrons kurz vorgestellt. Die *ReLU*-Aktivierung verfährt mit der Eingabe-Information des *A-Systems* – wie stark wurde der Schwellwert über- respektive unterschritten – wie folgt. Solange der Schwellwert nicht überschritten wurde, gibt die *ReLU*-Funktion invariant 0 zurück. Wurde der Schwellwert hingegen überschritten, ist der Wert der Funktion der Betrag, um den der Schwellwert überschritten wurde, welches in diesem Fall genau den Wert der Eingabe bedeutet. Formal lässt sich die *ReLU*-Funktion direkt mathematisch so angeben.

$$\mathrm{ReLU}(x) := \max\{0, x\} \tag{1}$$

Wir können die *ReLU*-Funktion, wenngleich etwas weniger elegant, äquivalent formulieren als

$$\mathrm{ReLU}(x) = y, \quad \text{mit } y := \begin{cases} 0, & \text{falls } x \leq 0 \\ x, & \text{falls } x > 0 \end{cases}, \tag{2}$$

um die logische Struktur der Funktion klarer hervorzuheben.

Perzeptronen werden in heutiger Terminologie als *künstliche Neuronen* oder auch nur als Neuronen bezeichnet, so etwa in Goodfellow et. al., [GBC16], oder in Albarghouthi [Alb21]. Wir wollen im Folgenden eine formale Darstellung für Neuronen angeben, bevor wir sie innerhalb des Systems Neuronaler Netze verorten. Der mathematische Kern eines Neurons lässt sich demnach auf eine Funktionskomposition zweier Funktionen reduzieren und zwar die Komposition der Funktion Post mit der Funktion Pre. Die Funktion Post ist die Funktion, die darüber entscheidet, ob der Eingabewert $v \in \mathbb{R}$ zu einer Aktivierung des Neurons führt. Wir hatten uns bei der Aktivierungsfunktion auf die *ReLU*-Funktion festgelegt. Daher ergibt sich folgende Definition.

$$\mathrm{Post}(v) := \mathrm{ReLU}(v), \quad v \in \mathbb{R} \tag{3}$$

Die Argumente der Funktion Pre sind die n Eingänge in das Neuron. Der Begriff des Neurons lässt dabei jedoch offen, wie viele Eingänge ein Neuron besitzt, solange es mindestens einen Eingang hat. Der Definitionsbereich der Funktion Pre ist, für ein $0 < n \in \mathbb{Z}$, also jeder Vektor $(x_1, \ldots, x_n)^\top =: \boldsymbol{x} \in \mathbb{R}^n$ – um künftig einen Vektor $\boldsymbol{v}$ von einem Skalar v unterscheiden zu können, notieren wir, so wie eben, Vektoren in Fettdruck oder schreiben $\vec{v}$; die Entitäten, wie die obigen $x_1, \ldots, x_n$, aus denen ein Vektor besteht, nennen wir *(Vektor-)Elemente* oder *(Vektor-)Komponenten* des entsprechenden Vektors. Da, neben der Anzahl n, auch die Art und Zusammensetzung der Gewichte $(w_1, \ldots, w_n)^\top =: \boldsymbol{w} \in \mathbb{R}^n$ der Eingabe sowie der neuronen-eigene Bias $b \in \mathbb{R}$ von Neuron zu Neuron variieren kann, handelt es sich bei der Funktion Pre um eine – mit n, w und b – parametrisierte Funktion, die sich nach dem Gesagten wie folgt angeben lässt.

$$\mathrm{Pre}(\boldsymbol{x}; n, \boldsymbol{w}, b) := \sum_{i=1}^{n} w_i x_i + b = \boldsymbol{w}^\top \boldsymbol{x} + b \tag{4}$$

Um eine möglichst kurze Formel für eine Neuronenfunktion zu haben, definieren wir für die entsprechende Funktionskomposition die folgende Funktionenfamilie f.

$$f(\boldsymbol{x}; n, \boldsymbol{w}, b) := \mathrm{Post}(\mathrm{Pre}(\boldsymbol{x}; n, \boldsymbol{w}, b)) = \mathrm{ReLU}(\boldsymbol{w}^\top \boldsymbol{x} + b) \tag{5}$$

Wir sprechen insbesondere dann von einer Familie von, unterschiedlich parametrisierten, Neuronenfunktionen f, wenn wir es mit einem System von Neuronen zu tun haben, wie dies etwa bei Neuronalen Netzen der Fall ist. Zur Unterscheidung der individuellen Neuronenfunktionen verwenden wir dann indizierte Funktionssymbole und lassen, sofern sich diese aus dem Kontext ergeben, die Parameter weg. Die Bezeichnung $f_{i,j}(\boldsymbol{x})$ ist dafür ein mögliches Beispiel. Wir wollen nun klären, was es mit dem Konzept Neuronaler Netze auf sich hat.

2.2 Neuronale Netze als Graphen

Ein sprachanalytischer Ansatz liefert uns einen ersten Hinweis: Neuronale Netze sind Netze, oder Netzwerke, bestehend aus den eben definierten Neuronen. Genau wie wir bei den hier verwendeten Neuronen streng genommen von *künstlichen* Neuronen sprechen müssten, so müssten wir bei Neuronalen Netzen von *künstlichen* Neuronalen Netzen sprechen; der Einfachheit halber bleiben wir aber in der Regel bei der Kurzform, meinen aber stets die ausführliche Form. Aus der sprachlichen Analyse erhellt indes nicht, was mit „Netzen“ gemeint ist und wie Neuronen darin firmieren könnten.

Unter einem Netzwerk, kurz Netz, wollen wir hier einen zusammenhängenden gerichteten Graphen $G = (V, E)$ verstehen. Ein Graph G besteht demnach aus einem Tupel der Menge V – den sogenannten *Knoten*, englisch *vertices*, Plural von *vertex* – und einer Menge *gerichteter Kanten* $E \subseteq \{(v, w) \mid v, w \in V\}$, für englisch *edges*. Für jede Kante $(v, w) \in E$ mit $v, w \in V$ gibt es eine Verbindung von Knoten v, dem *Vorgänger*, zu Knoten w, dem *Nachfolger*. Eine Verbindung von w zu v gibt es, umgekehrt, dann und nur dann, wenn (w, v) ebenfalls Element der Menge E ist. Ein *gewichteter* gerichteter Graph verfügt zusätzlich über eine *Gewichtsfunktion* g, von lateinisch *gravitas*, die jeder Kante aus E eine Reelle Zahl zuordnet. Wir sagen, dass es einen *Pfad* von Knoten v aus V nach w aus V gibt, genau dann wenn die Kante (v, w) in E liegt oder es einen Knoten u in V gibt, sodass die Kante (v, u) in E liegt und es einen Pfad von u nach w gibt. Ein *Zyklus*, Zirkel oder Kreis ist ein Pfad in dem der erste Knoten des Pfades mit dem letzten Knoten des Pfades identisch ist. Ein Graph $G = (V, E)$ ist *zusammenhängend*, falls für alle v, w aus V es einen Pfad von v nach w im Graphen $G' = (V, E')$ gibt, wobei E' die Kantenmenge ist, die aus E entsteht, indem man zu jeder Kante (v, w) aus E auch die Kante (w, v) hinzufügt. Graphisch werden Graphen $G = (V, E)$ wie folgt repräsentiert. Knoten werden als Kreise gezeichnet in deren Mittelpunkt eine Markierung, zumeist mit der Funktion des jeweiligen Knotens, steht, worauf wir gleich zu sprechen kommen werden. Eine gerichtete Kante $(v, w) \in E$ wird als ein Pfeil von der Knotenrepräsentation von v, wir sagen auch aus v *auslaufend*, zur Knotenrepräsentation von w, wir sagen auch

in w *einlaufend*, gezeichnet; das zugehörige Kantengewicht $g((v,w))$, wird in der Nähe, meist oberhalb, der entsprechenden Kante notiert. Fehlende Kantengewichte stehen implizit stets für ein Kantengewicht von 1. Die Abbildung 2 ist zugleich die Repräsentation eines Perzeptrons in Form eines Graphen.

2.3 Syntax und Semantik Neuronaler Netze

Auf „syntaktischer“ Ebene entstehen Neuronale Netze nun wie folgt. Der Ausgang eines Neurons, was seinem *R-System* entspricht, wird zugleich Eingangsreiz eines anderen Neurons, was somit einem Knoten in dessen *S-System* entspricht. Auf diese Weise können Neuronen-Ketten, Neuronen-Kreise, kurz Neuronen-Netze, also Neuronale Netze, entstehen. Die Menge aller Knoten die in solchen Netzen keine Vorgänger besitzen – mit Ausnahme der Eins-Knoten, welche den neuronen-eigenen Bias transportieren, vergleiche hierzu Abbildung 2 –, werden *Eingabe-Knoten* des Netzes genannt. Knoten ohne Nachfolger bilden die Menge der *Ausgabe-Knoten* des Neuronalen Netzes. Wir gehen stets davon aus, dass über den Kanten eine Ordnung so definiert ist, das zu jeder Kante $(v,w) \in E$ feststeht, die wievielte in w einlaufende Kante sie ist; sei $\{(v_1,w),\ldots,(v_n,w)\} \subseteq E$ die Menge aller in w einlaufenden Kanten, so lässt sich diese Menge demnach auch als Vektor $\boldsymbol{v} := (v_1,\ldots,v_n)^\top$ angeben.

Mit Albarghouthi, [Alb21], aber darüber hinausgehend, spezifizieren wir die Semantik derartiger Netze wie folgt. In den Graphen $G = (V,E)$ Neuronaler Netze werden Operationen über die Reellen Zahlen wie folgt abgebildet. Mit Ausnahme der Netz-Eingabeknoten, bedeuten die Knoten Funktionen. Netz-Eingabeknoten, das sind die nicht-biasvermittelnden Knoten ohne Vorgänger, stehen für reellzahlige Variablen. Der Wert eines Eingabe-Knotens ist der Wert der Belegung der Variablen, für die der Knoten steht und zwar in Form derjenigen nullstelligen Funktion, die invariant den Wert der Belegung zurückgibt. Der Wert jedes biasvermittelnden Eins-Knotens, ist der Wert der nullstelligen Funktion, die invariant die Zahl 1 zurückgibt. Sei evalV die Bezeichnung für die Funktion, die einen Knoten auf seinen Wert abbildet. Sei ähnlich evalE die Bezeichnung der Funktion, die zu jeder Kante deren Wert angibt. Kanten $(v,w) \in E$ transferieren den Wert des Knotens v multipliziert mit dem Kantengewicht $g(v,w)$ in den Knoten w. Konkret ist der Wert einer Kante $(v,w) \in E$ demnach $\mathrm{evalE}((v,w)) := g((v,w)) \cdot \mathrm{evalV}(v)$. Schließlich ist der Wert eines Knotens $v \in V$ bestimmt als $\mathrm{evalV}(v) := \mathrm{apply}(v,\boldsymbol{a})$, wobei die Funktion apply die Funktion, für die der Knoten v steht, anwendet, appliziert, auf den, möglicherweise null-dimensionalen, Argumentvektor $\boldsymbol{a}$. Der Argumentvektor für Knoten v ist dabei definiert als $\boldsymbol{a} := \big(\mathrm{evalE}((u_1,v)),\ldots,\mathrm{evalE}((u_n,v))\big)^\top$, mit $\{(u_1,v),\ldots,(u_n,v)\} \subseteq E$, $n \geq 0$, als Menge aller in v einlaufenden Kanten. Man beachte, dass die Funktion Pre in Abbildung 2 in dieser Semantik einfach durch die Summe über alle einlaufenden Kantenwerte beschrieben ist.

Eine kompaktere Repräsentation eines Neurons in einem Graphen erhalten wir, wenn wir das *A*- und *S-System*, wie wir dies oben ja bereits getan hatten, zur Funk-

tion $f(\boldsymbol{x}) := \mathrm{ReLU}(\sum_{i=1}^{n} w_i x_i + b)$ komponieren, wobei wir auch die Parameter $w_1, \ldots, w_n$ und b in die Funktion verlagern. Wir nennen dies die einfache oder vereinfachte graphische Repräsentation eines Neurons. Das Ergebnis ist in Abbildung 3 dargestellt.

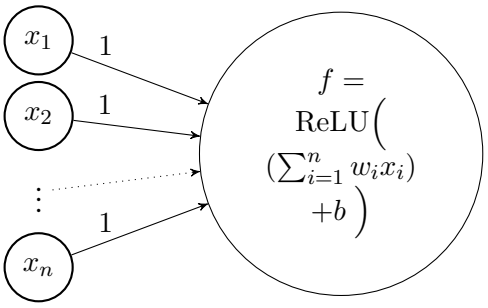


Abbildung 3: Vereinfachte Darstellung eines Neurons als Graph

Bisher haben wir offengelassen, auf welche Weise Neuronen in einem Neuronalen Netz untereinander verknüpft werden. Das Resultat der Spezifikation solcher Verknüpfungsregeln wird die *Architektur* des Netzes genannt, [Alb21]. In der vorliegenden Arbeit interessieren wir uns nur für einen Architekturentwurf, den wir im Folgenden vorstellen werden.

In der uns interessierenden Architektur sind die Knoten der jeweiligen, immer als zusammenhängend angenommenen, Neuronalen Netze stets in Schichten, englisch *layers*, angeordnet. Eine Schicht, in Bezug auf die vereinfachte graphische neuronale Darstellung, zeichnet sich wie folgt aus. Die Menge aller Eingabe-Knoten des Netzes, also der Menge aller Knoten ohne Vorgänger, bilden eine Schicht, die sogenannte *Eingabeschicht*. Die Menge aller Knoten ohne Nachfolger bilden eine Schicht, welche die *Ausgabeschicht* heißt. Alle anderen Knotenteilmengen V_ℓ müssen, um als sogenannte versteckte Schicht, englisch *hidden layer*, zu qualifizieren, folgende Bedingungen erfüllen. Die Menge aller Nachfolger-Knoten jedes Knotens in V_ℓ bilden eine Schicht und die Menge aller Vorgänger-Knoten jedes Knotens in V_ℓ bilden eine Schicht. Außerdem besitzt kein Knoten einer Schicht einen Nachfolger respektive Vorgänger in derselben Schicht. Die Größe einer Schicht ist mit der Anzahl der Knoten in der Menge, die die Schicht umfasst, identisch; sie kann von Schicht zu Schicht variieren, muss aber stets größer als 0 sein.

Der Struktur des Gebildes als verknüpfter Perzeptronen-Schichten geschuldet, wird diese Architektur auch *Multilayer Perceptron*, kurz *MLP* genannt, [Alb21]. Da sie außerdem als zyklenfrei angenommen wird, die Netzeingabe also immer nur – „vorwärts“ – in Richtung der Ausgabeschicht weiterverarbeitet wird, firmiert sie alternativ unter dem Namen *Feed-Forward Neural Network*, abgekürzt *FFN*. Man beachte, dass die Definition impliziert, dass Vorgänger- und Nachfolger-Schicht jeweils vollständig vermascht sind, was jedoch durch die Wahl von nullwertigen Kanten-Gewichten modifiziert werden kann: Eine nullgewichtete Kante,

die in einen Neuronenknoten einläuft, ist von einer an dieser Stelle nicht-vorhanden Kante funktional nicht zu unterscheiden. Wenn wir im Folgenden von Neuronalen Netzen sprechen, so meinen wir, falls nichts Gegenteiliges vereinbart wurde, stets Neuronale Netze mit der *FFN*-Architektur. In Abbildung 4 ist ein kleines *FFN*-Netz dargestellt.

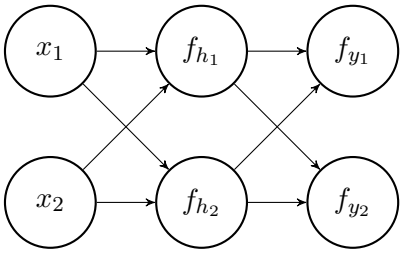


Abbildung 4: Ein einfaches Neuronales Netz N mit einer Eingabeschicht $(x_1, x_2)^\top$, den Neuronenfunktionen-Knoten f_{h_1} und f_{h_2} als einzigem *Hidden Layer* des Netzes, sowie den Knoten f_{y_1} und f_{y_2} als Ausgabeschicht. Netz N stellt die Funktion $N((x_1, x_2)^\top) = (y_1, y_2)^\top$, bei welcher $y_1 := f_{y_1}\Big(\big((f_{h_1}((x_1, x_2)^\top), f_{h_2}((x_1, x_2)^\top)\big)^\top\Big)$ gilt sowie des Weiteren $y_2 := f_{y_2}\Big(\big((f_{h_1}((x_1, x_2)^\top), f_{h_2}((x_1, x_2)^\top)\big)^\top\Big)$, graphisch dar.

Wenn wir auf ein so beschaffenes Neuronales Netz N, wie es etwa in Abbildung 4 abgebildet ist, die eingeführte Semantik für Neuronale Netze anwenden, so erkennen wir, dass es sich bei einem Neuronalen Netz um eine, aus vielen einzelnen Neuronenfunktionen komponierte, Funktionen handelt. Die Eingabeknoten $x_1, \ldots, x_n$ von N stehen für die n-dimensionalen vektorwertigen Urbilder $\boldsymbol{x}$ der Funktion und das m-dimensionale vektorwertige Ergebnis der m Knoten der Ausgabeschicht, $\boldsymbol{y}$, steht für den Wert der Funktion N – das ist, sein Bild – bei Eingabe $\boldsymbol{x}$. Wir können zu einem gegebenen Netz N, wenn wir seine Bedeutung als Funktion in den Blick nehmen, also immer $N(\boldsymbol{x}) = \boldsymbol{y}$ schreiben, wobei wir $\boldsymbol{x}$ den *Eingabevektor*, oder nur, kurz, die *Eingabe*, für N nennen und $\boldsymbol{y}$ den *Ausgabe-* oder *Ergebnis-Vektor* von N, abgekürzt auch nur *Ausgabe* oder *Ergebnis* von N. Wir halten die Erkenntnis dieses Absatzes in folgendem Satz fest.

Satz 1. *Sei N ein Neuronales Netz der FFN-Architektur mit $n > 0$ reellwertigen Eingabeknoten und einer reellwertigen Ausgabeschicht der Größe $m > 0$. Dann repräsentiert N eine Funktion $N : \mathbb{R}^n \to \mathbb{R}^m$, die durch die Semantik Neuronaler Netze für N als Graph festgelegt ist.*

Beweis. Die Behauptung folgt direkt durch Induktion über die Größe und Anzahl der Schichten des Netzes sowie der Bedeutung der Funktionskomposition, wonach das Ergebnis einer Funktionskomposition eine Funktion ist. Induktionsanker ist das Netz, welches aus genau einem Knoten in der Eingabeschicht sowie aus genau einem Neuronenfunktionenknoten in der Ausgabe komponiert ist. □

Der Einfachheit halber werden wir künftig auch kurz sagen, dass Neuronale Netze, im Sinne des obigen Satzes, Funktionen *sind*.

2.4 Interpretationen Neuronaler Netze

Kommen wir noch einmal zurück auf das Eingangszitat zu diesem Kapitel. Rosenblatts Vision war die Konstruktion einer Perzeptronen-Apparatur, die in menschenähnlicher Art und Weise das Wahrnehmen (*perception*), gar das (Wieder-)Erkennen (*recognition*) emulieren können sollte; gleichzeitig sollte sie in der Lage sein Begriffe zu entwickeln (*concept formation*) und die Fähigkeit besitzen, aus der Erfahrung zu lernen und das Erlernte zu generalisieren (*the ability to generalize from experience*). Mit den, insbesondere auch über die *FFN*-Architektur hinausgehenden, Neuronalen Netzen ist Rosenblatts Vision – in einer bestimmten Interpretation – Wirklichkeit geworden. Wir wollen im Folgenden diese Interpretation angeben.

Um zu verstehen, in welcher Weise bei einer „Apparatur" der Art eines Neuronalen Netzes überhaupt von „Wahrnehmung" und „Erkennen" die Rede sein kann, betrachten wir zunächst Folgendes. Gegeben sei die acht Binärziffern umfassende Bit-Folge $z = 10101100$. Was bedeutet, wofür steht, z? Man mag zunächst meinen, z stehe für die Dezimalzahl 172. Andererseits handelt es sich möglicherweise um die Zweierkomplement-Darstellung einer acht Bit breiten Ganzzahl. Dann wäre z die negative Zahl -84. Vielleicht ist es aber die Festkommanotation, bei der die ersten vier Binärziffern für den Vor- und die letzten vier für den Nachkommateil stehen: $z = 10.75$. Eventuell haben wir es hier aber auch mit einer Mengenbeschreibung über acht geordnete Gegenstände zu tun und z bedeutet – in der höchstwertige-Ziffer-zuerst MSB-Interpretation – die Menge, die den ersten, dritten, fünften und sechsten Gegenstand enthält. Wir sehen, dass Variablen der Art von z nur dann etwas bedeuten, wenn wir den Werten, die die Variable annehmen kann, eine Bedeutung zuordnen.

Ganz analog verhält es sich mit dem Urbild- und dem Bild-Raum Neuronaler Netze. Ein n-dimensionaler reellwertiger Eingabevektor $\boldsymbol{x}$ aus dem Urbildraum eines Neuronalen Netzes N und dem zugehörigen reellwertigen m-dimensionalen Ausgabevektor $\boldsymbol{y}$ aus dem Bildraum von N kann nur dann sinnvoll interpretiert werden, wenn zuvor festgelegt wurde, was die Elemente des Ein- und Ausgabevektors jeweils bedeuten. Eine naheliegende Familie von Interpretationen des Eingabevektors interpretiert die Elemente des Eingabevektors als reellwertige Kodierungen von Signalen aus der empirischen Außenwelt: Akustische oder optische Daten, Lage- oder Beschleunigungsdaten oder was dergleichen mehr sein mag. Im Fall der Lage- und Beschleunigungsdaten könnten die „Antworten", in Form des Ergebnisvektors $\boldsymbol{y}$, des Netzes kodierte Steueranweisungen sein, die es ermöglichen, eine stabile Bahn beizubehalten, das Gleichgewicht zu wahren oder Ähnliches.

Im Fall der optischen oder akustischen Dateneingänge in das Netz, hat sich für den Ausgabevektor eine Interpretation eingebürgert, die der Familie dieser Netzarchitektur einen eigenen Namen, nämlich die Bezeichnung als *Klassifizierer*,

eingebracht hat, [Alb21]. Dabei ordnet man jedem der m Vektorelement-Indizes $\{1, \ldots, m\}$ des Ausgabevektors $\boldsymbol{y} = (y_1, \ldots, y_m)^\top$ eine feste Klasse zu. So könnte der Eingabevektor etwa für ein Rastergrafik-Bild stehen, in denen die Vektorelemente die Graustufen der einzelnen Pixel des jeweiligen Bildes darstellen. Als Klassen könnten für $m := 3$ etwa gewählt werden: 1 ist die Klasse die für „Hund“ steht, 2 repräsentiert „Katze“ und 3 kodiert „Maus“. Der Ausgabevektor bedeutet in dieser Interpretation dann eine Art „Votum“ des Netzes, als was es die Eingabe klassifiziert: Je höher der Wert des Vektorelements y_i desto mehr „Stimmen“ erhält die Klasse i. Die Klasse mit den meisten „Stimmen“, also dem Index des höchstwertigen Vektorelements im Ausgabevektor, ist die Klassifikation von N für die Eingabe $\boldsymbol{x}$.

Definition 1. *Sei N ein FFN-Klassifizierer mit $m > 1$ Klassen. Sei für beliebige Vektoren $\boldsymbol{v} = (v_1, \ldots, v_\ell)^\top$ die Funktion* $\arg\max$ *beschrieben durch* $\arg\max(\boldsymbol{v}) = i$*, falls $i \in I$, mit $I = \{1, \ldots, \ell\}$ und für alle $j \in I$ gilt, dass $v_i \geq v_j$; gibt es mehr als ein i, das die Bedingungen erfüllt, so gibt* $\arg\max$ *nichtdeterministisch aber konstant irgendeines dieser i zurück. Wir sagen, dass N die Eingabe $\boldsymbol{x}$ als c, mit $c \in \{1, \ldots, m\}$, klassifiziert, falls*

$$c = \arg\max(N(\boldsymbol{x})), \tag{6}$$

Klassifizierer dieser Art sind nun ganz auf der Linie von Rosenblatts Vision: Eingaben in das Netz sind in diesem Fall sensorische Daten, die das Netz über den Eingabevektor „wahrnimmt“ und als Antwort auf die Eingabe eine Klasse angibt, zu der die Eingabe mutmaßlich gehört. Wenn das Neuronale Netz zuverlässig Eingaben korrekt klassifiziert – das heißt, die Eingaben „erkennt“ –, können wir sagen, dass das Netz einen Begriff (*concept*) davon hat, wofür die Klassen stehen. In obigem Beispiel könnte der Tierklassifizierer etwa einen Begriff von Hunde-, Katzen- und Mäuse-Bildern haben.

Eine andere Familie von Interpretationen interpretiert die Eingabe- und Ausgabevektoren jeweils als das, was sie tatsächlich sind, nämlich als Vektoren von Reellen Zahlen, wobei das Netz vom n-dimensionalen Vektorraum in den m-dimensionalen Vektorraum abbildet. Die Idee dahinter ist, Neuronale Netze als Funktionsapproximatoren zu verwenden, um eine beliebige andere Funktion, die von $\mathbb{R}^n$ nach $\mathbb{R}^m$ abbildet, zu emulieren. Abbildung 5 zeigt, als ein Beispiel für ein solches Neuronales Netz, ein Netz, welches die xor-Funktion – exakt – nachbildet; es ist, graphisch aufbereitet, aus [GBC16] entliehen. In Anlehnung an die Klassifizierer können wir sagen, das ein so beschaffenes Netz einen Begriff von der Funktion hat, die es emuliert. Was Rosenblatt, [Ros57], im Besonderen meint, wenn er davon spricht, dass Perzeptronen Begriffe entwickeln könnten, geht aus dem Text jedoch nicht eindeutig hervor; für mögliche Interpretationen, verweisen wir auf den Eintrag „*Concepts*“ der *Stanford Encyclopedia of Philosophy*, [ML23].

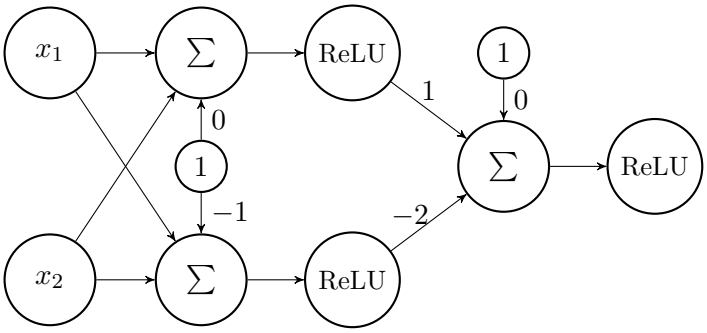


Abbildung 5: Neuronales Netz N, welches die xor-Funktion exakt inkorporiert: $N((x_1 := 0, x_2 := 0)^\top) = \max\{0, (\max\{0, (x_1 := 0)\cdot 1 + (x_2 := 0)\cdot 1 + 1\cdot 0\}\cdot 1) + (\max\{0, (x_1 := 0)\cdot 1 + (x_2 := 0)\cdot 1 + 1\cdot -1)\}\cdot -2) + 1\cdot 0\} = 0$; und analog $N((0,1)^\top) = 1$, $N((1,0)^\top) = 1$ und $N((1,1)^\top) = 0$. Die dem Knoten mit der Beschriftung $\sum$ zugeordnete Funktion ist die Summe über die Werte aller Kanten, die in den so beschrifteten Knoten eingehen. Kanten ohne Kantengewichtsbeschriftung haben ein Kantengewicht von 1. Das Beispiel stammt aus Goodfellow et. al., [GBC16].

2.5 Zwei Klassifiziererbeispiele

Um von den im letzten Abschnitt, in Definition 1, eingeführten Klassifizierern ein einfaches Beispiel zu haben, betrachten wir das in Abbildung 6 dargestellte Netzwerk N. Wir wollen folglich einen Klassifizierer N für ein aus zwei Pixeln – *Pixel* ist die englische Kurzform für *Picture Element*, „Bildelement“ – bestehendes Schwarz-Weiß-Bild, das ist, ein Bild, welches nur aus schwarzen, weißen und beliebigen grauen, zwischen schwarzen und weißen, Pixeln komponiert ist, vorstellen. In gängigen vorzeichenlosen 8-Bit-Ganzzahl-Repräsentationen von Graustufen auf Rechenanlagen, wird der kleinste Wert, 0, als weiß interpretiert und der größte Wert, 255 als schwarz. Die Werte 1 bis 254 dazwischen stellen Grautöne dar, die linear-monoton von weiß in schwarz übergehen. Zum Zwecke der Normierung werden alle diese Werte in Relation zum größten darstellbaren Wert gesetzt und entsprechend durch 255 geteilt, was die Grauskala somit als zwischen 0, dem weißen Pixel, und 1 dem schwarzen Pixel normiert zurücklässt. In dieser Arbeit werden wir stets von dieser normierten Grauskala ausgehen und unterstellen, dass die Skala nicht 256 Abstufungen, sondern beliebig viele Graduierungen zulässt. So wollen wir etwa einen normierten Grauwert von 0.1 als gültig zulassen, der, da $(25/255) < 0.1 < (26/255)$, nicht in das klassische 8-Bit-Grauschema passt; dass diese Werte bezüglich ähnlich großer Werte gegebenenfalls nicht diszerniert, unterschieden, werden respektive graphisch-visuell dargestellt werden können, ist schließlich ein Problem unseres Sehvermögens respektive unserer Anzeigegeräte, aber kein Problem dieser Werte. Man beachte, dass die hier vorgestellte Interpretation von 0 als weiß und 1 als schwarz der Intuition der (gleichzeitigen und gleichintensiven) Inkrementierung

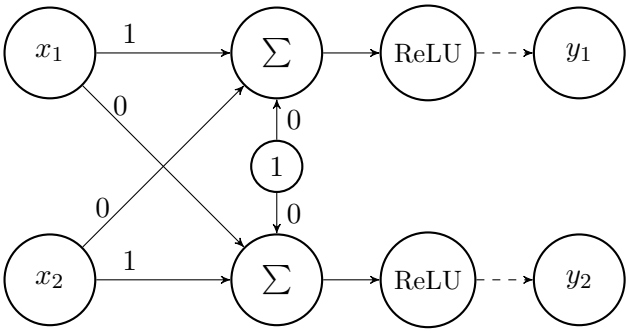


Abbildung 6: Neuronales Netz N, welches als Klassifizierer fungiert. Der Eingabevektor $\boldsymbol{x} = (x_1, x_2)^\top$ repräsentiert ein zwei Pixel großes schwarz-weiß-Bild. Bei einer Beschränkung der Eingabekomponenten auf $x1, x2 \in [0, 1] \subset \mathbb{R}$, wobei ein Wert von $x_i = 0$, mit $i \in \{1, 2\}$, ein ganz weißes Pixel und $x_i = 1$ ein ganz schwarzes Pixel und alle Werte $0 < x_i < 1$ einen entsprechend linear abgestuften Grauwert zwischen weiß und schwarz darstellen, spezifiziert die Klassifikation $c = \arg\max(N(\boldsymbol{x}))$ des Ausgabevektors $N(\boldsymbol{x}) = \boldsymbol{y} := (y_1, y_2)^\top$, für $c = 1$, dass $x_1 > x_2$, wonach x_1 ein dunkleres Pixel als x_2 ist sowie für $c = 2$, dass $x_2 > x_1$, wonach x_2 ein dunkleres Pixel als x_1 ist. Die gestrichelten Pfeile deuten an dass der Wert der *ReLU*-Knoten jeweils identisch ist mit y_1 beziehungsweise y_2.

der Intensitäten der Rot-Grün-Blau-Farbkanäle genau entgegengesetzt ist, wonach ein höherer Wert für „mehr Licht“, und somit die 0 für schwarz und die 1 für weiß, stehen müsste. Nach dieser Lesart beschreibt unsere Grau-Interpretation hier stets das Negativbild.

In Abbildung 7 ist eine Beispielklassifikation des Klassifizierers N für den Eingabevektor $\boldsymbol{x} = (0.25, 0.75)^\top$ dargestellt. Diese Eingabe wird von N als zur Klasse $c = 2$ gehörig klassifiziert. Im Speziellen haben wir, in der oben gegebenen Interpretation, nach der die Eingabewerte für die jeweiligen Grauwerte der Pixel x_1 und x_2 des Eingabebildes $\boldsymbol{x}$ stehen, dass die Ausgabe $\boldsymbol{y} = (y_1, y_2)^\top$, welche im Beispiel konkret $\boldsymbol{y} = (0.25, 0.75)^\top$ lautet, als $c = 2$ klassifiziert wird, da, nach unserer Interpretation der Ausgabe als Ausgabe eines Klassifizierers, der Index des (eines) größten Wertes hier der Index 2 ist, denn $y_2 \geq y_1$. Damit „drückt“ der Klassifizierer N „aus“, dass der zweite Pixel, durch x_2 beschrieben, dunkler ist, als der durch x_1 beschriebene erste Pixel, denn, in obigem Gleichnis reformuliert, y_1 hat mit 0.25 „weniger Stimmen bekommen“ als „das Votum“ für y_2, mit einem Wert von 0.75.

Der eben als Beispiel beschriebene Klassifizierer dient aufgrund seiner Einfachheit – man beachte etwa, dass sich die Matrix der Gewichte des Netzes als Identitätsmatrix darstellen lässt – lediglich didaktischen Zwecken. Für ein komplexeres und realitätsnäheres Beispiel für eine Familie von Klassifizierern wollen wir an dieser

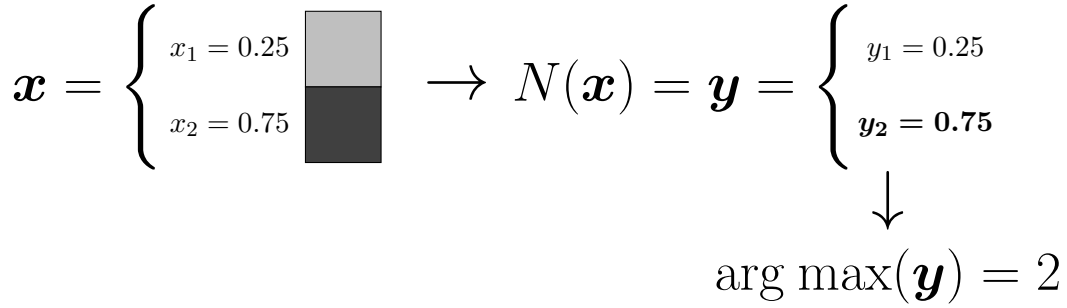


Abbildung 7: Beispieleingabe und zugehörige Ausgabe für den Klassifizierer N aus Abbildung 6. Für die im Text gegebene Interpretation der Ein- und Ausgabegrößen klassifiziert N das Eingabebild $\boldsymbol{x}$ als Klasse $c = 2$ und drückt, in der gegebenen Interpretion, somit die Behauptung aus, dass das durch x_2 beschriebene Pixel dunkler ist, als das durch x_1 beschriebene Pixel.

Stelle Klassifizierer von handgeschriebenen Arabischen Dezimalziffern – also die Ziffern von Null, 0, bis Neun, 9, – vorstellen. Eine umfangreiche Sammlung von vorklassifizierten – man spricht hier vom *Label*, einem Etikett, welches korrekt angibt, welche Ziffer auf dem entsprechenden Bild zu sehen ist – handschriftlichen Ziffern – Bildern dieser Art wurden in der sogenannten *MNIST database*, *„Modified National Institute of Standards and Technology Database“*, zusammengetragen. Dieser – unter anderem in der Programmbibliothek *Keras* unter der Rubrik *datasets* in der Programmiersprache *Python* – frei verfügbare Datensatz besteht aus 60000 Trainingsdaten und weiteren 10000 Testdaten. Ein Datum besteht dabei jeweils aus einem (28×28)-Pixel großen Bild, auf welchem, vermittelst der oben besprochen 8-Bit-Graustufenskala, die handgeschriebene Ziffer abgebildet ist einerseits und dem Label zu diesem Bild, der dargestellten Ziffer, andererseits. Als Beispiel eines solchen Datums ist in Abbildung 8 das als „3“ gelabelte Bild an Position 12345 des Trainingsdatensets dargestellt.

Anders als bisher, werden die Indizes der Eingabe und der Ausgabe nun als bei Index 0 beginnend angenommen, sodass der Eingabevektor $\boldsymbol{x}$ jetzt folglich stets als $\boldsymbol{x} = (x_0, \ldots, x_{783})^\top$ – man errechne 28 (Höhe eines *MNIST*-Bildes) mal 28 (Breite eines *MNIST*-Bildes) als 784 – und der Ausgabevektor fürderhin als $\boldsymbol{y} = (y_0, \ldots, y_9)^\top$ spezifiziert sind. Die Klassifikations-Interpretation dieses Vektors ist auch weiterhin $c = \arg\max(\boldsymbol{y})$, wobei die Indizierung der $\arg\max$-Funktion nun jedoch ebenfalls bei 0 beginnt: Als Indexbeschreibung kann c jetzt auch 0 sein und zwar dann, wenn für alle $y_i \in \{y_1, \ldots, y_9\}$ gilt, dass $y_0 \geq y_i$. Als eine weitere Neuerung ist die Aktivierungsfunktion in der letzten Schicht – der Ausgabeschicht – stets nicht die *ReLU*-, sondern lediglich die Identitätsfunktion, welche pro Knoten v dieser Schicht jeweils noch einmal die, gemäß der Kantengewichte, gewichtete Summe der Vorgänger bildet und den knoteneigenen Bias addiert *nicht jedoch* anschließend

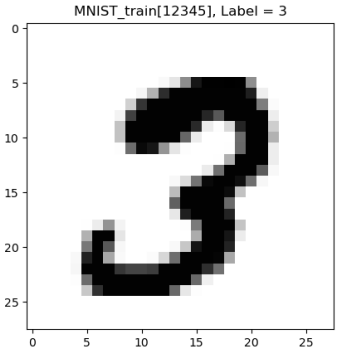


Abbildung 8: Beispiel eines *MNIST*-Trainingsdatums. Abgebildet ist das 12346ste (28×28)-Pixel große 8-Bit-Graustufen-Bild des Trainingsdatensatzes. Sein Label ist 3. Das Pixel x_k, für $k = (i \cdot 28 + j)$, befindet sich in der j-ten Spalte der i-ten Zeile, $i, j \in \{0, \ldots, 27\}$, der Abbildung.

das *ReLU*-typische $\max\{0, v\}$ anwendet. Auf diese Weise kann der Ausgabevektor nun auch negative Werte annehmen. Der Ausgabevektor $\boldsymbol{y} = (y_0, \ldots, y_9)^\top$ eines Klassifizierers – zuweilen auch, englisch, *model* genannt – für *MNIST*-Bilder zum Beispielbild aus Abbildung 8 könnte nun für einen Klassifizierer – konkret einen Klassifizierer namens `model784x10x10_float64_9355`, der über einen einzigen *Hidden Layer*, bestehend aus zehn Knoten, verfügt, bei dem alle Gewichte und Bias doppelt präzise (in Form von 64-Bit-Fließkommazahlen) dargestellt sind und der eine Genauigkeit, englisch *accuracy*, von 93.55% aufweist (der also von den, während des Trainings nicht angetroffenen, 10000 Testdaten 9355, gemäß des Labels, korrekt klassifiziert) – wie folgt aussehen.

$$\boldsymbol{y} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \boldsymbol{y_3} \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \end{pmatrix} = \begin{pmatrix} -8.929642274789787 \\ -23.596549774828215 \\ -3.7394853911872223 \\ \boldsymbol{-1.3908371337303267} \\ -27.944844737162022 \\ -4.27947218326491 \\ -32.40428998764842 \\ -8.374318597583049 \\ -5.162592733292586 \\ -5.644606829861474 \end{pmatrix}$$

Hierbei nimmt das vierte Vektorelement von $\boldsymbol{y}$, die vierte Vektorkomponente, y_3, mit – bei 0 beginnender Indizierung – Index 3 den größten Wert aller Vektorelemente an. Entsprechend lautet der Wert der Funktion $\arg\max$ für $\boldsymbol{y}$ dann konsequenterweise auch $c = \arg\max(\boldsymbol{y}) = 3$, was unserer Klassifikations-Interpretation zufolge bedeutet, dass das Netz das Eingabebild aus Abbildung 8 als die Ziffer Drei, 3, klas-

sifiziert.

Im nun folgenden Abschnitt sei kurz skizziert, auf welche Weise einem Neuronalen Netz ein derartig intelligentes Verhalten – etwa handgeschriebene Ziffern korrekt klassifizieren zu können – antrainiert werden kann.

2.6 Neuronale Netze als intelligente Apparaturen

Um das Kapitel abzuschließen, wollen wir noch kursorisch klären, wie der zweite Teil der Vision Rosenblatts – also, dass Perzeptronen-Maschinen aus der Erfahrung lernen können und imstande sind, das Gelernte zu generalisieren – Wirklichkeit werden kann. Es sei an dieser Stelle ausdrücklich betont, dass „intelligentes“ Verhalten Neuronaler Netze keine Voraussetzung für deren Verifikation bezüglich ihrer Eigenschaften darstellt. Wie Neuronale Netze intelligentes Verhalten erlernen können, und ob ein vorliegendes Netz domänenspezifische Intelligenz besitzt, ist für die vorliegende Arbeit also nur insofern von Interesse, als es für die Formulierung von Eigenschaften Relevanz besitzt, deren Verifikation uns ein Anliegen sein wird.

Das Konzept des sogenannten *Deep Learnings*, umfasst den Versuch, „tiefen“ Neuronalen Netzen, also – wie man etwa in Kapitel 6, im Buch von Goodfellow et. al. „Deep Learning“, [GBC16], sowie im *Nature*-Artikel „Deep learning, Review“ von LeCun, Bengio und Hinton, [LBH15], liest – Netzen mit mehreren, also mindestens einem, *Hidden Layer*, ein gewünschtes Verhalten anzulernen. Dabei verstehen wir unter dem *Verhalten* eines Neuronalen Netzes N, und analog dem Verhalten von Funktionen ganz allgemein, dessen Disposition, den jeweiligen Eingaben in das Netz diejenige Ausgabe zuzuorden, die über die Funktion, die N darstellt, beschrieben ist. Wie haben gesehen, dass die Funktion N, vollständig determiniert ist durch die Art der Komposition der Neuronenfunktionen, aus denen N besteht. Die Neuronen ihrerseits, so hatten wir uns klar gemacht, sind parametrisierte Funktionen, deren Verhalten, bei konstanter Eingabe, also abhängig ist, von der Wahl der Parameter: dem neuroneneigenen Bias und den Kantengewichten zwischen *S*- und *A-System* jedes Neurons. Es folgt, dass ein Neuronales Netz parametrisiert ist über die Gesamtheit der Werte der Kantengewichte und der Werte der Bias der es konstituierenden Neuronen. Um also das Verhalten eines Netzes zu verändern, müssen die Parameter des Netzes geändert werden. Für den Prozess des Lernens, das heißt, die Überführung von weniger gewünschtem Verhalten des Netzes hin zu gewünschterem Verhalten, bedeutet das eine dies bewirkende Manipulation der Parameter des Neuronalen Netzes.

Was die Arten des Lernens anbelangt, so können wir, etwa mit Goodfellow et. al., [GBC16], zwei Arten des Lernens unterscheiden. Im Falle des betreuten oder überwachten Lernens, englisch *Supervised Learning*, ist das Verhalten, welches wünschenswert ist, in Form von Beispielen gegeben. Die Beispiele sind hierbei konkret eine Menge von Tupeln, deren erstes Tupel-Element jeweils den Wert eines Eingabevektors enthält und deren zweites Tupel-Element jeweils den zu dieser Eingabe gewünschten Wert des Ausgabevektors enthält. Im Fall von Klassifikatoren, so hatten

wir im letzten Abschnitt gesehen, wird die zu einer Eingabe erwünschte Ausgabe als *Label* bezeichnet, da es die Eingabe gleichsam als das etikettiert, englisch *getting labeled as*, was sie darstellt. Wir können uns die Bedeutung dieser Beispielmenge auch so veranschaulichen, dass die darin enthaltenen Daten den „Erfahrungsfundus" des Betreuers, englisch *Supervisor*, darstellen und das Netz aus ebenjenen Erfahrungen lernen soll. Diese Beispiele erwünschten Verhaltens, sind eine Menge von Daten, die im Normalfall – wie wir dies im Fall der *MNIST*-Datenbank bereits gesehen hatten – noch einmal disjunkt und vollständig unterteilt wird in eine sogenannte *Trainingsmenge* und eine *Testmenge*. Erstere dient zum Anlernen oder „Training" des Netzes, letztere zum Test, inwieweit das Erlernte auf Eingaben generalisierbar ist, denen das Netz während des Trainings nicht begegnet ist. Die zweite Art des Lernens, das sogenannte *Unsupervised Learning*, verzichtet auf die explizite Angabe von Beispielen erwünschten Verhaltens – das Netz lernt nun nicht aus den Erfahrungen eines Betreuers, sondern muss beispielsweise aus eigenen Erfahrungen, etwa einer Entscheidung und der zugehörigen Reaktion der Umwelt auf diese Entscheidung, lernen. Da diese Art des Lernens für die vorliegende Arbeit ungegenständlich ist, verweisen wir für diesbezügliche Details auf Goodfellow et. al., [GBC16].

Wir wollen nun abschließend skizzieren, wie es möglich ist, dass Netze, die anfangs wenig Neigung zu gewünschtem Verhalten zeigen, dazu gebracht werden können, aus den Erfahrungen des Betreuers, also konkret den Trainingsdaten, zu lernen. Eine Form des Lernens besteht darin, Fehler zu machen und aus den begangenen Fehlern dergestalt zu lernen, ein Verhalten zu entwickeln, welches dazu führt, diese Fehler in der Zukunft möglichst nicht noch einmal zu begehen. Einen Fehler machen bedeutet im gegenwärtigen Kontext jedoch zuvorderst, zu einer Eingabe eines gewählten Beispiels aus der Trainingsmenge nicht die zugehörige, im Trainingsbeispiel als zweites Tupelelement spezifizierte, gewünschte Ausgabe anzugeben. Die Schwere des Fehlers bestimmt sich als der Abstand vom tatsächlichem zum gewünschten Ausgabevektor und wird über ein, für den Lernprozess als Parameter zu wählendes, Fehlermaß bestimmt, welches als *Error-*, *Loss-* oder *Cost-Function* bezeichnet wird. Um den Fehler künftig zu verringern, muss das Verhalten des Netzes verändert werden, welches, wie wir gesehen hatten, über die Parameter des Netzes, also den Werten der Kantengewichte und Bias der einzelnen Neuronen, kodifiziert ist.

Ein gängiges Lernverfahren, der sogenannte *Backpropagation*-Algorithmus, geht dementsprechend wie folgt vor. Zu einer Eingabe eines Trainingsbeispiels berechnet das Netz, in einem „Vorwärtsdurchlauf", englisch *forward propagation*, schichtenweise den Wert des Ergebnisvektors. Dieser wird daraufhin mit dem gewünschten Ergebnis zu dieser Eingabe verglichen und die Abweichung, also der begangene Fehler, ermittelt. Dieser Fehler beschreibt den induzierten Fehler der sich aus den in den verschiedenen Schichten „falsch" gewählten Kantengewichte und Bias ergibt. Um den Fehler also von der Ausgabeschicht auf die Fehler früherer Schichten zurückzuverfolgen, wird ein „Rückwärtsdurchlauf", englisch *back propagation*,

durch das Netz angestoßen, als deren Resultat feststeht, welchen Beitrag jeder einzelne Bias und jedes einzelne Kantengewicht jedes Neurons, und damit des Netzes insgesamt, am Fehler hatte. Die Gesamtheit dieser Informationen – wie würde sich der Fehler ändern wenn dieses spezifische Kantengewicht oder dieser spezielle Bias verändert würde – wird auch als *Gradient* bezeichnet und gibt die Richtung an, in die der Fehler verringert werden kann. Das Lernen besteht schließlich darin, einen Schritt in diese Richtung, die Richtung eines kleineren Fehlers, zu gehen, also die Parameterwerte so zu vergrößern oder zu verkleinern, wie im entsprechenden Gradienten spezifiziert. Für eine ausführlichere Darstellung sei auf Goodfellow et. al., [GBC16] verwiesen. Nielsen liefert eine sehr anschauliche Darstellung des *Backpropagation*-Algorithmus, [Nie15].

Da durch den *Backpropagation*-Algorithmus trainierte Netze das Potenzial haben, das Erlernte gut über im Training nicht angetroffene Daten zu generalisieren, [GBC16], können wir festhalten, dass Rosenblatts Vision, welche im Eingangszitat zum vorliegenden Kapitel ihren Ausdruck findet, in den, in diesem Kapitel ausgeführten, Hinsichten, wahr geworden ist.

3 Formalisierung Neuronaler Netze

> Etwas weitschweifige Formeln waren zum Ausdrucke complicirter Gedankenreihen leider manches Mal nicht zu vermeiden und ich kann mir lebhaft vorstellen, dass Manchem, der das Ganze nicht überschaut, die Resultate vielleicht wieder der aufgewandten Mühe nicht zu entsprechen scheinen werden.
>
> Ludwig Boltzmann,
> *Vorlesungen über Gastheorie* [Bol96]

> If one proves the equality of two numbers a and b by showing first that $a \leqq b$ and then that $a \geqq b$, it is unfair; one should instead show that they are really equal by disclosing the inner ground for their equality.
>
> Emmy Noether,
> *zitiert aus dem Kapitel „Emmy Noether (1935)", in Hermann Weyl, Levels of Infinity: Selected Writings on Mathematics and Philosophy* [Wey12]

In diesem Kapitel wollen wir zunächst eine Ordnung in unsere Neuronalen Netze bringen, indem wir die Menge der Kantengewichte und Bias systematisieren. Die Knoten des Netzes – darunter verstehen wir, konkret, neben den Eingabeknoten insbesondere die jeweiligen Knoten des *A-Systems* und des *R-Systems* sowie die Ausgabeknoten – sollen eindeutig identifizierbar sein, wofür wir ein *Marabou*-konformes Benennungssystem entwickeln wollen. Schließlich werden wir eine Funktion bestimmen, die ein gegebenes Neuronales Netz, oder dessen Semantik, in eine prädikatenlogische Formel übersetzt, die diese Semantik, gegeben eine naheliegende Interpretation, ausdrückt.

3.1 Eine kompakte Notation für Gewichte und Bias

Im letzten Kapitel hatten wir festgelegt, dass die Knoten eines Neuronalen Netzes stets in Schichten, englisch *Layer*, organisiert sind. Da die Eingabeschicht, eine Sonderrolle innerhalb dieses Schichtsystems spielt – nicht zuletzt im Hinblick darauf, dass in ihr keine *Neuronen*knoten firmieren –, bekommt sie den Index 0 zugewiesen; sie ist die nullte Schicht. Die erste Schicht ist also stets eine versteckte Schicht – im Folgenden zuweilen auch nur, gemäß dem Englischen, als *Hidden Layer* bezeichnet – oder bereits die Ausgabeschicht. Die Schichten werden, beginnend bei der Eingabeschicht aufsteigend nummeriert, sodass der größte Schichtenindex der Ausgabeschicht zukommt. Der Index der Ausgabeschicht ist identisch mit der *Tiefe* des Netzes, wobei die Tiefe, englisch *depth*, die Anzahl der Schichten des Netzes, abzüglich der Eingabeschicht, bezeichnet.

In jeder Schicht, so hatten wir vereinbart, gibt es eine fixe Reihenfolge der Knoten, sodass es stets sinnvoll ist, zu sagen: „Der i-te Knoten der ℓ-ten Schicht." – sofern

die ℓ-te Schicht existitiert und mindestens i Knoten umfasst. Jeder Knoten mit Index i einer Schicht mit Schichtenindex $\ell > 0$ hat stets jeden Knoten mit Index j der Schicht $\ell - 1$ als Vorgängerknoten. Die Kante, die den i-ten Knoten der Schicht ℓ mit dem j-ten Knoten der Vorgängerschicht $\ell - 1$ verbindet, besitzt, so wollen wir als Bezeichnungskonvention vereinbaren, das Kantengewicht $w_{\ell,i,j}$.

Abbildung 9: Benennungsvereinbarung für das Kantengewicht $w_{\ell,i,j}$, der Kante welche, für $\ell > 0$, den j-ten Knoten der Schicht $\ell - 1$ mit dem i-ten Knoten der Schicht ℓ verbindet.

Ganz ähnlich wollen wir mit den Bias verfahren und festlegen, dass der neuroneneigene Bias des i-ten Knotens in Schicht $\ell > 0$ – Knoten der Eingabeschicht besitzen, wie erinnerlich, keinen Bias – die Bezeichnung $b_{\ell,i}$ zugewiesen bekommt.

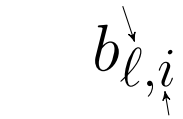


Abbildung 10: Benennungsvereinbarung für den Biaswert $b_{\ell,i}$, des i-ten Knotens der Schicht ℓ.

Diejenigen Schichten, welche mindestens so viele Knoten wie jede andere Schicht desselben Netzwerks enthalten, bestimmen die *Breite* oder Weite, englisch *width*, des Netzwerks: Die Breite gibt die Knotenkardinalität einer Schicht des betrachteten Netzes mit den meisten Knoten wieder.

Im vorigen Kapitel hatten wir gesehen, dass Verbindungen eines Knotens zu einem bestimmten Knoten in der Vorgängerschicht „gekappt" werden können, indem man das Gewicht der die Knoten verbindenden Kante auf Null setzt. Ein Knoten, der auf dies Weise *alle* Verbindungen zu Knoten in der Vorgängerschicht *und* Nachfolgeschicht abgebrochen hat, hat folglich keinen Einfluss auf das berechnete Ergebnis des Netzes und kann, ohne dass sich die Funktion, welche das Netz beschreibt verändern würde, entfernt werden – oder *hinzugefügt*. Einen solchen Knoten, der keinerlei Teilhabe am berechneten Ergebnis hat, wollen wir einen *Dummyknoten* nennen.

Um unsere Notation weiter zu vereinheitlichen, fügen wir – *hypothetisch* – jeder Schicht, inklusive der Ein- und Ausgabeschicht, Dummyknoten hinzu, ohne die Rei-

henfolge der Knoten zu verändern, welche bereits in dieser Schicht vorhanden sind. Dabei zählen wir, ausgehend vom höchsten Index eines regulären Knotens dieser Schicht, mit jedem zugefügten Dummyknoten den Knotenindex für diese Schicht weiter hoch, bis, und inklusive diesen, wir den Index erreicht haben, der der Breite des Netzes entspricht. In mindestens einer Schicht werden demnach keine Dummyknoten insertiert. Jeder Dummyknoten jeder Schicht wird mit jeweils einer nullgewichtigen Kante mit allen Knoten der Vorgängerschicht verbunden, außer der Dummyknoten ist ein Knoten der Eingabeschicht. Alle Knoten einer Schicht, welche keine Dummyknoten sind, verbinden sich mit den gegebenenfalls hinzugefügten Dummyknoten der Vorgängerschicht ebenfalls mit einer nullgewichtigen Kante. Man beachte, dass damit die Verbindungen der Dummyknoten in die Nachfolgeschicht bereits definiert sind. Das Auffüllen der Schichten mit Dummyknoten nennen wir auch *Zero-Padding*, oder auch nur *Padding*, was im Englischen einfach „Auffüllen“ bedeutet. Das Resultat des *Padding*s ist, dass in jeder Schicht des Netzes *width* viele Knoten existieren.

Um Gewichte und Bias in einer vereinigten „Datenstruktur“ führen zu können, ergänzen wir unsere Netze, wiederum hypothetisch, um einen weiteren Knoten, mit Index *width*$+1$, in jeder Schicht. Für jeden dieser Knoten in Schicht $\ell - 1 \geq 0$, außer der Ausgabeschicht, fügen wir Kanten, und insbesondere Kanten*gewichte*, in die Nachfolgeschicht ℓ, welche n Nicht-Dummyknoten besitzt, wie folgt hinzu. Die Kante, die den Knoten mit Index *width*$+1$ in Schicht $\ell - 1$ mit einem Knoten mit Index $1 \leq i \leq n$ in Schicht ℓ, das sind die Nicht-Dummyknoten dieser Schicht, verbindet, haben das Gewicht $b_{\ell,i}$; das entspricht genau dem Wert des Bias für diesen Knoten. Alle anderen Knoten in Schicht ℓ haben eine nullgewichtige Kante zu ihrem Vorgängerknoten mit Index *width*$+1$ in Schicht $\ell - 1$; eine Ausnahme hiervon bildet die Verbindung des *width*$+1$-ten Knotens der Schicht $\ell - 1$ mit dem Knoten mit Index *width*$+1$ in Schicht ℓ, welche das Kantengewicht 1 besitzt. Desweiteren sind alle Kanten – außer der eben genannten –, die in den *width*$+1$-ten Knoten in Schicht ℓ, aus der Schicht $\ell - 1$ kommend, einlaufen, nullgewichtig.

Es sei noch einmal nachdrücklich daraufhingewiesen, dass die eben besprochene Erweiterung des Netzes *hypothetisch*, oder konditional, zu verstehen ist, im Sinne von: „Angenommen, wir hätten unser Netz auf die so beschriebene Weise erweitert, dann . . ..“

Wir sind nun in der vorteilhaften Lage, die Verbindung zwischen zwei benachbarten Schichten unseres hypothetischen Netzes kompakt als Matrix darstellen zu können, wovon Abbildung 11 eine Anschauung geben soll.

Das Kantengewichtssystem zwischen den Knoten zweier benachbarter Schichten, sowie das Biassystem der letzteren Schicht lässt sich also stets in Form einer quadratischen Matrix repräsentieren. Und die Gesamtheit aller, in die entsprechende Reihenfolge gebrachten, solcher Matrizen beschreibt das Gesamtsystem aller Bias- und Gewichtsmatrix eines solchen Netzes vollständig. Wir nennen dieses System $\mathcal{W}$. Es handelt sich dabei um eine dreidimensionale Matrix, die entsprechend dreifach indizierbar ist über $\mathcal{W}_{\ell,i,j}$. Der Konvention von Goodfellow et. al., [GBC16], folgend,

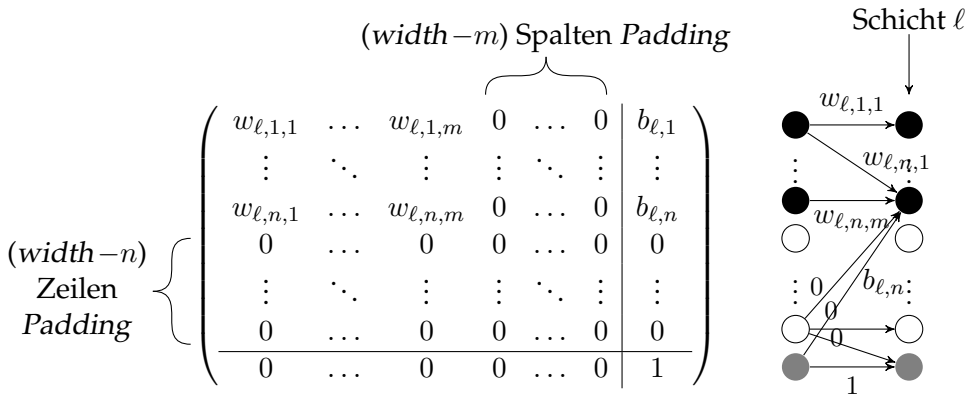


Abbildung 11: Kompakte Darstellung des Gewichts- und Biassystems eines Neuronalen Netzes nebst seiner Verortung im hypothetischen Netzwerk. Der besseren Übersichtlichkeit wegen, sind nicht alle Kanten eingezeichnet.

wählt eine Indizierung „:“ alle entsprechenden Elemente, ein negativer Index $-k$ referenziert das k-te Element beginnend vom größten Index in Richtung der kleineren Indizes und $[r, s]$ wählt für $r \leq s$ den Bereich von r bis s. So gilt beispielsweise $\mathcal{W} = \mathcal{W}_{:,:,:}$,

$$\mathcal{W}_{\ell,:,:} = \begin{pmatrix} w_{\ell,1,1} & \dots & w_{\ell,1,m} & 0 & \dots & 0 & b_{\ell,1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{\ell,n,1} & \dots & w_{\ell,n,m} & 0 & \dots & 0 & b_{\ell,n} \\ 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 & 1 \end{pmatrix}, \qquad \mathcal{W}_{\ell,:,-1} = \begin{pmatrix} b_{\ell,1} \\ \vdots \\ b_{\ell,n} \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix},$$

$$\mathcal{W}_{\ell,[1,n],[1,m]} = \begin{pmatrix} w_{\ell,1,1} & \dots & w_{\ell,1,m} \\ \vdots & \ddots & \vdots \\ w_{\ell,n,1} & \dots & w_{\ell,n,m} \end{pmatrix}$$

und $\mathcal{W}_{\ell,i,j} = w_{\ell,i,j}$, für $1 \leq i \leq n$ und $1 \leq j \leq m$.

3.2 Bezeichnungen der Knoten

Im letzten Abschnitt haben wir für die Neuronen die einfache graphische Repräsentation verwendet: Jedes Neuron wurde durch genau einen Knoten im Neuronalen Netz repräsentiert. Es wird sich aber als nützlich erweisen, eine möglichst detailierte Darstellung Neuronaler Netze zur Verfügung zu haben. Wir konzipieren im Folgenden daher Neuronen wieder, wie in Abbildung 2 veranschaulicht, als Komposition

aus *A-System* und *R-System*, welches wir im letzten Kapitel durch die beiden Knotenfunktionen Pre, Gleichung 4, und Post, Gleichung 3, ausgedrückt hatten.

Die im *Marabou*-System verwendete Knotenbezeichnungskonvention ist wenig subtil. Alle Knoten des Netzes werden beginnend bei der Zahl Null *aufgezählt* und mit dem Wert der Aufzählung identifiziert. Der i-te Knoten *heißt* gemäß *Marabou*-Konvention schlicht i. Als Besonderheit bei der Aufzählung kommt hinzu, dass dabei zunächst die Knoten der Eingabeschicht aufgezählt werden, dann die Knoten der Ausgabeschicht und erst dann die Knoten der *Hidden Layer*. Die Aufzählung letzterer geht dabei wie folgt vor sich. Sei ℓ ein *Hidden Layer* mit n Neuronen. Zunächst werden die n Pre-Knoten der Schicht aufgezählt, dann die n Post-Knoten der Schicht. Sei i eine Zahl, die in der Schicht ℓ als Pre-Knoten aufgezählt wurde. Die Zahl des zum Pre-Knoten namens i zugehörigen Post-Knoten bestimmt sich demnach zu $i + n$.

Eingedenk dieser *Marabou*-spezifischen Eigenheit bezüglich der Benennung der Knoten, wählen wir davon abweichend ein Bezeichnungssystem, welches mit den im vorigen Abschnitt eingeführten Notationen besser zusammenstimmt. Dies möge wie in Abbildung 12 dargestellt geschehen.

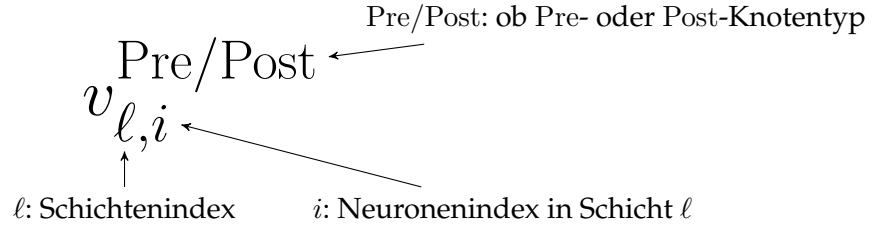


Abbildung 12: Benennungsvereinbarung für die Knoten eines Neuronalen Netzes. Dabei bezeichnet $v_{\ell,i}^{\mathrm{Pre}}$ den Pre-Knoten des i-ten Neurons in Schicht ℓ und $v_{\ell,i}^{\mathrm{Post}}$ den entsprechenden Post-Knoten des i-ten Neurons in der ℓ-ten Schicht.

Demnach lässt sich die Postition eines Knotens $v_{\ell,i}^{\mathrm{Pre/Post}}$ im Netz aus den Indizes bestimmen. Der Index ℓ steht dem gemäß für die Schichtennummer und i für das i-te Neuron in Schicht ℓ. Der Typ des Knotens wird über den hochgestellten Index angegeben: Index Pre steht für einen Knotentyp Pre und Index Post für einen Knoten des Typs Post. Da die Eingabeschicht $\ell = 0$ nicht aus Neuronenknoten besteht, ist die Unterscheidung zwischen Pre- und Post-Knotentyp nicht sinnvoll; wir vereinbaren daher die Bezeichnungskonvention $v_{0,i} :=: v_{0,i}^{\mathrm{Pre}} :=: v_{0,i}^{\mathrm{Post}}$ für alle Knoten der Eingabeschicht. Die Benennung v hat drei mnemonische Anknüpfungspunkte. Im Englischen bezeichnet das Wort *vertex* einen Knoten. Zweitens insinuiert v, dass die Knoten jeweils für eine Variable, englisch *variable*, stehen, denen drittens ein Wert, englisch *value*, zukommt.

Die Gesamtheit, der bis hierhin getroffenen Benennungsvereinbarungen ist zusammenfassend in Abbildung 13 schematisch dargestellt.

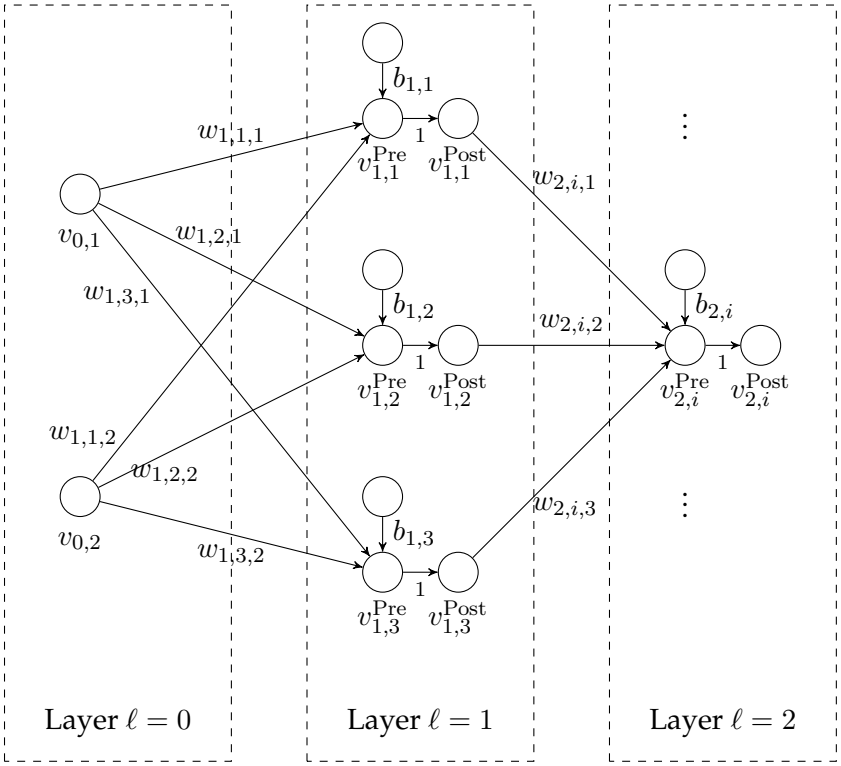


Abbildung 13: Zusammenfassung der getroffenen Benennungsvereinbarungen für Kantengewichte, Bias und Knoten eines Neuronalen Netzes. Abgebildet sind die ersten drei Schichten des Netzes. Layer $\ell = 0$ ist hierbei die Eingabeschicht. In Layer $\ell = 2$ ist nurmehr der i-te Knoten dargestellt.

3.3 Neuronale Netze als Formel

Nachdem wir nun alle relevanten Teile eines Neuronalen Netzwerkes, so wie wir es in der Verifikation verwenden wollen, benannt haben, wollen wir im Folgenden formal beschreiben, wie sich ein gegebenes Neuronales Netz genau verhält. Konkret werden wir ein gegebenes Netz *formalisieren*, was wiederum bedeutet, dass wir die Essenz, das dem Netz Wesentliche, in einer Formel in Prädikatenlogik Erster Stufe ausdrücken werden.

Es gehört zu den Treppenwitzen der jüngeren Wissenschaftsgeschichte, dass ausgerechnet der Aufsatz der Autoren Luca Pulina and Armando Tacchella, [PT10], welcher die bahnbrechende Idee, Neuronale Netze in Prädikatenlogik zu formalisieren – auf die wir den gegenwärtigen Abschnitt stützen –, der wissenschaftlichen Gemeinschaft verfügbar machte, diese Idee als so unwichtig empfunden haben muss, dass sie im erwähnten Aufsatz keinerlei Explikation oder auch nur Erwähnung findet. Die Formalisierungsidee ist stillschweigend lediglich in der Begleitsoftware *Ne-*

Ver inkorporiert. Der im Aufsatz auf *NeVer* verweisende *Hyperlink* ist tot. Wir folgen in diesem Abschnitt der Rekonstruktion der besagten Formalisierungsidee im Sinne Albarghouthis, [Alb21]; insbesondere Kapitel 5 ebendort.

In Kapitel 4 der dieser Arbeit werden wir die Details bezüglich der Formalisierung in Prädikatenlogik Erster Stufe genauer untersuchen und in Kapitel 5 tiefer in die mathematischen Theorien, nicht zuletzt die Theorie der Reellen Zahlen, einsteigen. Für den Moment genügen unsere Kenntnisse der sekundären Schulmathematik. Insbesondere setzen wir gegenwärtig voraus, dass die mathematischen Gegenstände ihre gewohnte Bedeutung haben. Die Zeichenfolge 123 bedeutet demnach die Zahl Einhundertdreiundzwanzig und nicht etwa Schlange-Schwan-Möwe. Wir haben es, so wollen wir voraussetzen, mit keinen anderen Gegenständen, respektive Konstanten, als den Reellen Zahlen zu tun: Der Diskursbereich umfasst die Reellen Zahlen und nur diese. Desweiteren setzen wir voraus, dass mathematische Operatorzeichen auch als solche interpretiert werden; also beispielsweise das Funktionszeichen „$+$“ als die mathematische Addition von Reellen Zahlen. Schließlich mögen auch die mathematischen Relationssymbole ihre aus dem Schulunterricht bekannte Bedeutung haben. Die Relationszeichen „$<$“ und „$=$“ stehen demnach für die „Kleiner-als“- sowie die Identitätsbeziehung zwischen zwei Termen, die zu Reellen Zahlen auswerten.

Die Idee der Formalisierung ist nun diese. Wir behaupten einen, gegebenenfalls sehr langen, formalen Satz, ausgedrückt in der Prädikatenlogik erster Stufe, der exakt konstatiert, welche Knoten es im Netz gibt und welchen Wert jeder Knoten des Netzes jeweils hat. Dass Letzteres abhängig davon ist, welchen Wert andere Knoten des Netzes, insbesondere die Eingabeknoten, haben, berücksichtigt der Behauptungssatz ebenfalls.

Da der Allgemeingültigkeit damit genüge getan ist, wollen wir – *Pars pro Toto* – das Netz aus Abbildung 13 in eine solche Formel überführen. Dabei wollen wir inkrementell vorgehen und die gesuchte Formel nach und nach aufbauen. Zunächst müssen wir überhaupt die *Existenz* aller Knoten des Netzes behaupten:

$$\exists v_{0,1} \exists v_{0,2} \exists v_{1,1}^{\mathrm{Pre}} \exists v_{1,1}^{\mathrm{Post}} \exists v_{1,2}^{\mathrm{Pre}} \exists v_{1,2}^{\mathrm{Post}} \exists v_{1,3}^{\mathrm{Pre}} \exists v_{1,3}^{\mathrm{Post}} \ldots \exists v_{2,i}^{\mathrm{Pre}} \exists v_{2,i}^{\mathrm{Post}} \ldots : \left(F_N \right) \tag{7}$$

Die Kaskade der existentiellen Quantifizierer eröffnet einen *Skopus* oder Geltungsbereich – der nach dem „:“ beginnt und durch die Klammern abgegrenzt ist – über den Ausdruck F_N, „Netzwerkformel“, welcher ein Platzhalter für eine noch zu spezifizierende Formel ist. Wir hatten früher schon bemerkt, dass das v der Knotenbezeichnungen auch als „Variable“ interpretiert werden kann, der verschiedene, variable, Werte zukommen können. Da die Genese, der Ursprung, der Werte der Eingabeknoten aus Sicht des Netzes nicht spezifizierbar ist, bleiben diese Werte variabel. Jemand oder etwas außerhalb des Netzes legt fest, was die Eingabewerte für das Netz sind. Über das Zustandekommen der Werte der Knoten der Eingabeschicht können wir also in unserer Formalisierung keine weiteren Angaben machen. Anders verhält es sich bei den Werten der Pre-Knoten. In Gleichung 4 wurde dargelegt,

wie der Wert eines jeden Pre-Knotens zustande kommt. Für den ersten Pre-Knoten des ersten Hidden Layers können wir demnach die Formel

$$(v_{1,1}^{\mathrm{Pre}} = w_{1,1,1} \cdot v_{0,1} + w_{1,1,2} \cdot v_{0,2} + b_{1,1}) \tag{8}$$

angeben, da der Wert des Knotens $v_{1,1}^{\mathrm{Pre}}$ sich, gemäß Gleichung 4, dadurch ergibt, dass die – im allgemeinen Fall gegebenenfalls: Post- – Knotenwerte der Vorgängerschicht gewichtet mit dem entsprechenden Kantengewicht summiert und anschließend der neuroneneigenen Bias $b_{1,1}$ hinzugefügt wird.

Man beachte bei den Formalisierungen hier und im Folgenden besonders den Unterschied zwischen „:=“ und „=“: Der Ausdruck „LHS := RHS“ bedeutet, dass der LHS-Ausdruck künftig ein, kürzerer, Alias des RHS-Ausdrucks ist (LHS ist ein Name für den RHS-Ausdruck); hingegen bedeutet $\mathrm{lhs} = \mathrm{rhs}$, dass der Term lhs und der Term rhs, im Sinne der mathematischen Identitätsrelation, auf denselben Gegenstand referieren. Ein weiterer Hinweis betrifft den Unterschied zwischen der Bezeichnerklasse v, etwa $v_{1,1}^{\mathrm{Pre}}$ und w, etwa $w_{1,1,1}$. Erstere bezeichnen Variablen, die durch die in Formel 7 spezifizierten Quantoren gebunden sind. Die ws hingegen bezeichnen Gegenstände aus dem Diskursbereich; sie stehen für konkrete konstante Reelle Zahlen, die das jeweilige Kantengewicht im Netz beschreiben; gleichartig auch die Biasfamilie der bs.

Die allgemeine Formalisierung des i-ten Pre-Knotens in Schicht $\ell > 0$ mit n Knoten in der Vorgängerschicht ist nun, analog, zunächst einfach die folgende.

$$(v_{\ell,i}^{\mathrm{Pre}} = w_{\ell,i,1} \cdot v_{\ell-1,1}^{\mathrm{Post}} + \cdots + w_{\ell,i,n} \cdot v_{\ell-1,n}^{\mathrm{Post}} + b_{\ell,i}) \tag{9}$$

Offenkundig – bei allem Respekt gegenüber der am Eingang des Kapitels zitierten Emmy Noether – ist jene Formel äquivalent zu dieser:

$$\begin{aligned} F_{\ell,i}^{\mathrm{Pre}} := (v_{\ell,i}^{\mathrm{Pre}} &\leq w_{\ell,i,1} \cdot v_{\ell-1,1}^{\mathrm{Post}} + \cdots + w_{\ell,i,n} \cdot v_{\ell-1,n}^{\mathrm{Post}} + b_{\ell,i}) \wedge \\ (v_{\ell,i}^{\mathrm{Pre}} &\geq w_{\ell,i,1} \cdot v_{\ell-1,1}^{\mathrm{Post}} + \cdots + w_{\ell,i,n} \cdot v_{\ell-1,n}^{\mathrm{Post}} + b_{\ell,i}) \end{aligned} \tag{10}$$

Die Gesamtheit der Teilformeln, die die Bedeutung aller Pre-Knoten des Netzes N in Abbildung 13 beschreiben, lässt sich in der Formel

$$F_N^{\mathrm{Pre}} := (F_{1,1}^{\mathrm{Pre}} \wedge F_{1,2}^{\mathrm{Pre}} \wedge F_{1,3}^{\mathrm{Pre}} \wedge \cdots \wedge F_{2,i}^{\mathrm{Pre}} \wedge \cdots) \tag{11}$$

festhalten.

Um eine Formel für die Post-Knoten zu finden, betrachten wir die Gleichung 3 in der Interpretation von *ReLU* im Sinne der Gleichung 2. Die direkte Formalisierung in diesem Sinne, exemplarisch für $v_{1,1}^{\mathrm{Post}}$, führt auf die folgende Variante einer Formalisierung, die auch in Albarghouthi, [Alb21], Verwendung findet.

$$\big(\ \big((v_{1,1}^{\mathrm{Pre}} \leq 0) \Rightarrow (v_{1,1}^{\mathrm{Post}} = 0)\big)\ \wedge\ \big((v_{1,1}^{\mathrm{Pre}} > 0) \Rightarrow (v_{1,1}^{\mathrm{Post}} = v_{1,1}^{\mathrm{Pre}})\big)\ \big) \tag{12}$$

Oder, allgemein für einen beliebigen Post-Knoten $v_{\ell,i}^{\mathrm{Post}}$ des Netzes,

$$\big(\ \big((v_{\ell,i}^{\mathrm{Pre}} \leq 0) \Rightarrow (v_{\ell,i}^{\mathrm{Post}} = 0)\big)\ \wedge\ \big((v_{\ell,i}^{\mathrm{Pre}} > 0) \Rightarrow (v_{\ell,i}^{\mathrm{Post}} = v_{\ell,i}^{\mathrm{Pre}})\big)\ \big). \tag{13}$$

Man beachte, dass die Semantik korrekt wiedergegeben ist: Ist der Wert des Pre-Knotens kleiner oder gleich Null, so ist der Wert des zugehörigen Post-Knotens genau Null. Ist, andernfalls, der Pre-Knotenwert größer Null, so ist der Wert des Post-Knotens identisch mit dem Wert des Pre-Knotens. Wir wollen der Formel 13 aber eine andere, wenngleich äquivalente, Gestalt geben. Wir beobachten, dass ganz allgemein für $x, y \in \mathbb{R}$ gilt: $(x \leq y)$ ist äquivalent zu $\neg(x > y)$. Demnach haben wir im Speziellen, dass die Formel $(v_{\ell,i}^{\mathrm{Pre}} \leq 0)$ äquivalent ist zur Formel $\neg(v_{\ell,i}^{\mathrm{Pre}} > 0)$. Legen wir fest, dass $P := (v_{\ell,i}^{\mathrm{Pre}} \leq 0)$, $Q := (v_{\ell,i}^{\mathrm{Post}} = 0)$ und $R := (v_{\ell,i}^{\mathrm{Post}} = v_{\ell,i}^{\mathrm{Pre}})$ bezeichne. Betrachten wir Abbildung 14.

P	Q	R	$(P \Rightarrow Q)$	$\wedge$	$(\neg P \Rightarrow R)$	$(P \wedge Q)$	$\vee$	$(\neg P \wedge R)$	$(P \wedge Q)$	$\dot{\vee}$	$(\neg P \wedge R)$
0	0	0	1	**0**	0	0	**0**	0	0	**0**	0
0	0	1	1	**1**	1	0	**1**	1	0	**1**	1
0	1	0	1	**0**	0	0	**0**	0	0	**0**	0
0	1	1	1	**1**	1	0	**1**	1	0	**1**	1
1	0	0	0	**0**	0	0	**0**	0	0	**0**	0
1	0	1	0	**0**	0	0	**0**	0	0	**0**	0
1	1	0	1	**1**	1	1	**1**	0	1	**1**	0
1	1	1	1	**1**	1	1	**1**	0	1	**1**	0
				I			II			III	

Abbildung 14: Wahrheitstafel, die für $P := (v_{\ell,i}^{\mathrm{Pre}} \leq 0)$, $Q := (v_{\ell,i}^{\mathrm{Post}} = 0)$ und $R := (v_{\ell,i}^{\mathrm{Post}} = v_{\ell,i}^{\mathrm{Pre}})$ belegt, dass die Formalisierung der Post-Knoten ein Logisches Exklusives Oder involviert. Die Werte der Spalten I, II, und III sind jeweils identisch, was damit die Äquivalenz der Formeln im Tabellenkopf zeigt.

Die erste Formelspalte I der abgebildeten Wahrheitstafel berechnet den Wahrheitswert zu allen möglichen Welten der Formel 13. Die zweite Formelspalte II berechnet den Wert zur Formel, welche in Formel 14 dargestellt ist.

$$\big(\ \big((v_{\ell,i}^{\mathrm{Pre}} \leq 0) \wedge (v_{\ell,i}^{\mathrm{Post}} = 0)\big)\ \vee\ \big((v_{\ell,i}^{\mathrm{Pre}} > 0) \wedge (v_{\ell,i}^{\mathrm{Post}} = v_{\ell,i}^{\mathrm{Pre}})\big)\ \big) \tag{14}$$

Aus der jeweiligen Gleichheit der Wahrheitswerte beider Formeln in jeder möglichen Welt – also der Gleichheit der jeweiligen Zeilen in der Tabelle in der entsprechenden Spalte –, ergibt sich die Äquivalenz beider Formeln. Die Formel 13 enthält also implizit ein Logisches Oder, welches in der äquivalenten Formel 14 explizit gemacht wird. Der in der III-Spalte gelistete, wiederum, zur I- und II-Spalte, identische Spaltenwert belegt, dass es sich um ein Exklusives Oder, also eine Antivalenz, ausgedrückt durch den Junktor „$\dot{\vee}$“, handelt.

Betrachten wir, um die Formalisierung der Post-Knoten zu einem Abschluss zu bringen, noch den Sonderfall, dass die Variable, wohlgemerkt, $v_{\ell,i}^{\mathrm{Pre}}$ der bisherigen

Formalisierung in Formel 14 den Wert genau 0 annimmt. Dann haben wir den einzigartigen Fall, dass

$$(v_{\ell,i}^{\mathrm{Pre}} = 0) = (v_{\ell,i}^{\mathrm{Post}} = 0) = (v_{\ell,i}^{\mathrm{Post}} = v_{\ell,i}^{\mathrm{Pre}})$$

und damit insbesondere, im Hinblick auf Formel 14, dass

$$(v_{\ell,i}^{\mathrm{Pre}} \geq 0) \wedge (v_{\ell,i}^{\mathrm{Post}} = v_{\ell,i}^{\mathrm{Pre}}),$$

wobei man die „$\geq$"-Relation im ersten Konjunkt im Unterschied zur „$>$"-Relation ebendort beachte. Die folgende Formel 15 ist also äquivalent zu Formel 14.

$$\big(\ \big((v_{\ell,i}^{\mathrm{Pre}} \leq 0) \wedge (v_{\ell,i}^{\mathrm{Post}} = 0)\big)\ \vee\ \big((v_{\ell,i}^{\mathrm{Pre}} \geq 0) \wedge (v_{\ell,i}^{\mathrm{Post}} = v_{\ell,i}^{\mathrm{Pre}})\big)\ \big) \tag{15}$$

Ersetzen wir in dieser Formel wiederum alle Gleichungen – Teilformeln die das Gleichheitszeichen enthalten – durch die entsprechende, äquivalente, Konjunktion von Ungleichungen so gelangen wir schließlich zu

$$\begin{aligned} F_{\ell,i}^{\mathrm{Post}} := \big(\ &\big((v_{\ell,i}^{\mathrm{Pre}} \leq 0) \wedge (v_{\ell,i}^{\mathrm{Post}} \leq 0) \wedge (v_{\ell,i}^{\mathrm{Post}} \geq 0)\big) \quad \vee \\ &\big((v_{\ell,i}^{\mathrm{Pre}} \geq 0) \wedge (v_{\ell,i}^{\mathrm{Post}} \leq v_{\ell,i}^{\mathrm{Pre}}) \wedge (v_{\ell,i}^{\mathrm{Post}} \geq v_{\ell,i}^{\mathrm{Pre}})\big)\ \big). \end{aligned} \tag{16}$$

Betrachten wir die beiden Teilformeln, aus denen die Disjunktion der Formel 16 besteht, einmal genauer. Das erste Disjunkt,

$$F_{\ell,i}^{\mathrm{ReLU\_inactive}} := \big((v_{\ell,i}^{\mathrm{Pre}} \leq 0) \wedge (v_{\ell,i}^{\mathrm{Post}} \leq 0) \wedge (v_{\ell,i}^{\mathrm{Post}} \geq 0)\big), \tag{17}$$

beschreibt den Fall, in dem das Neuron i in Schicht ℓ inaktiv ist: Der Post-Knoten des Neurons nimmt, *ReLU*-vermittelt, den Wert Null an, da der Wert des Pre-Knotens kleiner oder gleich Null ist. Der andere mögliche Fall, beschrieben durch das zweite Disjunkt, ist, dass die *ReLU* aktiv ist: Das Neuron „feuert". Es nimmt, da dieser größer oder gleich Null ist, den Wert des Pre-Knotens, als Wert für den Post-Knoten, an:

$$F_{\ell,i}^{\mathrm{ReLU\_active}} := \big((v_{\ell,i}^{\mathrm{Pre}} \geq 0) \wedge (v_{\ell,i}^{\mathrm{Post}} \leq v_{\ell,i}^{\mathrm{Pre}}) \wedge (v_{\ell,i}^{\mathrm{Post}} \geq v_{\ell,i}^{\mathrm{Pre}})\big) \tag{18}$$

Wir können also zusammenfassen, dass

$$F_{\ell,i}^{\mathrm{Post}} \ :=\ \big(\ F_{\ell,i}^{\mathrm{ReLU\_inactive}}\ \vee\ F_{\ell,i}^{\mathrm{ReLU\_active}}\ \big). \tag{19}$$

Die Konjunktion aller Teilformeln, welche die Semantik der Post-Knoten des Netzes N angibt, lässt sich, im Hinblick auf Abbildung 13, in der folgenden Formel festhalten.

$$F_N^{\mathrm{Post}} := (F_{1,1}^{\mathrm{Post}} \wedge F_{1,2}^{\mathrm{Post}} \wedge F_{1,3}^{\mathrm{Post}} \wedge \cdots \wedge F_{2,i}^{\mathrm{Post}} \wedge \cdots) \tag{20}$$

Da damit nun alle Knoten des Netzes formal beschrieben sind, ergibt sich die gesuchte Netzwerkformel, F_N, im Skopus der Existenzquantoren von Formel 7, als Konjunktion der Formel 11, zur Beschreibung aller Pre-Knoten, und der Formel 20, zur Beschreibung aller Post-Knoten, als

$$F_N := F_N^{\text{Pre}} \wedge F_N^{\text{Post}}. \tag{21}$$

Damit ist die Formalisierung des Netzwerks abgeschlossen. Für einen Beweis der Korrektheit und Vollständigkeit dieser Formalisierung verweisen wir auf Albarghouthi, [Alb21], der ebendort nachweist, dass die im vorliegenden Kapitel vorgenommene, an Albarghouthi, [Alb21], orientierte, Formalisierung einerseits das tatsächliche Verhalten des Netzes wiedergibt und andererseits jede widerspruchsfreie, „erfüllende", Belegung der Variablen in der Formel F_N der Formalisierung, dem tatsächlichen Verhalten des Netzes entspricht.

Wir wollen dieses Kapitel damit beschließen, dass wir auf einige Besonderheiten der Formalisierung hinweisen.

Zunächst sei betont, dass zur formalen Beschreibung des Netzes, welche in der Formel F_N ihren Ausdruck findet, als einzige Relation das „$\leq$"- respektive „$\geq$"-Prädikat benötigt wurde. Insbesondere kann die Semantik des Netzes ohne Verwendung der Prädikate „$=$", „$<$", und „$>$", beschrieben werden. Man beachte, dass dies nicht-trivial ist, da etwa die „$<$"-Relation nicht – ohne Zuhilfenahme der Logischen Negation – als Kombination von „$\leq$"- und „$\geq$"-Relationen ausgedrückt werden kann.

Ein ähnlicher Sparsamkeitsverweis, was die Repräsentation der F_N-Formel betrifft, ist im Hinblick auf die verwendeten Logischen Junktoren – die Wahrheitsfunktionen – angebracht: Die Formel 21 kann formuliert werden, ohne andere Logische Junktoren, als lediglich das Logische Und, „$\wedge$", und das Logische Oder, „$\vee$", verwenden zu müssen. Bemerkenswerterweise wird insbesondere die Logische Negation, „$\neg$", nicht benötigt.

Eine Frage, die sich des weiteren stellen mag, ist die Frage nach der Länge der Formel, gemessen als Anzahl der Literale – das sind hier diejenigen Teilformeln, in denen ein Prädikat „$\leq$" oder „$\geq$" vorkommt, aber keine Logischen Junktoren –, in Relation zur Anzahl der Neuronen des Netzes. Offenbar müssen für jeden neu hinzugefügten Neuronenknoten $v_{\ell,i}$ zwei Teilformeln wie in Formel 10 beschrieben und sechs Teilformeln wie in Formel 16 angegeben, hinzugefügt werden. Die Größe der Netzwerkformel wächst also linear mit der Größe des Netzes.

Gehen wir einen Schritt weiter und untersuchen, welche Größe die Disjunktive Normalform, kurz *DNF*, der F_N-Formel 21 besitzt. Die *DNF*, ist dabei eine Disjunktion von Konjunktionen von Literalen. Offenkundig ist die Teilformel 11, F_N^{Pre}, *DNF*-konform, da sie nur aus den Konjunkten der in Formel 10 beschriebenen Art besteht. Nicht *DNF*-konform jedoch sind die Teilformeln 19, in der Formel 20, F_N^{Post}, da diese jeweils nicht ausschließlich aus Konjunkten bestehen. Das „innere" Oder der Teilformeln 19 muss jeweils über den Rest der Teilformeln, mittels des Distributivgesetzes der Disjunktion, distribuiert werden, um die Formel in eine, äußere, Disjunktion von, inneren, Konjunktionen zu überführen.

Um das Resultat dieser Distribution besser nachvollziehen zu können, betrachten wir ein Netz – bei beliebiger Größe $n > 0$ der Eingabeschicht – mit nur einem

Neuronen. Die zur Formel

$$F_{1,1}^{\text{Pre}} \wedge (F_{1,1}^{\text{ReLU\_inactive}} \vee F_{1,1}^{\text{ReLU\_active}})$$

gehörige *DNF* ist offenkundig

$$\begin{aligned} F_1^{\text{DNF}} := \quad & (F_{1,1}^{\text{Pre}} \wedge F_{1,1}^{\text{ReLU\_inactive}}) \quad \vee \\ & (F_{1,1}^{\text{Pre}} \wedge F_{1,1}^{\text{ReLU\_active}}). \end{aligned}$$

Man beachte, dass die Formel $F_{1,1}^{\text{ReLU\_inactive}}$ beziehungsweise Formel $F_{1,1}^{\text{ReLU\_active}}$, als Instanz von Formel 17 beziehungsweise Formel 18, jeweils reine Konjunktionen sind. Fügen wir einen weiteren Knoten in der selben Schicht hinzu, so ergibt sich aus

$$\begin{aligned} & \big((F_{1,1}^{\text{Pre}} \wedge F_{1,1}^{\text{ReLU\_inactive}}) \vee (F_{1,1}^{\text{Pre}} \wedge F_{1,1}^{\text{ReLU\_active}})\big) \wedge \\ & \big(F_{1,2}^{\text{Pre}} \wedge (F_{1,2}^{\text{ReLU\_inactive}} \vee F_{1,2}^{\text{ReLU\_active}})\big) \end{aligned}$$

die *DNF*-Formel als

$$\begin{aligned} F_2^{\text{DNF}} := \quad & (F_{1,1}^{\text{Pre}} \wedge F_{1,2}^{\text{Pre}} \wedge F_{1,1}^{\text{ReLU\_inactive}} \wedge F_{1,2}^{\text{ReLU\_inactive}}) \quad \vee \\ & (F_{1,1}^{\text{Pre}} \wedge F_{1,2}^{\text{Pre}} \wedge F_{1,1}^{\text{ReLU\_inactive}} \wedge F_{1,2}^{\text{ReLU\_active}}) \quad \vee \\ & (F_{1,1}^{\text{Pre}} \wedge F_{1,2}^{\text{Pre}} \wedge F_{1,1}^{\text{ReLU\_active}} \wedge F_{1,2}^{\text{ReLU\_inactive}}) \quad \vee \\ & (F_{1,1}^{\text{Pre}} \wedge F_{1,2}^{\text{Pre}} \wedge F_{1,1}^{\text{ReLU\_active}} \wedge F_{1,2}^{\text{ReLU\_active}}). \end{aligned}$$

Der Fall, dass der weitere Knoten nicht in der selben, sondern in der nachfolgenden Schicht eingefügt wird, ist analog, nur dass sich die Indizes vertauschen: Aus $F_{1,2}^{\text{Pre}}$ wird $F_{2,1}^{\text{Pre}}$ und so weiter. Wie wir sehen, hat sich mit dem Hinzufügen des Knotens die Anzahl der Disjunkte, welche die *DNF*-Formel beschreiben, verdoppelt. Setzen wir die Hinzufügung weiterer Knoten gedanklich fort, so erkennen wir, dass die jeweils resultierende Disjunktive Normalform bei jeder weiteren Hinzufügung eines Knotens eine Verdoppelung der Anzahl der Disjunkte bewirkt. Allgemein können wir dies wie folgt angeben.

Satz 2. *Sei N ein Neuronales Netz mit einer nicht näher spezifizierten Anzahl Eingabeknoten, welche jedoch größer als Null sei, sowie n Neuronenknoten welche in n* Pre*-Knoten, $v_1^{\text{Pre}}, \ldots, v_n^{\text{Pre}}$, und n* Post*-Knoten $v_1^{\text{Post}}, \ldots, v_n^{\text{Post}}$ untergliedern. Sei F_i^{Pre}, für $i \in \{1, \ldots, n\}$, wie in Formel 10 und $F_N^{\text{Pre}} := (F_1^{\text{Pre}} \wedge \ldots \wedge F_n^{\text{Pre}})$ wie in Formel 11. Sei, für alle $i \in \{1, \ldots, n\}$, als Kurzschreibweise, $F_i^0 := F_i^{\text{ReLU\_inactive}}$ und $F_i^1 := F_i^{\text{ReLU\_active}}$, wobei $F_i^{\text{ReLU\_inactive}}$ und $F_i^{\text{ReLU\_active}}$ wie in Formel 17 und Formel 18. Sei F_N die Netzwerkformel gemäß Formel 21. Dann besteht die Disjunktive Normalform der Formel F_N*

$$\begin{aligned} F_N^{\text{DNF}} := & (F_N^{\text{Pre}} \wedge F_1^0 \wedge \ldots \wedge F_{n-1}^0 \wedge F_n^0) \quad \vee \\ & (F_N^{\text{Pre}} \wedge F_1^0 \wedge \ldots \wedge F_{n-1}^0 \wedge F_n^1) \quad \vee \\ & \vdots \\ & (F_N^{\text{Pre}} \wedge F_1^1 \wedge \ldots \wedge F_{n-1}^1 \wedge F_n^0) \quad \vee \\ & (F_N^{\text{Pre}} \wedge F_1^1 \wedge \ldots \wedge F_{n-1}^1 \wedge F_n^1) \end{aligned} \tag{22}$$

aus 2^n Disjunktionsgliedern.

Beweis. Die Anzahl der Disjunkte lässt sich durch folgende Überlegung direkt auflisten. Wir definieren die Formeln $F^{d_{n-1}\ldots d_1 d_0} := (F_N^{\mathrm{Pre}} \wedge F_1^{d_{n-1}} \wedge \ldots F_{n-1}^{d_1} \wedge F_n^{d_0})$, wobei die d_i, mit $i \in \{0, \ldots, n-1\}$, Elemente der Menge $\{0, 1\}$ sind. Die Konkatenation der Binärziffern $d_{n-1} \ldots d_1 d_0$ ergibt eine n-stellige Binärzahl. Zu jeder der 2^n verschiedenen n-stelligen Binärzahlen b gibt es in F_N^{DNF} das Disjunkt F_n^b. Auf eine andere Weise gibt es keine Disjunkte in F_N^{DNF}. □

Der Satz hält fest, dass die Länge der Formel 21 in der zu ihr äquivalenten *DNF*-Darstellung als Formel 22 asymptotisch exponentiell in der Anzahl der Neuronen wächst.

4 Prädikatenlogische Grundlagen der Formalisierung

> Zur Erreichung dieser letzten und tiefsten Zurückführung bedarf es nun freilich einer Reihe von Hilfsmitteln und Tatsachenzusammenhängen, die zunächst bereitgestellt und erklärt werden müssen.
>
> Moses Schönfinkel,
> *Über die Bausteine der mathematischen Logik* [Sch24]

> Logic is an old subject, and since 1879 it has been a great one.
>
> Willard Van Orman Quine,
> *Methods of Logic* [Qui50]

> [S]ie kauften [. . .] Bücher über Logik. „Wir müssen lernen, wie man richtig denkt", sagten sie. Als ob die Logik irgend etwas damit zu tun hätte.
>
> Paul Feyerabend,
> *Zeitverschwendung* [Fey97]

> I shall suppose for simplicity that the thinker with whom we are concerned uses a systematic language without irregularities and with an exact logical notation like that of *Principia Mathematica*.
>
> Frank Plumpton Ramsey,
> *Facts and Propositions* [Ram27]

Das gegenwärtige Kapitel und das folgende Kapitel 5 wird die im letzten Kapitel 3 gewonnene Formalisierung Neuronaler Netze auf eine theoretische Grundlage stellen. Dazu wollen wir zunächst das grundlegende System, auf dem die Formalisierung fußt – die Prädikatenlogik Erster Stufe – allgemein einführen. Im nächsten Kapitel 5 werden wir die Prädikatenlogik Erster Stufe als allgemeinen Fall auf den Spezialfall der Linearen Arithmetik einschränken.

4.1 Behauptungssätze als Gegenstandsbereich der Logik

Erfolgreiche Sprachen bedürfen wohl stets zweier wesentlicher Dinge. Einerseits eines Zeichensystems – das ist, eine eindeutige Lautfolge, reproduzierbare Gesten- oder Schriftfolgen oder was dergleichen mehr als Zeichen dienen mag –, welches nach bestimmten Regeln Zeichen zu Zeichenfolgen zusammenzustellen erlaubt. Dieses System wird *Syntax* – nach dem Griechischen σύνταξις [sýntaxis], aus σύν [sýn], „zusammen", „zugleich", und ταssω [tassō], „aufstellen", „an seinen Platz stellen" demnach „Zusammenstellung", „Anordnung"; hier insbesondere der Zei-

chen – der entsprechenden Sprache genannt. Andererseits, der sogenannten *Semantik*, nach dem Griechischen σημαντικός [sēmantikós], „bezeichnend“, welches vom Verb σημαίνω [sēmáinō] „bezeichnen“, „mit einem Zeichen oder Siegel versehen“, kommt. Dadurch – durch das Be-zeichnen – werden die Zeichen mit dem Bezeichneten verknüpft und es entsteht eine Beziehung zwischen den Zeichen und der Welt; wie wenn, um eine Fremdsprache, etwa Englisch, zu lernen, zuweilen Alltagsgegenstände, wie Stuhl, Tisch, Kühlschrank, mit Haftzetteln, auf denen die jeweils passende Vokabel notiert ist, beklebt werden: „*chair*“, „*table*“, „*fridge*“. Die Semantik ist demnach ein Bedeutungssystem, welches festlegt, was die Zeichen und die Zeichenfolgen des Zeichensystems bedeuten.

Das Zusammenspiel beider Systeme – die Sprache – ermöglicht, in der Terminologie von Wittgensteins „Tractatus“, [Wit22], „Sachverhalte“ der Welt zu behaupten. Sachverhalte sind dabei nicht notwendigerweise nur, *eo ipso* wahre, Tatsachen, also *bestehende* Sachverhalte. Auch Aussagen über die Welt, die gar nicht stimmen – ein nicht bestehender Sachverhalt wird behauptet –, können getätigt werden. Es lässt sich demnach Wahres und Falsches behaupten. Diejenige Person, welche die Zeichenfolge syntaxkonform verfasst, *behauptet* damit das, wofür die Zeichenfolge gemäß ihrer Semantik steht. Die Einheiten der Behauptung sind die Behauptungs- oder, was das gleiche ist, Aussage*sätze*, welche – gemäß Freges „Der Gedanke“, [Fre19], worauf wir diese Sprechweise hier stützen – einen Gedanken, eben das Behauptete, englisch *Proposition*, ausdrücken. Diese Sätze sind, *qua* Behauptungssätze, der Gegenstandsbereich der Aussagenlogik, englisch *Propositional Logic*. Andere Sätze – etwa Fragesätze, die nur im Verbund mit einer Antwort auf die gestellte Frage eine Behauptung darstellen, oder Imperativ- oder Wunschsätze, welche gar keine Behauptung aufstellen – gehören nicht zum Kern der klassischen Logik und wir werden sie hier nicht weiter untersuchen, weshalb im Folgenden immer Behauptungssätze gemeint sind, wenn von Sätzen allgemein die Rede ist.

Der durch einen Satz ausgedrückte Gedanke, der „Sinn des Satzes“, [Fre19], seine Proposition, ist dasjenige was konstant bleibt, wenn man den Satz korrekt in eine andere Sprache übersetzt: Der Gedanke, „dass Gras grün ist“ und „that grass is green“ ist offenkundig derselbe. Dieser subtile Unterschied zwischen einem Satz und dem durch ihn ausgedrückten Gedanken ist für die vorliegende Arbeit immateriell und wir werden künftig freimütig unterschiedslos von einem Satz P und dem durch ihn ausgedrückten Gedanken reden, indem „der Gedanke (, dass) P“ stets als „der Satz, der den Gedanken P ausdrückt“ verstanden werden will.

Behauptungssätze, so hatten wir gesehen, können, ja müssen, wahr oder falsch sein. Alternativ lässt sich das auch so ausdrücken: Der Wert eines Satzes ist stets entweder wahr oder falsch. Um einen Ausdruck für diese beiden Werte zu haben, verwenden wir die sogenannten *Booleschen Konstanten*, wobei der Wert der Wahrheit durch das Zeichen *true* oder 1 und der Wert der Falschheit durch *false* oder 0 dargestellt wird. Die Boolesche Menge $\mathbb{B}$ definieren wir entsprechend als $\mathbb{B} := \{0, 1\}$, respektive $\mathbb{B} := \{\mathrm{false}, \mathrm{true}\}$.

Bisher haben wir nicht näher spezifiziert, wie und auf welche Weise Behaup-

tungssätze aufgebaut sind. Insbesondere haben wir übergangen, worum es in Behauptungssätzen strukturell zu gehen pflegt, nämlich um Gegenstände, ihre Eigenschaften und ihre Beziehungen untereinander. Einfache oder „atomare“ Sätze – wir werden später sehen, dass es auch aus atomaren Sätzen zusammengesetzte Sätze gibt –, zum Mindesten diejenigen, welcher Art uns im gegenwärtigen Kontext der Prädikatenlogik interessieren, lassen sich zergliedern in Gegenstände einerseits und etwas, das sogenannte *Prädikat*, was über diese Gegenstände ausgesagt wird andererseits. Einer prominenten Ausnahme von dieser Regel, nämlich Prädikate ohne Gegenstände, werden wir später ebenfalls begegnen. Wir wollen die Begriffe „Gegenstand“ und „Prädikat“ im Folgenden ausführlich vorstellen.

4.2 Gegenstände und Terme

Wir fassen den Begriff „Gegenstand“ hier und im Folgenden stets so weit, dass damit prinzipiell alles gemeint sein kann, worüber man Aussagen machen kann – ausgenommen der Prädikate selbst, hier ist die Stelle des Aufstiegs von der Prädikatenlogik *Erster*, welche dies nicht erlaubt, zu Prädikatenlogiken höherer Stufen, welche dies durchaus erlauben –, sodass insbesondere auch Personen Gegenstände sind; wir unterscheiden folglich auch nicht zwischen Subjekten und Objekten. Objekte und Gegenstände sind uns für unsere Untersuchung im Folgenden stets dasselbe. Mit Objekten sind, ausdrücklich, Gegenstände *der Welt* gemeint, welche sich von ihren Bezeichnern unterscheiden. William Shakespeare ist ein Gegenstand, eine Person, welcher bezeichnet werden kann durch den Ausdruck: Der, 1616 gestorbene, Autor von „Romeo und Julia“, Teilhaber des Londoner *Globe Theatre* und so weiter. Davon zu unterscheiden ist das *Zeichen*, die Buchstabenfolge, „Shakespeare“, die kein (oder zumindest nicht derselbe wie das Bezeichnete) Gegenstand ist. Mit Letzterem können wir – im Rahmen einer Semantik, worauf wir später zu sprechen kommen werden – auf Ersteren referenzieren; aber nur jener, Shakespeare die Person, ist ein Gegenstand.

Um den Diskurs – vom lateinischen *discurrere*, das Hin und Herlaufen, übertragen auf die Gedanken, also der Gedankenaustausch, die Unterhaltung, die Untersuchung – zu erleichtern, wird in der Regel eine Einschränkung über die Menge aller Gegenstände der Welt getroffen, die vom Kontext abhängt. Diese Einschränkung wird durch die Angabe eines sogenannten *Diskursbereichs* – zuweilen auch, nicht ganz passend, da dieses im gängigen Verständnis, das Welt-All umfasst, *Universum* genannt – vorgenommen, welcher eindeutig festlegt, von welchen Gegenständen, im gegenwärtigen Kontext, prinzipiell Aussagen gemacht werden dürfen. Bei der Formalisierung einer Theaterkritik einer klassischen Inszenierung von „King Lear“ dürfte der Gegenstand Riemann-Integral also ebensowenig zum Diskursbereich gehören, wie der Gegenstand Beifahrerairbag in den zur Formalisierung der Peano-Axiome nötigen Diskursbereich.

Um überhaupt über Gegenstände, nicht zuletzt im Rahmen der Prädikatenlogik, reden zu können, muss eine Möglichkeit gefunden werden, um auf die in Frage

stehenden Gegenstände Bezug nehmen – oder sie *referenzieren*, von lateinisch *referre*, „zurückbringen“, „wiederherstellen“ – zu können, selbst wenn diese gerade nicht präsent sind. Die Verwendung von Zeichen erfüllt diesen Zweck: Sie bezeichnen – und stehen insofern stellvertretend für – den Gegenstand. Konkret werden die Zeichen im Kontext der Prädikatenlogik *Symbole* genannt, welches vom Griechischen σύν [sýn], hier in der Bedeutungsnuance „zusammentreffend“, „in Übereinstimmung mit“ und βαλλειν [ballein], „werfen“, „treffen“, als σύμβολον [sýmbolon], neben „Übereinkunft“ auch „Kennzeichen“, „Merkmal“ bedeutet. Symbole sind Kennzeichen, oder Bezeichner, durch Übereinkunft, nämlich der Festlegung der Semantik, und ermöglichen, unter anderem, erst die Referenz.

Diese im Kontext der Prädikatenlogik zur Untersuchung stehende Referenz, der Gegenstandsbezug, kann auf zwei Weisen geschehen: direkt oder indirekt. Die direkte Referenz geschieht über den Namen des Objekts: „Die (Zahl) Zwei“, „Marie-Henri Beyle“, „John Dunbar“. Die indirekte Referenz, auf die, in dieser Reihenfolge, selben Gegenstände, nimmt einen Umweg über eine Beschreibung, welche andere Gegenstände involviert: Die Beschreibung „Der Nachfolger der Zahl Eins.“ nimmt auch, direkt, auf die Zahl Eins Bezug. Die künftige Art der Notation vorwegnehmend, kann dies auch kurz als $\mathrm{nachfolgerVon}(1)$ notiert werden. Die Referenz „Der Schriftsteller (lies hier: „Gegenstand“), der sich nach der größten Hansestadt in der Altmark im heutigen Sachsen-Anhalt benannt hat.“ involviert die, wiederum indirekte, Beschreibung der Stadt Stendal. Sie kann so dargestellt werden: $\mathrm{hatSichBenanntNach}(\mathrm{groessteHansestadtIn}(\mathrm{altmarkInSachsenAnhalt}))$. Die Bezeichnung „Der mit dem Wolf tanzt.“ ist ein Grenzfall, den man sowohl indirekt oder direkt verstehen kann: Indirekt als „Der Gegenstand, der hervorgehoben ist, als derjenige, welcher mit dem spezifischen, von Dunbar „Socke“ genannten, Wolf-Gegenstand tanzt.“, kurz $\mathrm{tanztMit}(\mathrm{socke})$, oder, direkt, als der auf den Namen „Der-Mit-Dem-Wolf-Tanzt“ getaufte Gegenstand, kurz $\mathrm{derMitDemWolfTanzt}$, der alternativ direkt mit dem Namen $\mathrm{johnDunbar}$ bezeichnet wird.

Wir können festhalten, dass auf Gegenstände in der Prädikatenlogik Erster Stufe, stets über eine Beschreibung Bezug genommen wird – sofern wir den Namen eines Gegenstandes als diejenige, triviale, „Beschreibung“ verstehen, die keine anderen Gegenstände involviert. Beschreibungen firmieren in der Literatur – etwa in, Bradley und Mannas Buch, „The Calculus of Computation“, [BM07] – unvorteilhafterweise schlicht unter *Functions*, „Funktionen“; so auch in Schönings „Logic for Computer Scientists“, [Sch01]. Kleene unterscheidet in seinem Buch „Mathematical Logic“, [Kle67], immerhin die gegenstandsreferenzierenden „*Functions*“ *simpliciter* noch von den „*Propositional Functions*“, das sind die noch zu besprechenden Prädikate, und den „*Truth Functions*“, den ebenfalls noch zu besprechenden Logischen Junktoren. Der Literatur folgend, werden wir Gegenstandsbeschreibungen im angegebenen Sinne, ebenfalls lediglich als „Funktionen“ bezeichnen. Tatsächlich sind Beschreibungen Funktionen oder, genauer, eine Familie von Funktionen, und zwar derart, dass sie Gegenstände auf Gegenstände abbilden.

Dabei ist die jeweilige Anzahl der Parameter einer Funktion, durch ihre *Stellig-*

keit, englisch *arity*, festgelegt. Die Stelligkeit σ – das griechische kleine Binnensigma, dessen Schluss-Sigma-Äquivalent ς an das lateinische s (und das kyrillische c) erinnert, steht uns mnemonisch für „Stelligkeit“ – bildet dabei jedes Funktionssymbol f auf die Anzahl der von f erwarteten Argumente ab. Funktionen mit $n > 0$ Stellen – das sind die indirekten Beschreibungen der Gegenstände – werden, sofern der spezifizierte Diskursbereich keine naheliegenderen Symbole vorgibt, als f, g, h und so weiter, symbolisiert. Nullstellige Funktionen – das heißt, die jeweiligen Namen und damit die direkten Bezeichnungen von Gegenständen – werden durch Symbole der Art a, b und so weiter, also Kleinbuchstaben an früher Stelle im Alphabet, dargestellt werden, wenn wir anders keinen spezifischen Diskursbereich spezifiziert haben, welcher gegebenenfalls andere Symbole nahelegt.

Da nullstellige Funktionen einen konstanten Wert besitzen, also hier konkret immer den selben Gegenstand bezeichnen müssen, werden die Namen der Gegenstände alternativ auch als *(Individuen-)Konstanten* bezeichnet; wobei „Individuen“ sich an dieser Stelle auf Objekte, im Unterschied zu Prädikaten, bezieht. Dass es auch *(Individuen-)Variablen* gibt, gehört ebenfalls zu den Grundtatsachen der Prädikatenlogik. Diese, mit u, v, w, x, y, z bezeichneten und gegebenenfalls mit Indizes, etwa $v_{\ell,i}^{\mathrm{Pre}}$, versehenen, Symbole dienen als Platzhalter für Gegenstände des Diskursbereichs und können zunächst einmal jeden Wert aus dieser Menge annehmen. Wir gehen stets davon aus, dass die Kardinalität der Menge der Variablen (abzählbar) unendlich ist; dass dafür die Anzahl der Symbole hinreicht, sichern nötigenfalls indizierte Variablensymbole mit Indizes aus der Menge der Natürlichen Zahlen.

Im Rahmen der Syntax werden die verwendeten Symbole durch die Menge Σ – einer Art Glossar, oder Wörter-(genauer: Symbol-)Verzeichnis, der Sprache – spezifiziert, wobei Σ, das *Sigma*, als der griechische Großbuchstabe, der dem lateinischen „S“ entspricht, mnemonisch für „Symbole“ (oder „Signatur“) steht. Die Menge der Funktionssymbole wird hierbei im Speziellen durch die Menge Σ_{func} angegeben; die Menge der Variablensymbole durch Σ_{V}. Wir setzen fest, dass die Mengen Σ_{func} und Σ_{V} disjunkt sind: Die Symbole für Funktionen sind von den Variablensymbolen stets unterscheidbar.

Wir hatten oben bereits angedeutet, dass die Argumente von Funktionen wiederum Funktionen involvieren können, dass, anders ausgedrückt, Bezeichner von Gegenständen, auch rekursiv, andere Bezeichner von Gegenständen beinhalten können. Formal können wir dies nun konkretisieren, indem wir den Begriff *Term* einführen. Er ist induktiv wie folgt definiert.

Definition 2. *Seien Σ_{func} eine Menge von Funktionssymbolen, mit σ als zugehöriger Stelligkeitsfunktion, sowie Σ_V eine Menge von Individuenvariablen. Seien die Symbole $a \in \Sigma_{\mathrm{func}}$, mit $\sigma(a) = 0$, eine Individuenkonstante, $f \in \Sigma_{\mathrm{func}}$, mit $n := \sigma(f) > 0$, eine mehrstellige Funktion und $x \in \Sigma_V$ eine Individuenvariable, dann gilt das Folgende. Jede Konstante a ist ein Term. Jede Variable x ist ein Term. Wenn $t_1, \ldots, t_n$ Terme sind, so ist $f(t_1, \ldots, t_n)$ ein Term. Nichts sonst ist ein Term.*

Damit ist die vollständige Syntax spezifiziert, um in der Prädikatenlogik Erster Stufe auf Objekte Bezug nehmen, sie bezeichnen oder beschreiben, zu können. Wie

man anhand einer solchen, rein syntaktischen, Beschreibung herausbekommt, auf welchen Gegenstand durch den vorliegenden Term konkret referenziert wird, spezifiziert die Semantik der Terme, der wir uns nun zuwenden.

4.3 Semantik der Terme

Die Semantik, als Lehre von der Bedeutung, erklärt die Beziehung zwischen Zeichen und Bezeichnetem. Das Bezeichnete sind die Objekte der Welt, oder konkreter, wie wir bereits festgelegt hatten, die Objekte des Diskursbereichs $\mathbb{D}$, welcher eine Teilmenge der Gegenstände der Welt ist. Wenn wir einem kleinen Kind, welches über den aktiven Gebrauch unserer Sprache noch nicht disponiert, bei der Arbeit zuschauen, so bedeutet, das Kind – indem es mit dem Finger oder der Hand darauf, oder zumindest in die entsprechende Richtung, zeigt – verschiedene Gegenstände. Personen, welche mit der jeweiligen Sprache hinreichend vertraut sind, liefern die den Gegenständen jeweils entsprechenden Namen als Laut-Zeichen: „Zwillingsschwester Anna“ (um einen kürzeren Namen zu haben: „a“), „Papa Ben“ („b“), „Onkel Clemens“ („c“), „Oma Dora“ („d“) und so weiter; wobei der Diskursbereich hier mit $\mathbb{D} :=$ „anwesende Personen am Kaffeetisch bei Oma Doras sechzigstem Geburtstag“ spezifiziert sein könnte. (Dieses Beispiel ist etwas unrealistisch, da Kinder meist Begriffe, nicht Namen, einfordern: „Teller“, „Gabel“, „Kuchen“, „Gebiss“, und so weiter. Begriffe setzen aber Prädikate voraus, welche noch ihrer Einführung harren.) Formal betrachtet, wird eine Funktion – als eine Art Wörterbuch „Welt–Symbol“ – für das Kind spezifiziert, wodurch den Gegenständen der Welt die zugehörigen Zeichen zugeordnet werden.

In der Klassischen Logik wird, im Gegenteil, das umgekehrte Wörterbuch, nämlich das Wörterbuch „Symbol–Welt“, angelegt. Die Symbole werden ausgedeutet oder interpretiert, indem sie den Gegenständen der Welt zugewiesen werden. Die dies vermittelnde Funktion wird entsprechend *Interpretation*, I, genannt; bezüglich der gegenstandsbeschreibenden Funktionen insbesondere kurz I_{func}. Diese Zuweisung kann so konzipiert werden, dass (auf der „Symbol“-Seite des Wörterbuchs) für jedes Funktionssymbol f (auf der „Welt“-Seite) eine $(n+1)$-Tupelmenge, wobei $n := \sigma(f)$, angegeben wird, die eine Abbildung von $\mathbb{D}^n \to \mathbb{D}$ beschreibt. Hierbei stehen, per Konvention, die jeweils ersten n Tupelelemente in dieser Reihenfolge für die konkreten Parameterwerte und das letzte, also $(n+1)$-te, Tupelelement steht für den Wert der Funktion, gegeben diese konkreten Parameterwerte. Man beachte, dass diese Konzeption auch den Spezialfall für nullstellige Funktionen a abdeckt, die auf eine einelementige Eintupelmenge, wobei das jeweils einzige Element, (e), aus $\mathbb{D}^1$ stammt, abgebildet werden; der Einfachheit halber werden wir in diesen Fällen den Wert der Interpretation mit e, und nicht mit $\{(e)\}$ angeben. Wir können das formal wie folgt konkretisieren.

Definition 3. *Sei $\mathbb{D}$ ein Diskursbereich und Σ_{func} eine Menge von Funktionssymbolen, mit σ als zugehöriger Stelligkeitsfunktion, sowie Σ_V eine Menge von Individuenvariablen Seien die Symbole $a \in \Sigma_{\text{func}}$, mit $\sigma(a) = 0$, eine Konstante, $f \in \Sigma_{\text{func}}$, mit $n := \sigma(f) > 0$, eine*

mehrstellige Funktion und $x \in \Sigma_V$ eine Variable. Dann ist eine Interpretationsfunktion I_{func} wie folgt definiert.

$$I_{\text{func}}(a) : \Sigma_{\text{func}} \to \mathbb{D}$$
$$I_{\text{func}}(f) : \Sigma_{\text{func}} \to (\mathbb{D}^n \to \mathbb{D})$$
$$I_{\text{func}}(x) : \Sigma_V \to \mathbb{D}$$

Insbesondere ist die Interpretationsfunktion I_{func} mit Bezug auf Terme t – wobei t_1 bis t_n wiederum Terme und a, f und x wie eben sind – so definiert:

$$I_{\text{func}}(t) := \begin{cases} I_{\text{func}}(a), & \text{falls } t = a \\ I_{\text{func}}(f)(I_{\text{func}}(t_1), \ldots, I_{\text{func}}(t_n)), & \text{falls } t = f(t_1, \ldots, t_n) \\ I_{\text{func}}(x), & \text{falls } t = x \end{cases}$$

Man beachte bei dieser Definition im Speziellen den Ausdruck

$$I_{\text{func}}(f)(I_{\text{func}}(t_1), \ldots I_{\text{func}}(t_n)).$$

Hierbei wertet $I_{\text{func}}(f)$ zu einer, möglicherweise anonymen, Funktion $\lambda : \mathbb{D}^n \to \mathbb{D}$ aus, die sofort angewendet, *appliziert*, wird auf die n Argumente von λ, deren Werte diejenigen Gegenstände sind, zu denen die Funktionen $I_{\text{func}}(t_1)$ bis $I_{\text{func}}(t_n)$ – gegebenenfalls rekursiv, wobei $I_{\text{func}}(a)$ und $I_{\text{func}}(x)$ die Rekursionsanker darstellen – jeweils auswerten. Für alle Terme t ist also der Wert von $I_{\text{func}}(t)$ stets ein Element des Diskursbereichs $\mathbb{D}$, also ein Gegenstand.

Wir haben definiert, dass auch Variablen Terme und damit Bestandteile von Termen sein können. Es ist aber noch unklar, wie wir Terme interpretieren – auswerten – sollen, welche Variablen enthalten. In der Tat sind solche, eine Variable x enthaltenden, Terme nur auswertbar, wenn aus dem Kontext heraus klar ist, was die Bedeutung von x ist, das heißt, auf welchen Gegenstand des Diskursbereichs das Variablensymbol x abbildet. Diesen Kontext, so werden wir später sehen, liefern etwa die sogenannten Quantoren. Für den Moment genügt es uns, dass Variablen, sofern ihre jeweilige Bedeutung fixiert wurde, zu Gegenständen des Diskursbereichs auswerten. Aus der Definition der Syntax und Semantik für Terme folgt damit, dass auch Terme den Wert eines Objekts haben. Wir können das so zusammenfassen: Die Bedeutung eines Terms ist sein – der durch ihn beschriebene – Gegenstand.

4.4 Beispiel zur Syntax und Semantik der Terme

Geben wir der obigen Kaffeerunde bei Oma Dora eine Interpretation. Dazu wählen wir hier den Diskursbereich mit

$$\mathbb{D} := \{\text{Anna}, \text{Ben}, \text{Clemens}, \text{Dora}, \text{Ella}, \text{Fritz}\},$$

wohlgemerkt alles Objekte der Welt, wobei die Namen der Gegenstände Ella und Fritz, der Kürze wegen, in dieser Reihenfolge die Symbole „e“ und „f“ sein mögen. Als Funktionssymbole verwenden wir die Menge

$$\begin{aligned}\Sigma_{\text{func}} := \{\ & \mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}, \mathrm{f}, \\ & \mathrm{zwillingsSchwesterVon}, \\ & \mathrm{vaterVon}, \\ & \mathrm{ehemannVon}\ \}\end{aligned}$$

mit entsprechender Stelligkeit

$$\begin{aligned}\sigma = \{\ & (\mathrm{a}, 0), (\mathrm{b}, 0), (\mathrm{c}, 0), (\mathrm{d}, 0), (\mathrm{e}, 0), (\mathrm{f}, 0), \\ & (\mathrm{zwillingsSchwesterVon}, 1), \\ & (\mathrm{vaterVon}, 1), \\ & (\mathrm{ehemannVon}, 1)\ \}\end{aligned}$$

und interpretieren:

$$\begin{aligned}I_{\text{func}} := \{\ & \mathrm{a} \mapsto \mathrm{Anna}, \mathrm{b} \mapsto \mathrm{Ben}, \mathrm{c} \mapsto \mathrm{Clemens}, \mathrm{d} \mapsto \mathrm{Dora}, \mathrm{e} \mapsto \mathrm{Ella}, \mathrm{f} \mapsto \mathrm{Fritz}, \\ & \mathrm{zwillingsSchwesterVon} \mapsto \{(\mathrm{Anna}, \mathrm{Ella}), (\mathrm{Ella}, \mathrm{Anna})\}, \\ & \mathrm{vaterVon} \mapsto \{(\mathrm{Anna}, \mathrm{Ben}), (\mathrm{Ella}, \mathrm{Ben}), (\mathrm{Ben}, \mathrm{Fritz}), (\mathrm{Clemens}, \mathrm{Fritz})\}, \\ & \mathrm{ehemannVon} \mapsto \{(\mathrm{Dora}, \mathrm{Fritz})\}\ \}\end{aligned}$$

Die Zwillingsschwester von Anna, und damit das bedeutende Kind, ist dieser Interpretation zufolge Ella, da die Funktion $\mathrm{zwillingsSchwesterVon}(\mathrm{e})$, weil e den Wert Ella hat, zu Anna auswertet. Wir können in dieser Interpretation, mittels der Termsyntax und der Semantik, die durch I_{func} festgelegt ist, nun auch komplexere Gegenstandsbezüge herstellen, etwa: der Vatersvater der Zwillingsschwester von Ella, *ergo* der Opa väterlicherseits von Anna und Ella, kann so angegeben werden:

$$\underbrace{\mathrm{vaterVon}(\mathrm{vaterVon}(\mathrm{zwillingsSchwesterVon}(\overbrace{\mathrm{e}}^{I_{\text{func}}(\mathrm{e})=\mathrm{Ella}})))}_{I_{\text{func}}(\mathrm{vaterVon}(\mathrm{Ben}))=\mathrm{Fritz}}$$

$$\overbrace{\overbrace{\phantom{\mathrm{vaterVon}(\mathrm{vaterVon}(\mathrm{zwillingsSchwesterVon}(\mathrm{e})))}}^{I_{\text{func}}(\mathrm{zwillingsSchwesterVon}(\mathrm{Ella}))=\mathrm{Anna}}}^{I_{\text{func}}(\mathrm{vaterVon}(\mathrm{Anna}))=\mathrm{Ben}}$$

und wertet zum Gegenstand Fritz aus.

Nicht zuletzt die Probleme bei der Formalisierung der Umgangssprache in eine exakte Sprache wie die der Logik aufzeigend, sei betont, dass die Prädikatenlogik keine Mehrdeutigkeiten zulässt. So könnte man in obigem Beispiel geneigt sein, die „Funktion“ $\mathrm{onkelVon}$ dem Sprachumfang hinzuzufügen. Damit wäre die Formalisierung allerdings nicht in jedem Fall eindeutig. Es könnte, neben Onkel Clemens noch einen weiteren Onkel Gustav geben, dann müsste der Term $\mathrm{onkelVon}(\mathrm{e})$ zwei

Werte gleichzeitig annehmen und wäre keine Funktion. Die selben Überlegungen sprechen gegen die Einführung einer „Funktion“ omaVon im Hinblick auf die Existenz der Mutter der Mutter von Anna und Ella.

Wenngleich Termsysteme, insbesondere Termersetzungssysteme, englisch *Term Rewriting Systems*, kurz *TRS*, mächtige, genauer: turing-mächtige oder, was dasselbe ist, turingvollständige, Systeme sind – wovon die Kombinatorische Logik des, mittlerweile nahezu in Vergessenheit geratenen, Moses Schönfinkel, der das, nach diesem benannte, „Currying“ des, weniger vergessenen, Haskell Curry, Namenspate einer berühmten Programmiersprache, vorwegnahm, ein Exempel gibt –, in der Prädikatenlogik können wir nicht bei den Gegenständen verharren, wenn wir letztendlich doch auf Sätze zielen. Für einen Behauptungssatz fehlt uns noch das, *was* von den Gegenständen aussagt wird, das sogenannte Prädikat.

4.5 Aussagesätze als Nullstellige Prädikate

Wir hatten bereits angesprochen, dass in einem einfachen, atomaren, Behauptungssatz eine Aussage über einen oder mehrere Gegenstände gemacht wird. Im Lateinischen wird das jeweils über – einen oder mehrere – Gegenstände Ausgesagte als *praedicatum*, deutsch „das Zugesprochene“, „das Ausgesagte“, oder, als Fremdwort, „das Prädikat“, bezeichnet. Betrachten wir nun aber folgenden Satz. „Es regnet.“ Wir könnten nun geneigt sein, wie folgt zu analysieren. Von einem Gegenstand, „es“, wird etwas ausgesagt, nämlich, dass *er* regnet. Wir gehen davon aus, dass der gegenwärtige Diskursbereich $\mathbb{D}$ so weit wie möglich gewählt wurde, und somit alle Objekte umfasst, die es gibt. Nun sind wir in der peinlichen Situation, aus $\mathbb{D}$ einen geeigneten Gegenstand a auswählen zu müssen, auf den das Prädikat „regnet“ zutreffen könnte. Es wird uns nicht gelingen.

Wir sehen uns daher genötigt, unsere Untersuchung der Prädikate, mit der Feststellung beginnen zu müssen, dass es Prädikate gibt, die zwar, in der Tat, eine, wahre oder falsche, Aussage machen, aber nicht von einem Gegenstand. Das entsprechende Prädikat, das keiner Gegenstände bedarf und somit – in einem noch zu spezifizierendem Sinne – *nullstellig* ist, ist dann der ganze Satz. Solche Sätze können, dies ist eine syntaktische Regel, als Zeichen repräsentiert werden, indem sie jeweils durch einen Großbuchstaben der Art P – wie, wahlweise, (nullstelliges) *Predicate* oder *Proposition* –, Q, R und so weiter dargestellt werden. Diese Zeichen werden, wenn sie, wie eben erläutert, für einen ganzen Aussagesatz stehen, *Aussagenvariablen* genannt. Sie stehen für *atomare* Sätze, das sind Sätze, die aus genau einem – wohlgemerkt, null- oder mehrstelligen, wobei im letzeren Fall, die Gegenstände über die das jeweilige Prädikat eine Aussage macht, in Form von Termen, ebenfalls gegeben sein müssen – Prädikat bestehen. Diejenige Teilmenge der Prädikatenlogik Erster Stufe, die sich ausschließlich mit nullstelligen Prädikaten, und damit insbesondere objekt- und also auch termfreien Aussagesätzen, befasst, wird *Aussagenlogik*, englisch *Propositional Logic*, genannt. Sie ist ein Spezialfall der generellen Prädikatenlogik Erster Stufe.

Wir wollen an dieser Stelle kurz bei der Aussagenlogik pausieren, weil wir hier zum ersten Mal in der Lage sind, Gedanken, in Form von atomaren nullstelligen Prädikaten-Sätzen der Art P, Q und so weiter ausdrücken zu können. Gedanken, also das, was durch die Sätze ausgesagt wird, lassen sich zueinander in Relation setzen, und drücken dann einen neuen, einen *zusammengesetzten* Gedanken aus. Diese Relation wird, syntaktisch, durch Logische Operatoren ausgedrückt, den sogenannten *Logischen Junktoren*, von denen wir hier die Elemente der Menge $\{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, \dot{\vee}\}$ verwenden. Die Bedeutung der Junktoren, ihre Semantik, ist fix und insbesondere nicht vom Kontext oder dem Diskursbereich abhängig. Der Wahrheitswert des zusammengesetzten Gedankens ergibt sich, abhängig von der jeweiligen Relation, aus den Wahrheitswerten der Gedanken, aus denen er zusammengesetzt ist. Die Logischen Junktoren sind demnach Wahrheitsfunktionen, welche wahrheitswertige Argumente, nämlich Sätze, auf Wahrheitswerte, und zwar einen neuartigen, zusammengesetzten, Satz, abbilden; sie definieren einen nicht unwesentlichen Teil der Semantik der Prädikatenlogik und den wesentlichen Teil der Aussagenlogik, indem sie spezifizieren, wie die Wahrheitswerte von Sätzen – das heißt, wie wir sehen werden, deren Bedeutung – auf den Wahrheitswert des, mittels Logischer Junktoren aus diesen Sätzen zusammengesetzten, neuen Satzes abgebildet werden. Neben den Zeichen P, Q, R und so weiter, welche zur Mehrung des Zeichenvorrats auch mit Indizes versehen sein dürfen, welche *nur für atomare* Sätze stehen, werden die, potentiell mit Indizes versehenen, Zeichen F, G, H und so weiter für *zusammengesetzte oder atomare* Sätze gewählt. Dabei steht F mnemonisch für „Formel“, englisch *formula*, denn in der Sprache der Prädikatenlogik werden Behauptungssätze, egal ob atomar oder zusammengesetzt, in der Regel als Formeln bezeichnet. Wir werden dieser Konvention folgen und gleichbedeutend auch Formel statt Satz sagen.

Betrachten wir nun, in folgender induktiver Definition, was, im Rahmen der Syntax, der Lehre von der korrekten Zusammenstellung der Zeichen, gültige Formeln – sogenannte *wohlgeformte Formeln*, auf Englisch *well-formed formulas*, kurz WFF – sind.

Definition 4. *Jede atomare Formel ist eine wohlgeformte Formel. Wenn F und G wohlgeformte Formeln sind, so sind auch $\neg F$, $(F \wedge G)$, $(F \vee G)$, $(F \Rightarrow G)$, $(F \Leftrightarrow G)$, $(F \dot{\vee} G)$ wohlgeformte zusammengesetzte Formeln, bei denen die Formeln F und, sofern präsent, G echte Teilformeln sind. Jede wohlgeformte Formel ist sich selbst eine (nicht-echte) Teilformel.*

Es gibt darüberhinausgehend weitere wohlgeformte Formeln, wie sich später herausstellen wird. Um einige Klammern in Formeln zu sparen, wollen wir Festlegen, dass die äußersten Klammern einer zusammengesetzten Formel nicht-rekursiv, weggelassen werden dürfen aber nicht müssen. Insbesondere wollen wir keine potentiell klammernanzahlreduzierenden Operatorpräzedenzen einführen. Wir gehen künftig stets, sofern nicht ausdrücklich gegenteilig angemerkt, davon aus, dass F, G und so weiter für wohlgeformte Formeln stehen und lassen entsprechend die Qualifizierung „wohlgeformt“ weg.

Um aufzuklären, was es mit der Semantik der Logischen Junktoren auf sich hat, erinnern wir uns daran, welchem Zweck sie dienen sollen. Sie sollen, so hatten wir angedeutet, Gedanken verknüpfen, um einen neuen Gedanken ausdrücken zu können. So kann, konkret, die *Negation* eines Gedankes G ausgedrückt werden, indem behauptet wird, dass G *nicht der Fall* ist, wozu das „$\neg$"-Zeichen dient: $F := \neg G$. Dabei ist F, der neue Gedanke, wahr, wenn G, der negierte Gedanke, falsch ist, andernfalls aber ist die Behauptung F falsch, da der durch G ausgedrückte Gedanke, anders als behauptet, eben doch der Fall, also wahr, ist. Die *Konjunktion* F zweier Gedanken G und H, den beiden *Konjunkten*, behauptet, dass G *und zugleich* H, in Zeichen: $F := (G \wedge H)$, wahr ist. Ist eines, oder sind beide der Konjunkte falsch, so, und nur so, ist auch F falsch. Ist F hingegen eine *Disjunktion* der beiden Sätze G und H, den *Disjunkten*, so ist die durch die Zeichenfolge $F := (G \vee H)$ repräsentierte Behauptung dann und nur dann wahr, wenn G *oder* H wahr ist, wobei das Oder beinhaltet, inkludiert, dass *auch beide* zusammen wahr sein dürfen, weshalb man hierbei auch vom „*inklusiven Oder*" spricht.

Eine *Materielle Implikation*, auch *Konditional-* oder Bedingungssatz genannt, liegt vor, wenn der zusammengesetzte Satz F, aus einem Vordersatz G, dem sogenannten *Antezedenz*, und einem, *Konsequenz* genannten, Nachsatz H besteht, syntaktisch $F := (G \Rightarrow H)$, der dann und nur dann falsch ist, wenn G wahr ist aber H falsch. Hierbei impliziert – vom lateinischen *implicatio*, „die Verflechtung", „das Enthaltensein" –, enthält, die Wahrheit des Vordersatzes bereits die Wahrheit des Nachsatzes, weshalb diese Satzkonstruktion zuweilen auch „Wenn-Dann-Satz" genannt wird: Wenn G (wahr ist), dann (ist auch) H (wahr). Man beachte aber die, gegebenenfalls unintuitive, Folge aus dieser Definition, nach der der gesamte Satz $G \Rightarrow H$ bereits dann wahr ist, wenn der Vordersatz falsch ist. Eine Implikation in „beide Richtungen gleichzeitig", wird durch die *Materielle Äquivalenz*, lateinisch *aequi*, „gleich" und *valere*, „wert sein", beschrieben. Demnach beschreibt $F := (G \Leftrightarrow H)$ den Gedanken, dass G wahr ist, *genau dann, wenn* H wahr ist. Der Satz F ist also nur wahr, wenn G und H *beide zugleich wahr oder beide zugleich falsch* sind, andernfalls ist F falsch. Insbesondere kann der Satz $G \Leftrightarrow H$ falsch sein, etwa wenn G falsch aber H wahr ist.

Um einen Ausdruck dafür zu haben, dass der Satz $G \Leftrightarrow H$ unbedingt wahr sein muss, gibt es den Ausdruck der *Logischen Äquivalenz*, als Zeichen „$\equiv$", mithin $G \equiv H$. Auch wenn wir die Einführung der *Logischen Implikation*, für welche das Zeichen „$\models$", ein zweistelliger Infix-Operator ($M \models F$), steht, erst in Kapitel 5 vornehmen, so wollen wir bereits an dieser Stelle festhalten, dass $G \equiv H$, welches eine Abkürzung ist für $\models (G \Leftrightarrow H)$, ausdrückt, dass $(G \Leftrightarrow H)$, die rechte Seite des $\models$-Infixoperators, bedingungslos gilt, was dadurch ausgedrückt ist, dass die linke, die Bedingungs-, Seite des Operators leer ist. Man kann das auch so ausdrücken: $(G \equiv H)$ ist eine *Tautologie* – vom Griechischen τὰ αὐτά [tà autá], „auf dieselbe Weise" und λόγος [lógos], „Gesagtes" –, ein Satz der immer wahr ist. Das Obige zusammenfassend, lässt sich die Materielle Äquivalenz in Form dieser Logischen Äquivalenz ausdrücken: $(P \Leftrightarrow Q) \equiv ((P \Rightarrow Q) \wedge (Q \Rightarrow P))$.

Vom „Gegenteil“ des Bikonditionals spricht man bei der *Antivalenz* oder Kontravalenz, lateinisch „entgegengesetzte Wertigkeit“, $F := \neg(G \Leftrightarrow H)$ beziehungsweise, unter Einführung eines neuen Operators, $F := (G \dot{\vee} H)$. Man beachte den Punkt über dem „$\dot{\vee}$“, welcher den Operator von der inklusiven Disjunktion, ausgedrückt durch den $\vee$-Operator, unterscheidet. Der Antivalenz zufolge ist F genau dann wahr, wenn *entweder* G wahr ist *oder* H, weshalb diese Funktion auch exklusive Disjunktion oder *xor*-Funktion genannt wird, vom englischen *Exclusive* („*X*“) *Or*.

Ein Atom P oder ein negiertes Atom, $\neg P$, und nur solche, bezeichenen wir als *Literal*.

Die *Semantik der Aussagenlogik* – wohlgemerkt noch nicht der Semantik der Prädikatenlogik Erster Stufe im Allgemeinen, auf die wir im Folgenden zu sprechen kommen werden, deren Semantik deutlich voraussetzungsreicher ist – lässt sich so veranschaulichen. Wenn wir Aussagen über die Welt mittels Aussagenlogik formalisieren möchten, formalisieren wir die darzustellenden Sachverhalte mittels der Aussagenvariablen P, Q, R und so fort, und verknüpfen sie mittels der Logischen Junktoren in passender Weise. Stellen wir uns beispielsweise vor, wir erhalten Kenntnis von folgender Aussage einer weisen Person: „Wenn Planck Protonen proferiert, dann quadriert sich die Quantenqualität und wenn Planck Protonen nicht proferiert, dann rotieren die Radiomere reaktiv.“ Wir formalisieren den Satz, als Formel F, so: $F := ((P \Rightarrow Q) \wedge (\neg P \Rightarrow R))$, wobei P: „Planck proferiert Protonen“, Q: „Die Quantenqualität quadriert sich.“ und R: „Die Radiomere rotieren reaktiv.“ repräsentiert. Es ist nicht ausgeschlossen, dass wir uns von der, durch Satz F zum Ausdruck gebrachten, Weisheit derart beeindruckt zeigen, dass wir herausfinden möchten, wie die Welt beschaffen sein muss, dass der Satz F wahr ist. Wir werden dann Konstellationen der Welt folgender Art durchgehen: Angenommen, P ist wahr, Q ist wahr aber R ist falsch. Wir untersuchen hier folglich Sachverhalte, also Möglichkeiten, wie die Welt konstituiert sein könnte.

Tatsächlich lässt sich die Semantik der Aussagenlogik durch die Spezifikation von *möglichen Welten* veranschaulichen. Hierbei ist eine mögliche Welt in Bezug auf einen Satz F dadurch charakterisiert, dass die in F vorkommenden und atomare Sätze repräsentierenden Aussagenvariablen P, Q, R und so weiter – die als Variablen zunächst einmal ja nur Platzhalter für Wahrheitswerte sind – mit Booleschen Konstanten, also jeweils einem Wert aus der Menge $\{\mathrm{false}, \mathrm{true}\}$, respektive $\{0, 1\}$, belegt werden. Diese vollständige Wertbelegung, auch Zuweisung genannt, wird als *Interpretation* bezeichnet. Die Bedeutung eines atomaren Satzes ergibt sich demnach direkt aus dem ihm im Rahmen der Interpretation zugewiesenen Wahrheitswert. Die Bedeutung des zusammengesetzten Satzes F ergibt sich durch die Auswertung der F konstituierenden Wahrheitsfunktionen, also den in F enthaltenen Logischen Junktoren. Dies ist die gesamte Semantik der Aussagenlogik, die angibt, wie aus, gegenbenenfalls, mit Junktoren und Klammern syntaxkonform verknüpften Aussagenvariablen, also einer Formel F, der (Wahrheits-)Wert des Satzes, für den F steht, ermittelt werden kann.

Die Semantik kann bequem durch die Verwendung von sogenannten *Wahrheitstafeln* vor Augen geführt werden. Dabei gibt es eine Spalte für jede, einen atomaren Satz repräsentierende, Ausagenvariable, deren in Gesamtheit kombinatorisch mögliche Belegungen zeilenweise eingetragen werden. Jede Zeile steht somit für eine alternative mögliche Welt. In weiteren Spalten können zur Untersuchung stehende zusammengesetze Sätze F derart eingetragen werden, dass unter den Junktoren von F – welche, wie wir definiert hatten, jeweils echte Teilformeln von F untereinander verknüpfen –, der Wert, zu dem der jeweilige Junktor in dieser Welt auswertet, geschrieben wird. Die Reihenfolge der Auswertung der Junktoren ergibt sich aus der Klammerung in F. Auf diese Weise kann der Wert, die Bedeutung, von F in der jeweiligen möglichen Welt ermittelt werden. Den Wert unseres obigen Satzes $F := ((P \Rightarrow Q) \wedge (\neg P \Rightarrow R))$ berechnet die in Abbildung 14 dargestellte Wahrheitstafel für jede mögliche Welt. Eine ausführliche Einführung in die Aussagenlogik, die auch die Berechnung der Werte eines zusammengesetzten Satzes mittels Wahrheitstafel detailliert darstellt, leistet Peter Smiths „An Introduction to Formal Logic“, [Smi03] .

4.6 Eigenschaften, Begriffe und Relationen als mehrstellige Prädikate

Nun wollen wir uns denjenigen Sätzen zuwenden, welche tatsächlich Aussagen von Objekten machen. Hierfür ist zunächst zu klären, was mehrstellige Prädikate, was hier und im Folgenden stets heißt, Prädikate mit mehr als null Stellen, konzeptionell, eigentlich sind. Dazu können wir folgende pseudoarithmetische Überlegung anstellen: Prädikate (genauer: zunächst einmal Prädikatssymbole) sind das, was übrig bleibt, wenn man von einem Satz diejenigen Symbolfolgen, die Gegenstände bezeichnen – also die Terme –, abstrahiert. Vom Satz „Bukephalos ist ein Pferd.“ bleibt, nachdem wir „Bukephalos“, der Name des Pferds von Alexander dem Großen, entfernt haben, das Fragment „_ ist ein Pferd.“, wobei „_“ eine Freistelle andeutet. Das Überbleibsel des Satzes „Die Zahl Eins ist kleiner als die Zahl Zwei.“, nachdem die Namen Eins und Zwei entfernt wurden, ist nur noch „_ ist kleiner als _.“ Anstatt der Leerstellen können nun offenbar andere Terme gesetzt werden und wir erhalten wieder einen Satz. „Kleiner Onkel ist ein Pferd.“ ist ein Satz, der den Wert wahr annimmt, da „Kleiner Onkel“ der Name des Pferds der Pippilotta Viktualia Rollgardina Pfefferminz Efraimstochter Langstrumpf ist. Im Satz „Der Nachfolger der Zahl Eins ist kleiner als die Zahl Zwei.“ wurde die erste Freistelle des Prädikats mit dem Term „Nachfolger der Zahl Eins“, belegt und die zweite wiederum mit dem Namen „Zwei“, was zur Falschheit dieses Satzes führt.

Fassen wir das Ergebnis unserer Überlegung zusammen. Prädikate sind in der Hinsicht unvollständig, dass sie Freistellen haben. Werden die Freistellen mit Gegenstandsbeschreibungen, das heißt, Termen, belegt, so erhalten wir einen Satz, der wahr oder falsch ist. Demnnach sind Prädikate n-stellige Funktionen, „Abbildungen“, um sie nicht mit denjenigen Funktionen zu verwechseln, welche im Kontext von Termen eine Rolle zu spielen pflegen – wobei n die jeweilige Anzahl der Frei-

stellen angibt –, die von, durch Terme beschriebenenen, Gegenständen auf Sätze, oder Gedanken, Propositionen, abbilden. Prädikate werden deshalb auch als „*Propositional Functions*“ bezeichnet, etwa von Kleene, [Kle67]. Da der Wert eines Satz stets wahr oder falsch ist, lässt sich genauer sagen, dass Prädikate Abbildungen von Gegenständen auf Wahrheitswerte sind.

Für die Syntax der Prädikatenlogik Erster Stufe ist der bisherige Zeichenvorrat – die Menge Σ_{func} der Funktionssymbole, sowie die Menge Σ_V der Symbole für Objektvariablen – zu erweitern um die Menge Σ_{pred}, der Symbole, mit denen die Prädikate bezeichnet werden, was insbesondere auch die nullstelligen Prädikate einschließt. Prädikate werden, wie bereits erwähnt, mit P, Q, R und so weiter bezeichnet, wenn der Diskursbereich keine treffenderen Bezeichner suggeriert. Wir gehen wiederum davon aus, dass Σ_{pred} disjunkt zu Σ_{func} und Σ_V ist, also bei jedem Symbol eindeutig ist, zu welcher der drei Mengen es gehört. Damit kann nun auch die Stelligkeitsfunktion σ dahingehend ergänzt werden, dass sie zusätzlich zu den Funktionssymbolen auch die Prädikatensymbole auf die Anzahl der von ihnen jeweils erwarteten Argumente abbildet.

Prädikate lassen sich gemäß ihrer Stelligkeit klassifizieren. Die nullstelligen Prädikate hatten wir bereits als besonders herausgestellt, da jedes nullstellige Prädikat unmittelbar, ohne eines Terms zu bedürfen, ein *Satz* ist. Die einstelligen Prädikate P, welche genau einen Gegenstand a erwarten, drücken *Eigenschaften* des jeweiligen Gegenstandes aus. Der Satz $P(a)$ behauptet also, dass (das Objekt) a (die Eigenschaft) P hat oder, was dasselbe sagt, dass (die Eigenschaft) P (dem Gegenstand) a zukommt oder auch dass P auf a zutrifft. Die einstelligen Prädikaten lassen sich auch als *Begriffe* verstehen. Unter den Begriff Pferd fällt der Gegenstand Bukephalos. Anders ausgedrückt, was möglicherweise vertrauter klingen mag, Bukephalos ist ein Pferd, wobei „ist ein“ die sogenannte „Kopula“ darstellt, die das Objektsymbol, oder genauer: die Termsymbole, an das Begriffsymbol „koppelt“. Schließlich lassen sich alle n-stelligen Prädikate $P(t_1, \ldots, t_n)$, wobei $n > 1$ ist und t_1 bis t_n Terme sind, als *Relationen* R verstehen. Dazu nehmen wir anstelle der Relationen R, welche ja Mengen von n-Tupeln sind, hier stets die jeweils zu ihnen gehörige charakteristische Funktion χ_R in folgendem Sinne. Die charakteristische Funktion χ_R – das χ [chi] ist, mnemonisch, der Anfangsbuchstabe des griechischen Worts χαρακτήρ [charaktēr], wörtlich „das Eingeprägte“, übertragen „die Eigentümlichkeit“, hier also das der Relation Eigentümliche, Charakteristische – einer Relation R erwartet ein Tupel τ als Argument und wertet zu *true* aus, wenn τ ein Element von R ist, andernfalls wertet es zu falsch aus. Das Prädikat P bezeichnet die Relation R dann und nur dann, wenn die Aussage $P(t_1, \ldots, t_n)$ wahr ist genau dann, wenn $\chi_R(t_1, \ldots, t_n) = \text{true}$, also das n-Tupel $(t_1, \ldots, t_n)$ ein Element von R ist. Aufgrund dieser äquivalenten Beziehung zwischen n-stelligen Prädikaten P und der Charakteristischen Funktion über n-Tupelmengen, welche Relationen R darstellen, lässt sich abkürzend einfach nur von Relationen reden, wenn ausführlich das wie eben ausgeführte Prädikat gemeint ist.

4.7 Quantoren

Wie sich in unserer Untersuchung bis hierher gezeigt hat, ist es in der Sprache der Prädikatenlogik möglich, Gegenständen Prädikate, etwa Eigenschaften also *Qualitäten*, zuzusprechen. Es ist darüber hinaus zusätzlich möglich, Aussagen über *Quantitäten* zu tätigen. So kann etwa ausgedrückt werden, dass *mindestens ein* Gewinner des *Turing Awards* ein Däne ist – was gleichbedeutend ist mit der Behauptung, dass ein solcher dänischer Preisempfänger *existiert* – oder dass *alle* Natürlichen Zahlen einen Nachfolger haben. Dafür werden als weitere Logische Zeichen die Zeichen „$\exists$“ und „$\forall$“, die sogenannten *Quantoren* eingeführt, von denen Ersterer *Existenzquantor*, da er eine Existenzbehauptung einleitet, und Letzterer *Allquantor*, da er eine universelle Behauptung initiiert, genannt wird. Jeder Quantor führt eine neue Variable $x \in \Sigma_V$ ein, die ihm syntaktisch folgt und eröffnet einen *Skopus*, einen Geltungsbereich, der, sofern er sich nicht anders aus der Klammerung ergibt, sich stets über den gesamten Formelteil erstreckt, welcher dem Quantor folgt. Einem variablenbegleiteten Quantoren oder einer Folge von jeweils variablenbegleiteten Quantoren kann ein Doppelpunkt folgen, der die Quantoren vom Rest der Formel abgrenzt. So, wie die Anwendung Logischer Junktoren auf Teilformeln wiederum eine Formel erzeugt, so erzeugt auch die Anwendung eines Quantoren auf eine Formel eine neue Formel. Demnach sind, falls F eine Formel ist, auch $(\forall x\colon F)$ und $(\exists x\colon F)$ Formeln, wobei wir die Klammern gelegentlich auch weglassen.

Um einige Beispiele zu haben, seien P und Q Prädikate sowie F und G, möglicherweise bereits selbst quantifizierte, Formeln. Die Formel $\exists x\colon (P(x) \wedge Q(x))$ kann gelesen werden als „Es existiert ein x so, dass x (die Eigenschaft) P hat und x (die Eigenschaft) Q hat.“ Der Geltungsbereich des Quantors erstreckt sich sowohl auf P als auch auf Q. In der Formel $\forall x\colon (P(x) \Rightarrow Q(x))$, welche als „Für alle x gilt: Wenn x die Eigenschaft P hat, dann hat x die Eigenschaft Q.“ oder, falls P und Q Begriffe sind, kürzer: „Alle Ps sind Qs.“ gelesen werden kann, überdeckt der Skopus wiederum P und Q. Der Geltungsbereich des die Variable x quantifizierenden Existenzquantoren in der Formel $\exists x\colon ((\forall y\colon F) \wedge (\forall z\colon G))$ umfasst die Formeln F und G, der des y-Quantifizierers nur F und der des z-quantifizierenden Allquantors nur G.

Wir begegenen hier nun den, bereits an früherer Stelle vorgestellten, Individuenvariablen wieder, den Elementen der Menge Σ_V. Wir hatten gesagt, dass sie „Platzhalter“ für Gegenstände, Elemente des Diskursbereichs, darstellen. Die, nicht notwendigerweise alle, Freistellen, oder besser: Argumentstellen, der Prädikate können statt mit Gegenstandbeschreibungen, das heißt, Termen, auch mit Platzhaltern für Gegenstände, also, nicht notwendigerweise verschiedenen, Variablen gefüllt werden. Die auf diese Weise eingeführten Variablen heißen *ungebundene Variablen*, solange sie nicht im Geltungsbereich eines Quantoren liegen, der dasselbe Variablensymbol ebenfalls einführt, in welchem Fall sie (durch eben diesen Quantoren) *gebundene Variablen* heißen. So ist in der Formel $\forall x\colon P(a, x, y)$, in der P ein dreistelliges Prädikat, a eine Individuenkonstante und sowohl x als auch y eine Variable ist, x gebunden und y ungebunden. Wir gehen, wie oben bereits angedeutet, stets davon aus, dass jeder Quantor eine *neue* Variable einführt: Wenn in einer Formel

mindestens zwei Quantoren $\mathfrak{Q}_1 x$ und $\mathfrak{Q}_2 y$ vorkommen, $\mathfrak{Q}_1, \mathfrak{Q}_2 \in \{\forall, \exists\}$, welche die Variablen x und y, mit $x, y \in \Sigma_V$, introduzieren, so gilt immer $x \neq y$; eine Variable kann daher niemals von zwei oder mehr verschiedenen Quantoren gleichzeitig gebunden sein.

Wenn Variablen Platzhalter für Gegenstände des Diskursbereichs darstellen, so ist es zweckdienlich, eine notationelle Möglichkeit vorzusehen, die den Austausch eines Platzhalters, einer Variablen, durch etwas, was diesen Platz einnimmt, einen Gegenstand des Diskursbereichs, gestattet. Dazu sehen wir die *Variablensubstitution* vor, die wir wie folgt verstanden wissen wollen. Sei $x \in \Sigma_V$ eine Variable und sei $a \in \mathbb{D}$ ein Gegenstand aus dem Diskursbereich sowie F eine Formel, in der x gar nicht, oder nur ungebunden vorkommt. Sei G die Formel, die entsteht, indem man alle Vorkommen von x in F gleichzeitig durch a ersetzt. Dann schreiben wir dies als: $G = F[x \leftarrow a]$. Das $[x \leftarrow a]$ ist also eine Ersetzungsvorschrift – man ersetze in der adressierten Formel jedes x durch a – mit Bezug auf Formel F.

Die Bedeutung der Quantoren „$\forall$“ und „$\exists$“ liegt nun darin begründet, dass es durch sie möglich wird, Behauptungen über den gesamten Diskursbereich auf einmal zu machen. Wenn $\mathbb{D}$, wie bisher, ein beliebiger Diskursbereich ist und $|\mathbb{D}|$ seine Mächtigkeit bezeichnet, also die Anzahl der Elemente dieser Menge, so lässt sich die Bedeutung des Allquantor in folgender Definition festhalten.

Definition 5. *Sei $\mathbb{D} := \{d_1, \ldots, d_n\}$ ein Diskursbereich mit, nicht notwendigerweise endlicher, Kardinalität $|\mathbb{D}| = n$. Sei F eine Formel. Dann ist die Bedeutung des Allquantors „$\forall$“ mit Bezug auf die durch ihn eingeführte Variable x wie folgt definiert.*

$$(\forall x\colon F) \;\equiv\; \bigwedge_{d \in \mathbb{D}} F[x \leftarrow d] \;:\equiv\; (F[x \leftarrow d_1] \;\wedge\; \cdots \;\wedge\; F[x \leftarrow d_n])$$

Das heißt, (die Wahrheit) *jedes* der Konjunkte auf der rechten Seite dieser Äquivalenz wird behauptet. Oder, anders, die Formel $(\forall x\colon F)$ steht für die Behauptung, dass die gleichzeitige Ersetzung aller Vorkommen von x in der Formel F durch einen – beliebigen – Gegenstand des Diskursbereichs zu einem wahren Satz führt. Ganz ähnlich lautet die Definition für den Existenzquantor „$\exists$“.

Definition 6. *Sei $\mathbb{D} := \{d_1, \ldots, d_n\}$ ein Diskursbereich mit, nicht notwendigerweise endlicher, Kardinalität $|\mathbb{D}| = n$. Sei F eine Formel. Dann ist die Bedeutung des Existenzquantors „$\exists$“ mit Bezug auf die durch ihn eingeführte Variable x wie folgt definiert.*

$$(\exists x\colon F) \;\equiv\; \bigvee_{d \in \mathbb{D}} F[x \leftarrow d] \;:\equiv\; (F[x \leftarrow d_1] \;\vee\; \cdots \;\vee\; F[x \leftarrow d_n])$$

In Definition 6 wird behauptet, dass mindestens eines der Disjunkte auf der rechten Seite der Äquivalenz der Fall ist: Es existiert, so die Aussage, ein Gegenstand $d \in \mathbb{D}$, welcher, alle Platzhalter x jeweils ersetzend, die Formel F zur Wahrheit führt.

Wir wollen, bezüglich obiger Definitionen, folgende Beobachtung anstellen. Der jeweilige Quantorenausdruck, auf der linken Seite der Definition, der Seite des zu

definierenden *Definiendum*s, ist die, lapidar ausgedrückt, redundante Kurzschreibweise für den, möglicherweise unendlich langen, definierenden Ausdruck auf der rechten Seite der Definition, dem *Definiens*, durch welchen der Quantorenausdruck jederzeit ersetzt gedacht werden kann. Wären wir in der Lage, unendlich lange Ausdrücke, der Art wie sie auf den rechten *Definiens*-Seiten der jeweiligen Definition vorkommen, zu formulieren, so wären Quantoren jederzeit erlässlich und ihre Verwendung diente nur der bequemeren Notation.

Es zeigt sich in den Definitionen auch die Bedeutung der „leeren", englisch *vacuous*, Quantifikation, in welcher $\mathbb{D} = \emptyset$. Die Formel $(\forall x\colon F)$ ist dann („*vacuously*") *true*, da das neutrale Element der Konjunktion, $((G \wedge \top) \equiv G)$, das *Verum*, „$\top$", eine *Satzkonstante* mit konstantem Wert *true*, ist (das „$\top$" erinnert an den Anfangsbuchstaben des Wortes „*Tautologie*", welches einen Satz bezeichnet, der immer wahr ist); entsprechend ist die leere Existenzquantifikation, stets falsch, weil das neutrale Element der Disjunktion, $((G \vee \bot) \equiv G)$, das *Falsum*, „$\bot$", die Satzkonstante mit konstantem Wert *false*, ist (das „$\bot$" erinnert an die „auf den Kopf gestellte Tautologie", welches die Negation der Tautologie, $\bot \equiv \neg\top$, ausdrückt und auch „*Kontradiktion*" oder „Widerspruch" heißt, also einen Satz bezeichnet, der immer falsch ist). Die leere Quantifikation ist in der vorliegenden Arbeit von untergeordneter Bedeutung und wir werden es hier bei dem eben Gesagten bewenden lassen.

Mit dem Gesetz von *De Morgan* – wonach, unter anderem, $\neg(G \wedge H) \equiv (\neg G \vee \neg H)$ gilt – können wir eine wichtige Äquivalenz

$$\neg(\forall x\colon F) \equiv (\exists x\colon \neg F)$$

nachvollziehen, da dementsprechend

$$\neg\Big(\bigwedge_{d\in\mathbb{D}} F[x \leftarrow d]\Big) \;\equiv\; \Big(\bigvee_{d\in\mathbb{D}} \neg F[x \leftarrow d]\Big)$$

gilt. Als Korollar haben wir, allgemein, die folgende Beziehung zwischen existenz- und allquantifizierten Sätzen.

$$\begin{aligned} \neg(\forall x\colon \neg F) &\equiv (\exists x\colon F) \\ \neg(\forall x\colon F) &\equiv (\exists x\colon \neg F) \\ (\forall x\colon \neg F) &\equiv \neg(\exists x\colon F) \\ (\forall x\colon F) &\equiv \neg(\exists x\colon \neg F) \end{aligned} \tag{23}$$

Mit der Einführung der Quantoren ist die Sprache der Prädikatenlogik Erster Stufe komplett und wir können das in diesem Kapitel Zusammengetragene zusammenführen.

4.8 Syntax und Semantik der Prädikatenlogik Erster Stufe

Die Syntax bestimmt, welche Folgen von Zeichen wohlgeformte Formeln darstellen. Der Zeichenvorrat der Prädikatenlogik Erster Stufe umfasst die Logischen Zeichen,

das sind die Logischen Junktoren sowie die Quantoren. Alle weiteren Zeichen, Symbole genannt, werden in der Menge $\Sigma = (\Sigma_{\text{func}} \cup \Sigma_{\text{pred}} \cup \Sigma_V)$ vereinigt; die Mengen der rechten Seite der Gleichung sind disjunkt. Die Menge Σ_{func} enthält die Funktionssymbole, Σ_{pred} die Symbole, die für Prädikate stehen. Die Stelligkeit der Elemente aus $\Sigma_{\text{func}} \cup \Sigma_{\text{pred}}$ wird durch die Funktion σ bestimmt. Die Menge Σ_V enthält einen unendlichen Zeichenvorrat an Variablen. Terme lassen sich, wie in Definition 2 angegeben, aus Variablen und Funktionen aufbauen.

Wohlgeformte Formeln der Prädikatenlogik Erster Stufe lassen sich nun wie folgt induktiv definieren.

Definition 7. *Für jedes Prädikatensymbol $P \in \Sigma_{\text{pred}}$ mit Stelligkeit $\sigma(P) = n$, mit $n \geq 0$ und der (für $n = 0$ leeren) Menge von Termen $\{t_1, \ldots, t_n\}$ ist $P(t_1, \ldots, t_n)$ eine* atomare *Formel. Jede atomare Formel ist eine wohlgeformte Formel. Wenn F und G wohlgeformte Formeln sind, so sind auch $\neg F$, $(F \wedge G)$, $(F \vee G)$, $(F \Rightarrow G)$, $(F \Leftrightarrow G)$, $(F \dot{\vee} G)$ wohlgeformte Formeln. Wenn H eine wohlgeformete Formel ist und $x \in \Sigma_V$ eine in H bislang nicht durch einen Quantoren gebundene Variable, so sind auch $(\forall x \colon H)$ und $(\exists x \colon H)$ wohlgeformte Formeln. Nichts sonst sind wohlgeformte Formeln.*

Obige Definition lässt ausdrücklich Formeln, in denen ungebundene Variablen vorkommen, als wohlgeformte Formeln zu. Später wird solchen Formeln, als, in Definiton 19 eingeführten, „Proto-Sätzen“, noch eine wichtige Rolle zukommen. Aufgrund der Schwierigkeit der Interpretation solcher Formeln, gehen wir für das Folgende aber davon aus, dass die von uns untersuchten Formeln, sofern sie nicht echte Teilformeln einer, sie umfassenden, Formel sind, nur Variablen enthalten, die quantorengebunden sind.

Die Semantik stellt einen Bezug zwischen den Zeichen aus Σ und denjenigen Gegenständen der Welt her, die in der Menge $\mathbb{D}$, welche „der Diskursbereich“ heißt, enthalten sind. Die Semantik der Terme ist dabei so, wie in Definition 3 angegeben. Analog wie dort, stellt eine Interpretation, hier die Interpretation – welche mit I_{pred} bezeichnet sein soll – der Prädikatssymbole $P \in \Sigma_{\text{pred}}$ den Bezug zwischen Zeichen und Bezeichnetem her. Nullstellige Prädikatssymbole werden – da sie, wie wir diskutiert hatten, für diejenigen Aussagesätze stehen, bei denen sich das Ausgesagte nicht auf Gegenstände bezieht – direkt als Wahrheitswerte, also einem Element aus der Booleschen Menge $\mathbb{B}$, interpretiert: Dem Satz „Es ist elf Uhr vierzig.“ kann entweder der Wert *true* oder *false* zugesprochen werden, womit der entsprechende Sachverhalt dann vollständig geschildert ist.

Definition 8. *Sei $\mathbb{D}$ ein Diskursbereich und Σ_{func} eine Menge von Funktionssymbolen und Σ_{pred} eine Menge von Prädikatssymbolen, mit σ als zugehöriger Stelligkeitsfunktion. Sei $P_0 \in \Sigma_{\text{pred}}$, mit $\sigma(P_0) = 0$, ein nullstelliges Prädikatssymbol, und sei $P_n \in \Sigma_{\text{pred}}$, mit $n := \sigma(P_n)$, $n > 0$, ein mehrstelliges Prädikatssymbol. Sei I_{func} eine bereits spezifizierte Interpretation der Funktionssymbole über die Mengen Σ_{func} und $\mathbb{D}$. Dann ist eine Inter-*

pretationsfunktion I_{pred} für atomare Formeln wie folgt definiert.

$$I_{\text{pred}}(P_0) : \Sigma_{\text{pred}} \to \mathbb{B}$$
$$I_{\text{pred}}(P_n) : \Sigma_{\text{pred}} \to (\mathbb{D}^n \to \mathbb{B})$$

Insbesondere ist die Interpretationsfunktion I_{pred} mit Bezug auf, nicht notwenigerweise atomare, Formeln F – wobei t_1 bis t_n Terme, G und H Formeln und P_0 und P_n wie eben sind – so definiert:

$$I_{\text{pred}}(F) := \begin{cases} I_{\text{pred}}(P_0), & \text{falls } F = P_0 \\ I_{\text{pred}}(P_n)(I_{\text{func}}(t_1), \ldots, I_{\text{func}}(t_n)), & \text{falls } F = P_n(t_1, \ldots, t_n) \\ \neg I_{\text{pred}}(G), & \text{falls } F = \neg G \\ I_{\text{pred}}(G) \star I_{\text{pred}}(H), & \text{falls } F = (G \star H), \\ & \quad \text{für alle } (\star \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow, \dot{\vee}\}) \\ \bigwedge_{d \in \mathbb{D}} \big(I_{\text{pred}}(G[x \leftarrow d])\big), & \text{falls } F = (\forall x \colon G) \\ \bigvee_{d \in \mathbb{D}} \big(I_{\text{pred}}(G[x \leftarrow d])\big), & \text{falls } F = (\exists x \colon G) \end{cases}$$

Um, im Rahmen der Semantik, den Bezug von den Zeichen zur Welt herstellen zu können, bedarf es zunächst der Spezifikation der Referenzen, der Gegenstandsbezüge. Wie dies geschieht, hatten wir im Rahmen der Darlegung der Termsemantik, in Definition 3, abgehandelt, wonach, vermittels I_{func}, Terme, also gegenstandsbeschreibende Zeichenketten, auf Gegenstände aus $\mathbb{D}$, dem Diskursbereich, abgebildet werden. Die Semantik der Prädikate im Speziellen und die Semantik der Sätze, das heißt, der Formeln, im Besonderen gibt nun an, wie durch die Verknüpfung von Prädikat und Objekt – wohlgemerkt, des tatsächlichen Objekts, nicht etwa der Objektbeschreibung, eines Terms – der *Wert* des, durch die Verknüpfung geschaffenen, Satzes bestimmt wird. Im Rahmen der Semantik der Prädikate, also den atomaren Formeln, muss dafür zuallererst diejenige Abbildung, deren Argumente Gegenstände sind, beschrieben werden, für die die Prädikatszeichen stehen. Die nullstelligen Prädikate P_0 werden wie Aussagenvariablen in der Aussagenlogik behandelt. Ihnen wird im Rahmen der Interpretation, I_{pred}, einfach ein Wahrheitswert, ein Wert der Menge $\mathbb{B}$, zugeordnet: $I_{\text{pred}}(P_0) \colon \Sigma_{\text{pred}} \to \mathbb{B}$.

Für die Interpretation von n-stelligen Prädikaten, mit $n > 0$, muss eine Abbildung, spezifiziert werden, welche die $\mathbb{D}^n$-Argumente, also jeweils n Gegenstände des Diskursbereichs $\mathbb{D}$, auf einen Wahrheitswert abbildet. Im Fall von Eigenschaften oder Begriffen, also einstelligen Prädikaten, kann dies mithilfe der Angabe derjenigen Teilmenge von $\mathbb{D}$ geschehen, für deren Elemente die Eigenschaft gilt, beziehungsweise, welche unter den entsprechenden Begriff fallen. So kann beispielsweise für das Prädikatensymbol prime, um es als die Primzahleigenschaft über dem Diskursbereich $\mathbb{D} = \mathbb{N}$ zu interpretieren, die Menge $R_{\text{prime}} := \{2, 3, 5, 7, 11, \ldots\}$ konstruiert werden und die Interpretation, bei der stets $a \in \mathbb{D}$ gelten muss, konkretisiert werden zu

$$I_{\text{pred}}(\text{prime}(a)) := \begin{cases} \text{true}, & \text{falls } a \in R_{\text{prime}} \\ \text{false} & \text{sonst} \end{cases}.$$

Für $n > 2$ kann die Abbildung mithilfe der Angabe der entsprechenden Relation geschehen; für das Prädikatssymbol „$>$“ bezüglich des Diskursbereichs $\mathbb{D} = \mathbb{N}$ beispielsweise als $R_> := \{(1,0),(2,0),(2,1),(3,0),\ldots\}$, wobei dann die eigentliche Abbildung als

$$I_{\text{pred}}((a > b)) := \begin{cases} \text{true}, & \text{falls (a, b)} \in \text{R}_> \\ \text{false} & \text{sonst} \end{cases}$$

in die Interpretation, bei der $(a, b) \in \mathbb{D}^2$, einginge. Tatsächlich ergeben sich die Gegenstände $a, b \in \mathbb{D}$, als Argumente der Prädikate, erst aus der Termauswertung, wie aus der Zeile „$I_{\text{pred}}(P_n)(\mathbf{I_{func}}(\mathbf{t_1}), \ldots, \mathbf{I_{func}}(\mathbf{t_n})), \text{falls } F = P_n(t_1, \ldots, t_n)$“ der Definition hervorgeht. So analysieren wir, um ein Beispiel zu geben, bei dem „geboxte“ Zeichen, $\boxed{\mathfrak{s}}$, für das Zeichen und ungeboxte, $\mathfrak{s}$, für den Gegenstand stehen, für die Formel $(\boxed{1}\ \boxed{+}\ \boxed{2})\ \boxed{>}\ \boxed{3}$, wobei wir die Präfixnotation $\boxed{>}(\boxed{+}(\boxed{1}, \boxed{2}), \boxed{3})$ wählen, $\boxed{+}$ die Standardinterpretation als Additionsfunktion geben, $\boxed{>}$ als die Standard-„Größer-Als“-Relation der Natürlichen Zahlen verstehen und $\boxed{\text{i}}$ als die jeweils eingeboxten Gegenstände i aus der Menge der Natürlichen Zahlen interpretieren:

$$\overbrace{\underbrace{I_{\text{pred}}(\boxed{>})}_{>:\,\mathbb{N}^2\to\mathbb{B}}\Big(\underbrace{I_{\text{func}}(\boxed{+})}_{+:\,\mathbb{N}^2\to\mathbb{N}}\Big(\underbrace{I_{\text{func}}(\boxed{1})}_{1}, \underbrace{I_{\text{func}}(\boxed{2})}_{2}\Big), \overbrace{I_{\text{func}}(\boxed{3})}^{3}\Big)}^{\text{false}} \qquad \overbrace{\phantom{I_{\text{func}}(\boxed{+})}}^{3}$$

Wie der Definition 8 zu entnehmen ist, ist der Wert einer atomaren Formel unter einer Interpretation stets ein Wahrheitswert aus der Menge $\mathbb{B}$. In allen weiteren Fällen, die $I_{\text{pred}}(F)$ in der Definition unterscheidet, ist F eine Zusammengesetzte Formel, deren Wahrheitswert wahrheitsfunktional, wobei jeweils Logische Junktoren die Wahrheitsfunktionen darstellen, aus den, gegebenenfalls rekursiv ermittelten, Wahrheitswerten der Teilformeln hervorgehen, die in der Definition – auf der linken Seite, vor dem „falls“ – spezifiziert sind. Damit kann nun, gegeben eine Interpretation $I = (I_{\text{func}} \cup I_{\text{pred}})$, zu jeder wohlgeformten Formel F, sofern, wovon wir stets ausgehen, alle Variablen in F im Skopus genau eines Quantoren liegen, also alle in F vorkommenden Variablen gebunden sind, ihre Bedeutung, das heißt, ihr Wahrheitswert, angegeben werden.

In diesem Kapitel haben wir die Prädikatenlogik Erster Stufe als ein System vorgestellt, in welchem Sachverhalte der Welt, durch die Angabe von Sätzen in Form einer Prädikat-Objekt-Struktur formalisiert werden können. Im nächsten Kapitel wollen wir die Expressivität dieser Sprache auf dasjenige Maß reduzieren, welches wir zur Verifikation von Eigenschaften Neuronaler Netze benötigen. Dazu werden wir das Konzept der Theorien vorstellen. Darüber hinaus werden wir die Logik als die Lehre vom gültigen Schließen – was ihrer ursprünglichen Bestimmung näher kommen dürfte als ihre bloße Verwendung als Formale Sprache – untersuchen.

5 Erfüllbarkeit und Theorien

> ‚Satis sunt' inquit ‚mihi pauci, satis est unus, satis est nullus'.
> [„Genug sind", sagt er, „mir wenige, genug ist ein Einziger, auch keiner ist genug".] [Eigene Übersetzung]
>
> Seneca,
> *Epistulae Morales Ad Lucilium, Epistula 7*

> Dank den Fortschritten der mathematischen Logik haben wir im Laufe der letzten Jahrzehnte gelernt, die mathematischen Disziplinen in Gestalt von formalisierten deduktiven Theorien darzustellen.
>
> Alfred Tarski,
> *Über den Begriff der Logischen Folgerung* [Tar36]

> Wenn die Tatsachen nicht mit der Theorie übereinstimmen – umso schlimmer für die Tatsachen!
>
> Pseudo-Zitat, welches Georg Wilhelm Friedrich Hegel zugeschrieben wird, in dessen Schriften jedoch nicht belegt ist;
> unbekannter Autor

Befähigt durch unsere Untersuchungen des letzten Kapitels 4, sehen wir uns in diesem Kapitel ermächtigt, mit der Prädikatenlogik Erster Stufe nicht nur über ein formales Sprachsystem disponieren zu können, sondern dieses zu instrumentalisieren, um gültige Schlussketten aufzustellen. Aus der Wahrheit eines Satzes F lässt sich in einer solchen Kette „schließen", dass ein anderer Satz G wahr sein muss: Wenn F wahr ist, dann ist es unmöglich, dass G falsch ist, denn – dies ist das, in diesem Kapitel zu lüftende, Geheimnis der Gültigkeit des Schließens – die Möglichkeiten des Wahrseins von F sind in den Möglichkeiten des Wahrseins von G bereits mit eingeschlossen. Aus G können wir vielleicht auf ein weiteres Kettenglied schließen und von diesem auf wieder weitere, woraus wir endlich auf einen Satz H schließen. Wir hätten durch die Kette $F, G, \ldots, H$ schließlich einen *Beweis* des Satzes H aus dem Satz F. Der Satz H ist wahr, und muss es sein, weil die Wahrheit jedes Gliedes der Kette – beginnend bei der unterstellten Wahrheit des ersten Satzes F der Kette – die Wahrheit des jeweils folgenden Satzes der Kette erzwingt. Die Prädikatenlogik entfesselt dieses Potential – von wahren Sätzen zu anderen Sätzen zu gelangen, welche dann ebenfalls wahr sein müssen –, indem sie das Vorgehen beschreibt, wie dieses gültige, das sogenannte *Logische, Schließen* einzurichten ist.

Es liegt nun nahe, anzunehmen, dass sich das Schließen leichter – sicherlich aber nicht schwerer – durchführen lässt, wenn anstelle der vollen Sprache der Prädikatenlogik Erster Stufe nur ein beschränkter Teil derselben zur Anwendung kommt. Die Restriktion der Menge der verfügbaren Zeichen – und, wohlgemerkt, der Fixie-

rung der Interpretation einiger dieser Zeichen –, konkret der Zeichen der Menge Σ, also der Funktionssymbole Σ_{func} und der Prädikatensymbole Σ_{pred}, ist eine solche Beschränkung der Sprache der Prädikatenlogik. Die Gesamtheit aller Sätze, die sich aus dieser eingeschränkten Symbolmenge, hinsichtlich ihrer fixierten Interpretation, erschließen lässt, heißt *Theorie*.

In diesem Kapitel werden wir zunächst auf die Erfüllung von Sätzen – ihr „Wahrmachen" – eingehen, um anschließend den Begriff der *Erfüllbarkeit* Logischer Formeln einzuführen, welcher sich der Möglichkeiten des Wahrseins von Sätzen widmet. Dies wird uns auf den Begriff des Logischen Schließens führen. Abschließend widmen wir uns den, im oben dargelegten Sinne eingeführten, Theorien.

5.1 Modelle und Erfüllbarkeit

Wir wollen unsere Untersuchung in diesem Kapitel dort fortsetzen, wo wir im letzten Kapitel geendigt hatten. Wir hatten dort gesagt, dass die Logischen Zeichen der Menge $\{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, \dot{\vee}, \forall, \exists\}$ eine fixe Bedeutung haben. Alle anderen Zeichen werden über die Symbolmenge, die Signatur, $\Sigma = (\Sigma_{\text{func}} \cup \Sigma_{\text{pred}})$, als Vereinigung der Menge der Funktionssymbole und der Menge der Prädikatensymbole, eingeführt, welche von der Stelligkeitsfunktion σ begleitet wird. Mittels des Diskursbereichs $\mathbb{D}$ wird diejenige Teilmenge von Gegenständen angegeben, von denen in den zur Untersuchung stehenden Formeln die Rede ist. Diese Menge ist offenbar variabel, denn sie vermag einmal diese, einmal jene Gegenstände zu umfassen. Unter Verwendung der Elemente des Diskursbereichs, kann eine Interpretation $I = (I_{\text{func}} \cup I_{\text{pred}})$ der Elemente aus Σ spezifiziert werden, bestehend aus der Menge der Interpretationen für die Funktionssymbole sowie der Menge der Prädikatssymbolinterpretationen. Klarerweise kann zu einem, selbst einem gegeben, Diskursbereich $\mathbb{D}$ und einer festgelegten Signatur Σ eine Vielzahl verschiedener Interpretationen existieren.

Es lässt sich nun die, zu einer gegebenen Signatur Σ, variable, veränderliche, Struktur der Semantik konzise im Tupel $\mathcal{A} = (\mathbb{D}, \Sigma, I)$ fixieren, indem man den Diskursbereich $\mathbb{D}$ und die Interpretationsfunktion I spezifiziert. Zuweilen, sofern diesbezügliche Eindeutigkeit gewährleistet ist, wird gelegentlich auch $\mathcal{A} = (\mathbb{D}, I)$ geschrieben, und das Tupel unter Auslassung der Angabe der gültigen Signatur Σ ins Feld geführt. Das durch $\mathcal{A}$ beschriebene Tupel wird *Struktur* genannt. Sie kann mit der gleichnamigen mathematischen Struktur verglichen werden, welche aus einer Grundmenge (welche hier durch $\mathbb{D}$ beschrieben ist) und einer Menge von Operationen (in unserem Fall beschrieben durch die Mengen Σ_{func} und I_{func}) sowie einer Menge von Relationen (hier durch Σ_{pred} und I_{pred} angegeben) über dieser Grundmenge besteht. Schöning insinuiert in [Sch01], dass der Bezeichner $\mathcal{A}$, als vom englischen *assignment*, „Zuweisung", kommend gedacht werden könnte. Gewiss, man mag im Tupel $\mathcal{A} = (\mathbb{D}, \Sigma, I)$ etwa die Zuweisung der Gegenstände aus $\mathbb{D}$ an die Symbole in Σ, vermittels der, durch I beschriebenen, Interpretation entdecken können.

Einen vielleicht anschaulicheren Bezeichner für das, wofür das Tupel $(\mathbb{D}, \Sigma, I)$ steht, hat Alfred Tarski in seinem bahnbrechenden Vortrag „Über den Begriff der Logischen Folgerung“, [Tar36], gewählt, und zwar den des (Σ-)*Modells* (über $\mathbb{D}$). Um dies zu würdigen, ist es an dieser Stelle erforderlich, den Begriff der *Erfüllungsrelation* einzuführen. Dabei handelt es sich formal um eine zweistellige Relation – also, nach unserer bisherigen Lesart, die die charakteristische Funktion der Relation involviert: um eine Abbildung, die zwei Argumente erwartet und zu einem Wahrheitswert auswertet –, die zwischen Tupeln der Art $(\mathbb{D}, \Sigma, I)$ auf der einen, der, in Infixnotation, linken, Seite und einer Formel auf der anderen, rechten, Seite. Die Erfüllungsrelation wird, mit Bezug auf eine Signatur Σ, durch das Zeichen „$\models_\Sigma$“ ausgedrückt, welches als Infixoperator verwendet wird. Wenn aus dem Kontext hervorgeht, um welche Signatur es sich jeweils handelt, lassen wir den Σ-Index einfach fort und schreiben lediglich das Zeichen „$\models$“. Der Satz $\mathcal{M} \models F$, bei dem $\mathcal{M} := (\mathbb{D}, \Sigma, I)$, besagt demnach, dass etwas, das nach Tarski „Modell“ genannt wird, die Formel F erfüllt. Damit ist zwar die Syntax der Erfüllungsrelation auf ein höheres Sprachniveau gehoben, aber zur Klärung der Bedeutung der Erfüllungsrelation noch kein Beitrag geleistet.

Dem sei nun dadurch abgeholfen, dass wir die Erfüllungsrelation $\mathcal{M} \models F$, mit $\mathcal{M} := (\mathbb{D}, \Sigma, I)$, von dieser Seite betrachten: Wie, auf welche Weise, kann die Formel F wahr sein? oder, die selbe Frage anders gestellt: Was sind die hinreichenden Bedingungen dafür, dass F wahr ist? Um entscheiden zu können, ob eine Formel wahr oder falsch ist, so hatten wir im letzten Kapitel gesehen, benötigen wir – neben der Angabe der nichtlogischen Symbole Σ_F, welche in F vorkommen, und insofern als gegeben angenommen werden können – eine Interpretation I dieser Symbole sowie einen Diskursbereich $\mathbb{D}$, welcher festsetzt, von welchen Gegenständen in der Formel F die Rede ist. Beides liegt mit der Spezifikation eines Modells $\mathcal{M} = (\mathbb{D}, I)$ vor. Wir können die Bedeutung der Erfüllungsrelation jetzt dadurch konkretisieren, dass wir sagen, dass Modell $\mathcal{M} := (\mathbb{D}, I)$ und Formel F (in dieser Reihenfolge) in der Erfüllungsrelation stehen – kurz, dass das Modell $\mathcal{M}$ die Formel F erfüllt – genau dann, wenn für den gewählten Diskursbereich $\mathbb{D}$, die Formel F in der Interpretation I zum Wahrheitswert *true* auswertet, formal:

Definition 9. *Sei $\mathbb{D}$ ein Diskursbereich, I eine Interpretation und Σ eine Signatur. Sei $\mathcal{M} := (\mathbb{D}, \Sigma, I)$ ein Σ-Modell über $\mathbb{D}$. Dann ist die Erfüllungsrelation, „$\models$“, wie folgt definiert.*

$$(\mathcal{M} \models F) \equiv (I(F) = \mathrm{true}) \tag{24}$$

Anstelle der Formulierung „$\mathcal{M}$ erfüllt F.“, lässt sich der Satz $\mathcal{M} \models F$, lesen als: „Das Modell $\mathcal{M}$ *ist ein Modell für* F.“, „$\mathcal{M}$ modelliert F.“ oder „Das Modell $\mathcal{M}$ *macht* die Formel F *wahr*.“

Die Nachsilbe „-bar“ deutet im Deutschen – im Englischen übernimmt diese Funktion oft die Silbe *-able* (oder *-ible*) – nicht selten ein Potenzial oder eine Möglichkeit an. So wie „sichtbar“ dafür steht, dass etwas gesehen werden kann und „drehbar“, die Möglichkeit einer Rotation verspricht, so zeigt das Wort „erfüllbar“,

im Bezug auf die Erfüllungsrelation für eine, fest gewählte, Formel die Möglichkeit zur Erfüllung dieser Formel durch ein Modell für diese Formel an. Die *Erfüllbarkeit* einer Formel F ist demnach gleichbedeutend mit der Existenz eines Modells für F: Durch geschickte Festlegung eines Diskursbereichs $\mathbb{D}$ sowie eine passende Wahl einer Interpretation I – was dem Aufstellen eines Modells $\mathcal{M} := (\mathbb{D}, I)$ entspricht –, ist es möglich, die Formel F zu erfüllen, sie wahr zu machen – falls F erfüllbar ist. Es soll künftig stets stillschweigend davon ausgegangen werden, dass nur so von Modellen $\mathcal{M} := (\mathbb{D}, \Sigma, I)$ die Rede ist, dass ihre Signatur Σ zu den jeweils zur Untersuchung stehenden Formeln F wie folgt „passt“: Für die Menge der in F vorkommenden nichtlogischen Zeichen Σ_F gilt, dass $\Sigma_F \subseteq \Sigma$. Wenn wir mit $\mathfrak{M}$ die *Menge aller Modelle* bezeichnen, so können wir das eben Gesagte formal folgendermaßen ausdrücken.

Definition 10. *Sei F eine Formel und $\mathfrak{M}$ die Menge aller Modelle. Dann ist die Erfüllbarkeit der Formel F wie folgt definiert.*

$$F \text{ ist erfüllbar} \equiv (\exists \mathcal{M} \colon (\mathcal{M} \in \mathfrak{M}) \wedge (\mathcal{M} \models F)) \tag{25}$$

Dass die Menge aller Modelle $\mathfrak{M}$ im Regelfall nicht leer ist, oder etwa nur genau ein Element enthält, liegt darin begründet, dass, wie wir bereits festgestellt hatten, sowohl die Menge $\mathbb{D}$, die Menge der Gegenstände, die den Diskursbereich ausmachen, als auch die Menge I, die Menge der möglichen Interpretationen, variabel in dem Sinne ist, als dass sie jeweils einmal diese, einmal jene Elemente zu umfassen vermag. Es kann insbesondere auch der Fall sein, der, wovon wir uns später überzeugen können, nicht selten auftritt, dass eine gegebene Formel unendlich viele Modelle – Möglichkeiten ihres Wahrseins – haben kann. Im Englischen wird an der Stelle der Erfüllbarkeit von *Satisfiability*, kurz *SAT*, gesprochen. Die Frage nach der Erfüllbarkeit einer Formel ist entsprechend als das „*SAT*-Problem“ addressierbar. Die lateinische Wurzel sind hierbei die Worte *satis*, „genügend“, „genug“ und *facere*, „machen“, wonach die Erfüllbarkeit in dieser Hinsicht das Potenzial zur „Genugmachung“ wäre. Damit eine Formel erfüllbar ist, genügt die Existenz einiger weniger Modelle. Genug ist ein einziges. Aber – anders als, der im Kapiteleingang zitierte, Seneca, allerdings in Bezug auf die Anzahl an Personen, deren Beifall es zu einer tugendhaften Handlung bedarf, meint – keines ist zu wenig. Formeln, welche, da für sie gar kein Modell existiert, nicht erfüllbar sind und damit *ipso facto* immer falsch sein müssen, darauf hatten wir früher bereits hingewiesen, heißen Widersprüche oder Kontradiktionen. Nicht erfüllbar (die Negation von erfüllbar) und unerfüllbar sind formal, das heißt, mit Bezug auf eine Formel F – wie das ja auch der Natürlichen Sprache eignet – gleichbedeutend.

Definition 11. *Sei F eine Formel und $\mathfrak{M}$ die Menge aller Modelle. Dann ist die Unerfüllbarkeit der Formel F wie folgt definiert.*

$$\begin{aligned} F \text{ ist unerfüllbar} &\equiv (\neg(\exists \mathcal{M} \colon (\mathcal{M} \in \mathfrak{M}) \wedge (\mathcal{M} \models F))) \\ &\equiv (\forall \mathcal{M} \colon \neg(\mathcal{M} \in \mathfrak{M}) \vee \neg(\mathcal{M} \models F)) \\ &\equiv (\forall \mathcal{M} \colon (\mathcal{M} \in \mathfrak{M}) \Rightarrow (\mathcal{M} \not\models F)), \end{aligned} \tag{26}$$

In der zweiten Zeile der Formel 26 von Definition 11 wurde die dritte Äquivalenz von Formel 23 ausgenutzt und anschließend die Gesetze von *De Morgan* angewendet, um schließlich zur dritten Zeile, über die Äquivalenz $(\neg F \vee G) \equiv (F \Rightarrow G)$, für beliebige Formeln F und G, zu gelangen, in der die Teilformel $\mathcal{M} \not\models F$ eine Abkürzung für $\neg(\mathcal{M} \models F)$ ist. Demnach ist eine Formel F unerfüllbar, ein Widerspruch, genau dann, wenn jedes Modell *kein Modell für* F ist.

Es wird nützlich sein, die Antwort auf die Frage zu kennen, ob $\mathcal{M} \not\models F$ dann und nur dann der Fall ist, wenn $\mathcal{M} \models \neg F$ wahr ist. Ob also der folgende Satz gilt.

Satz 3. *Sei $\mathcal{M}$ ein Modell und F eine Formel. Dann gilt:*

$$(\mathcal{M} \not\models F) \equiv (\mathcal{M} \models \neg F) \tag{27}$$

Beweis. Nehmen wir, um eine fundiert positive Antwort auf die Frage geben zu können, an, Modell $\mathcal{M} := (\mathbb{D}, \Sigma, I)$ modelliert die Formel F nicht: $\neg(\mathcal{M} \models F)$. Dann liegt nicht vor, dass $I(F) = \text{true}$, gegeben $\mathbb{D}$ – diesen, die Interpretation auf den Diskursbereich beschränkenden, Nachsatz lassen wir im Folgenden stets, als selbstverständlich einschlägig, weg. Es bleiben dann nur die Fälle, $I(F) = \text{false}$ oder $I(F)$ ist nicht definiert. Letzteres kann dann und nur dann der Fall sein, wenn in F vorkommende Symbole Σ_F, nicht zu den Symbolen Σ gehören, über welche die Interpretation I definiert ist, also insbesondere $\Sigma_F \not\subseteq \Sigma$. Diesen Fall hatten wir oben explizit ausgeschlossen, womit als einiziger Fall $I(F) = \text{false}$ verbleibt. Dann ist aber, aufgrund der Semantik negierter Formeln, die in Definition 8 angegeben ist, $I(\neg F) = \text{true}$, und somit $\mathcal{M}$ ein Modell für $\neg F$. Nehmen wir, für die andere Richtung der Behauptung, an, dass $\mathcal{M} \models \neg F$. Dann haben wir, aus der Bedeutung der Erfüllungsrelation, dass $I(\neg F) = \text{true}$, folglich, wiederum aufgrund der Semantik negierter Formeln, $I(F) = \text{false}$. Es kann also, ohne in Widerspruch mit der Definition des Begriffs der Erfüllungsrelation zu geraten, nicht sein, dass $\mathcal{M} \models F$. Es muss also $\mathcal{M} \not\models F$ gelten, womit abschließend die Behauptung gezeigt ist. □

Im letzten Kapitel hatten wir gesagt, dass die Negation der Kontradiktion die Tautologie ist. Anstatt von der Tautologie eines Satztes zu sprechen, kann man auch, gleichbedeutend, sagen, dass dieser Satz stets gilt oder *gültig* oder *allgemeingültig* ist, also ein immer, das heißt, wie wir nun insbesondere präzisieren können, in jeder Interpretation wahrer Satz, ist. Dies lässt sich nun folgendermaßen formal zum Ausdruck bringen.

Definition 12. *Sei F eine Formel und $\mathfrak{M}$ die Menge aller Modelle. Dann ist die Allgemeingültigkeit der Formel F wie folgt definiert.*

$$F \textit{ ist allgemeingültig} \equiv (\forall \mathcal{M} \colon (\mathcal{M} \in \mathfrak{M}) \Rightarrow (\mathcal{M} \models F)) \tag{28}$$

Wie wir sehen, ist F eine Tautologie genau dann, wenn $\neg F$ unerfüllbar, also eine Kontradiktion, ist. Analog ist ein Satz F falsifizierbar, wenn – man beachte die oben gezeigte Äquivalenz $(\mathcal{M} \not\models F) \equiv (\mathcal{M} \models \neg F)$ – die Formel $\neg F$ erfüllbar ist:

Definition 13. *Sei F eine Formel und $\mathfrak{M}$ die Menge aller Modelle. Dann ist die Falsifizierbarkeit der Formel F wie folgt definiert.*

$$F \text{ ist falsifizierbar} \equiv (\exists \mathcal{M} \colon (\mathcal{M} \in \mathfrak{M}) \Rightarrow (\mathcal{M} \not\models F)) \tag{29}$$

Wir wollen vereinbaren, dass künftig anstelle *einer einzigen* Formel F auch eine Menge von Formeln, etwa $\{F_1, F_2, \ldots\}$ als in der Erfüllungsrelation bezüglich einem Modell $\mathcal{M}$ stehend (oder nicht stehend) gedacht werden kann. Gemeint ist dann stets die Konjunktion der Sätze dieser Menge. Die folgende Definition formalisiert dies für den allgemeinen Fall.

Definition 14. *Sei $\{F_1, \ldots, F_n\}$ eine, möglicherweise leere, Menge von Formeln. Dann definieren wir die Äquivalenz von Formelmengen und Konjunktionen so, dass*

$$\begin{aligned} \emptyset &\equiv \top, \\ \{F_1\} &\equiv F_1, \\ \{F_1, \ldots, F_n\} &\equiv \Big(\bigwedge_{i \in \{1,\ldots,n\}} F_i \Big) \equiv (F_1 \wedge \ldots \wedge F_n). \end{aligned}$$

Ein Modell $\mathcal{M}$, welches die Formel F erfüllt und die Formel G erfüllt, ist zugleich auch ein Modell der Formel $(F \wedge G)$ und so auch andersherum, wie der Beweis des folgenden Satzes zeigt.

Satz 4. *Sei $\mathcal{M}$ ein Modell und $\{F_1, \ldots, F_n\}$ eine, möglicherweise leere, Menge von Formeln, wobei $n \in \mathbb{N}_0$. Dann gilt:*

$$\big(\mathcal{M} \models \{F_1, \ldots, F_n\}\big) \;:\equiv:\; \big(\mathcal{M} \models (F_1 \wedge \ldots \wedge F_n)\big) \;\equiv\; \big((\mathcal{M} \models F_1) \wedge \ldots \wedge (\mathcal{M} \models F_n)\big)$$

Beweis. Die erste Äquivalenz „:≡:“ ist eine Instanz der definierten Äquivalenz aus Definition 14. Wir dürfen im Beweis jeden der beiden Sätze $\big(\mathcal{M} \models \{F_1, \ldots, F_n\}\big)$ und $\big(\mathcal{M} \models (F_1 \wedge \ldots \wedge F_n)\big)$ gleichwertig verwenden. Die zweite Äquivalenz „≡“, auf der rechten Seite, ist zu zeigen. Für $n = 0$ ist die Äquivalenz von $(\mathcal{M} \models \emptyset)$, was nach Definition 14 gleichbedeutend mit $(\mathcal{M} \models \top)$ ist, und $\top$, welches als Neutrales Element der Konjunktion zugleich der Wert der „leeren“ Konjunktion ist, zu zeigen: $(\mathcal{M} \models \top) \equiv \top$. Wir hatten in Definition 12 gesehen, dass eine Tautologie $\top$ allgemeingültig ist und insbesondere von jedem Modell erfüllt wird. Der Satz $(\mathcal{M} \models \top)$ kann also, für beliebige Modelle $\mathcal{M}$, nicht falsch sein und ist selbst eine Tautologie, woraus $(\mathcal{M} \models \top) \equiv \top$ folgt. Für $n = 1$ ist – da $(\mathcal{M} \models \{F_1\}) \;:\equiv:\; (\mathcal{M} \models F_1) \;\equiv\; (\mathcal{M} \models F_1)$ eine Tautologie, und also wahr, ist – nichts zu zeigen. Für $n > 1$ sei $\mathcal{M} := (\mathbb{D}, \Sigma, I)$. Nehmen wir an, dass $\big(\mathcal{M} \models (F_1 \wedge \ldots \wedge F_n)\big)$, das heißt, dass $I\big((F_1 \wedge \ldots \wedge F_n)\big) = \text{true}$. Dann muss aber, nach Definition 8, welche insbesondere auch die Interpretation einer Konjunktionsformel definiert, jedes der Konjunkte durch I als wahr interpretiert worden sein, da andernfalls die $\wedge$-Wahrheitsfunktion zu *false* auswertet. Es gilt somit $I(F_1) = \text{true}, \ldots, I(F_n) = \text{true}$ und damit $\big((\mathcal{M} \models F_1) \wedge \ldots \wedge (\mathcal{M} \models F_n)\big)$. Nehmen wir nun, um die andere Richtung

der Äquivalenz zu zeigen, an, dass $\big((\mathcal{M} \models F_1) \wedge \ldots \wedge (\mathcal{M} \models F_n)\big)$ oder, was dasselbe ist, dass $I(F_1) = \text{true}, \ldots, I(F_n) = \text{true}$. Folglich ist der Satz $I(F_1) \wedge \ldots \wedge I(F_n)$ wahr. Dann kann aber, nach Definition 8 bezüglich Konjunktionen, der Satz $(F_1 \wedge \ldots \wedge F_n)$ nicht von I als *false* interpretiert werden und es gilt $I\big((F_1 \wedge \ldots \wedge F_n)\big) = \text{true}$ und somit $\big(\mathcal{M} \models (F_1 \wedge \ldots \wedge F_n)\big)$. Damit ist auch die andere Richtung und damit die Behauptung für $n > 1$ und somit der behauptete Satz gezeigt. □

5.2 Beispiele zu Modellen und Erfüllungsrelation

Unsere Handhabung der Begriffe des Modells und der Erfüllungsrelation festigend, seien nun einige Beispiele herangeführt. Der Einfachheit halber, gehen wir von Satzformeln aus, das heißt, durch Logische Junktoren verknüpfte nullstellige Prädikate – also um aus Aussagevariablen zusammengesetzte Sätze. Seinen dafür P und Q Aussagevariablen. Ganz formal können wir, die eingeführte Notation benutzend, schreiben: $\Sigma = (\Sigma_{\text{pred}} \cup \Sigma_{\text{func}})$, unter spezifischer Angabe $\Sigma_{\text{pred}} = \{P, Q\}$, $\Sigma_{\text{func}} = \emptyset$ sowie Stelligkeitsfunktion $\sigma = \{(P, 0), (Q, 0)\}$. Es sei erinnert daran, dass Aussagesätze, welche durch die Satzvariablen bezeichnet werden, die gegenstandslosen Sätze genannt werden können, in dem Sinne, dass sie zwar Aussagen machen, dafür aber keine Referenz auf Gegenstände – also Element des Diskursbereich $\mathbb{D}$, welche durch nullstellige Funktionen in der Menge Σ_{func} bezeichnet werden können – benötigen. Folgerichtig wählen wir die Mengen $\mathbb{D} = \Sigma_{\text{func}} := \emptyset$. Um ein Modell zu erhalten, wird eine Interpretation für die Prädikatssymbole benötigt. Wie erinnerlich bedeuten Aussagesätze einen Wahrheitswert, welcher in der Spezifikation einer Interpretation I festzulegen ist. Diese geben wir an als $I = I_{\text{pred}}$, da die Menge der Funktionssymbolinterpretationen I_{func}, mangels Gegenständen, natürlich ebenfalls leer ist. Konkret wollen wir nun setzen, dass $I^{P1Q1} := \{(P, \text{true}), (Q, \text{true})\}$, dass also durch die Interpretation I^{P1Q1} die Aussagenvariable P mit dem Wahrheitwert *true* und Q ebenfalls mit *true* belegt wird.

Zum Modell wählen wir als Beispiel zunächst einmal $\mathcal{M}^{P1Q1} := (\mathbb{D}, \Sigma, I^{P1Q1})$, in welchem der Satz P und der Satz Q wahr sind. Auf Kosten der Einfachheit und zugunsten der Präzision, lässt sich dies, für unser Beispiel, auch so ausdrücken: Das Modell $\mathcal{M}^{P1Q1}$ modelliert jeden Satz F – erfüllt ihn, „führt ihn zur Wahrheit" –, welcher, vermittels der Applikation der in F vorkommenden Wahrheitsfunktionen, den Logischen Junktoren, zu wahr auswertet, wenn man in der Formel F jedes Vorkommen der Aussagen-Variablen P, wie in I^{P1Q1} angegeben, gleichzeitig durch den Wert *true* und jede Variable Q, ebenfalls gemäß I^{P1Q1}, durch *true* ersetzt. Unser Modell $\mathcal{M}^{P1Q1}$ – nachgerade trivial dies zu erwähnen – modelliert den atomaren Satz P, formal $\mathcal{M}^{P1Q1} \models P$. Die (Wahrheit der) Behauptung P, wird unmittelbar dadurch modelliert, dass $\mathcal{M}^{P1Q1}$ den Wert $I^{P1Q1}(P) = \text{true}$ direkt spezifiziert. Dasselbe gilt natürlich ebenfalls für das Satzatom Q. Der Satz P hat, unter der spezifizierten Signatur Σ, aber noch ein weiteres Modell, nämlich $\mathcal{M}^{P1Q0} := (\mathbb{D}, \Sigma, I^{P1Q0})$, mit der Interpretation $I^{P1Q0} := \{(P, \text{true}), (Q, \text{false})\}$, da auch dieses den Satz P erfüllt. Im Satz P interessiert vordergründig, dass P wahr ist; welchen Wahrheitswert

die anderen Aussagevariablen Q haben, ist nicht wesentlich, sie dürfen als beliebig angenommen werden – solange die Variable P wahr ist. Nehmen wir ein weiteres Modell $\mathcal{M}^{P0Q1} := (\mathbb{D}, \Sigma, I^{P0Q1})$, mit Interpretation $I^{P0Q1} := \{(P, \text{false}), (Q, \text{true})\}$, hinzu, so ist $\{\mathcal{M}^{P0Q1}, \mathcal{M}^{P1Q0}, \mathcal{M}^{P1Q1}\}$ die Menge aller Modelle, welche die Aussage $(P \vee Q)$ erfüllen, da jedes Modell dieser Menge mindestens eines der Disjunkte P oder Q, und damit die Disjunktion insgesamt, wahr sein lässt. Gleichzeitig ist der Satz $(P \vee Q)$ jedoch falsifizierbar. Eine – hier die einzige – Möglichkeit seines Falschseins stellt insbesondere das Modell $\mathcal{M}^{P0Q0} := (\mathbb{D}, \Sigma, I^{P0Q0})$, mit $I^{P0Q0} := \{(P, \text{false}), (Q, \text{false})\}$, dar, in welchem beide Disjunkte zugleich als *false* interpretiert werden. Das Modell $\mathcal{M}^{P0Q0}$ falsifiziert die Formel $(P \vee Q)$, was im formalen $\mathcal{M}^{P0Q0} \not\models (P \vee Q)$ seinen Ausdruck findet. Damit steht zugleich fest, dass der Satz $(P \vee Q)$ nicht allgemeingültig ist.

Die Formel $(P \wedge Q)$ hat nur eine einzige Möglichkeit des Wahrseins und zwar die durch Modell $\mathcal{M}^{P1Q1}$ ausgedrückte. Noch weniger Modelle, konkret: gar keines, besitzt die Formel $(P \wedge \neg P)$, welche behauptet, dass der Satz P wahr und auch gleichzeitig seine Negation $\neg P$ wahr seien; ein Widerspruch, da – unabhängig davon, wie Q interpretiert wird – für die Wahl einer Interpretation $I(P) = \text{true}$ der Teilsatz $\neg P$ der zur Untersuchung stehenden Konjunktion zu falsch auswertet, die alternative Interpretation $I(P) = \text{false}$ unvermittelt zur Falschheit der Teilformel P führt, wodurch, zusammenfassend, genau eines der Konjunkte falsch sein muss, wodurch schließlich die, zur Wahrheit der Konjunktion benötigte, Wahrheit beider Konjunkte gleichzeitig durch kein Modell eingerichtet werden kann. Die Negation dieses Satzes, also $\neg(P \wedge \neg P)$, respektive, aufgrund der Kommutativität der Disjunktion, $\neg(\neg P \wedge P)$, oder, was nach *De Morgan* äquivalent ist, $(\neg\neg P \vee \neg P)$, was – da die Negation eines negierten Satzes den Satz bedeutet – dasselbe besagt wie $(P \vee \neg P)$. Diese – von der, aufgrund ihrer mathematikhistorischen Bedeutung erwähnenswerten, Intuitionistischen Logik, die weiters zu berücksichtigen, in dieser Arbeit kein Bedarf ist, hinsichtlich ihrer putativen Allgemeingültigkeit in die Kritik genommene – Formel besagt, dass eines von beiden gelten muss: Jeder Satz P ist entweder selbst wahr oder seine Negation, der Satz $\neg P$, ist wahr, ein Drittes, *Tertium*, ist nicht gegeben, *non datur*. Sie kann nicht falsch sein. Jedes Modell modelliert die Formel $(P \vee \neg P)$, weil – unabhängig davon, wie der Satz Q interpretiert wird – jede Interpretation von P diese Disjunktion wahr macht: Die Interpretation $I(P) = \text{true}$ führt zur Wahrheit des ersten und, alternativ, $I(P) = \text{false}$ zur Wahrheit des zweiten Disjunkts der Formel, womit die Disjunktion insgesamt in jeder Interpretation wahr, und damit durch jedes Modell erfüllt, sein muss.

Wir beobachten zum Abschluss unseres Beispiels noch das Folgende. Die Menge aller Modelle, die den Satz $(P \wedge Q)$ erfüllen, $\mathfrak{M}_{P \wedge Q}$, lässt sich formal angeben als $\mathfrak{M}_{P \wedge Q} := \{\mathcal{M} \mid \mathcal{M} \models (P \wedge Q)\}$. Wir hatten gesehen, dass nur ein einziges Modell, und zwar $\mathcal{M}^{P1Q1}$, welches die Satzvariablen P und Q jeweils als wahr interpretiert, zu dieser Menge gehört. Wir haben nach dem Gesagten folglich $\mathfrak{M}_{P \wedge Q} = \{\mathcal{M}^{P1Q1}\}$. Ganz ähnlich lässt sich die Menge aller Modelle, die den Satz $(P \vee Q)$ erfüllen, angeben als $\mathfrak{M}_{P \vee Q} := \{\mathcal{M} \mid \mathcal{M} \models (P \vee Q)\}$ oder, wie dargestellt, konkret als

$\mathfrak{M}_{P\vee Q} = \{\mathcal{M}^{P0Q1}, \mathcal{M}^{P1Q0}, \mathcal{M}^{P1Q1}\}$. Der Umstand, dass die Menge $\mathfrak{M}_{P\wedge Q}$ eine Teilmenge der Menge $\mathfrak{M}_{P\vee Q}$ ist, formal $\mathfrak{M}_{P\wedge Q} \subseteq \mathfrak{M}_{P\vee Q}$, wonach die Möglichkeiten des Wahrseins der Formel $(P \wedge Q)$ bereits in der Menge der Möglichkeiten des Wahrseins der Formel $(P \vee Q)$ enthalten sind, oder anders formuliert, dass jedes Modell, das die Aussage $(P \wedge Q)$ erfüllt, auch die Aussage $(P \vee Q)$ erfüllt, führt im allgemeinen Fall auf Überlegungen, welche die Bemerkungen wert sein mögen, die sich im folgenden Abschnitt dargelegt finden.

5.3 Die Logische Folgerung

Nachdem wir die für uns wesentlichen Nuancen der Erfüllungsrelation ausgedeutet haben, sind wir nunmehr bereit für einen Meilenstein in der jüngeren Geschichte der Logik, der in folgendem Zitat seinen Ausdruck findet.

> „Die Aussage [G] *folgt logisch* aus [der Aussage F] dann und nur dann, wenn jedes Modell [von F] zugleich ein Modell der Aussage [G] ist."
>
> — Alfred Tarski, *Über den Begriff der Logischen Folgerung*, [Tar36]

Der *status quo ante tarskium fuerat* bestand zunächst – wie Tarski am angegebenen Ort zumindest selbst zu erkennen gibt – darin, dass bisherige, laut Tarski nicht vollumfänglich überzeugende, Versuche einer Definition der *Logischen Folgerungsrelation*, auf rein syntaktischen Konstrukten basierten. Die revolutionäre Idee, den Begriff der Logischen Folgerung – welche aus diesem Grund auch *Semantische Folgerung* heißt – in der Semantik, nämlich über die, in den Modellen spezifizierten, Interpretationen, ihr Fundament finden zu lassen, war ein entscheidender Durchbruch.

Doch gehen wir systematisch vor. Das obige Zitat definiert die Logische Folgerung als eine zweistellige Relation über Aussagen, also Formeln, F und G. Wohl auch weil diesbezüglich wenig Raum zur Verwechslung besteht, existiert die, vielleicht nicht sonderlich glückliche, Konvention, dieser Relation denselben Bezeichner wie den der Erfüllungsrelation zuzuweisen, nämlich ebenfalls das Zeichen „$\models$". Wir werden dieser Konvention hier Folge leisten. Somit besagt der Satz $F \models G$, in dem behauptet wird, dass F und G in dieser Reihenfolge in der Logischen Folgerungsrelation stehen, wobei F und G Formeln sind, soviel wie „Aus F folgt logisch (oder semantisch) G.", „Die Formel F impliziert logisch die Formel G.", „Aus (der Wahrheit von) F darf auf (die Wahrheit von) G geschlossen werden." „Der Schluss von F auf G ist (logisch) gültig.".

Was die Semantik dieser Relation betrifft, so lässt sich dem Tarski-Zitat – wonach eine Formel G aus einer Formel F genau dann logisch folgt, wenn alle Modelle $\mathcal{M}$, die F modellieren auch G modellieren – nichts hinzufügen, außer, vielleicht, ihre, folgende, Formalisierung.

Definition 15. *Seien F und G Formeln und $\mathfrak{M}$ die Menge aller Modelle. Dann ist die Logische Folgerungsrelation, „$\models$", wie folgt definiert.*

$$(F \models G) \equiv \big(\forall \mathcal{M} \colon ((\mathcal{M} \in \mathfrak{M}) \wedge (\mathcal{M} \models F)) \Rightarrow (\mathcal{M} \models G)\big) \tag{30}$$

Wir wollen erlauben, dass die Elemente der Logischen Folgerungsrelation alternativ auch Formelmengen $\{F_1, \ldots, F_n\}$ respektive $\{G_1, \ldots, G_m\}$ sein dürfen, wobei wir $\{F_1, \ldots, F_n\} \models \{G_1, \ldots, G_m\}$ dann stets verstehen als $(F_1 \wedge \ldots \wedge F_n) \models (G_1 \wedge \ldots \wedge G_m)$.

Mit dem Konzept der Logischen Folgerung verfügen wir nun über ein mächtiges Instrument, mit welchem nicht zuletzt das wichtige Konzept der Theorien hergeleitet werden kann, auf welches wir nun zu sprechen kommen wollen.

5.4 Theorien

Unter einer *Theorie* wird nicht selten ein System von, als wahr angenommenen, Sätzen verstanden, welche untereinander in einer gewissen – sich ein- oder wechselseitig unterstützenden – Beziehung stehen. Sie umfasst einen festgelegten Diskursbereich und die Symbole, vermittelst welcher die Sätze einer Theorie formuliert werden können, haben – im Idealfall – eine eindeutig bestimmte Bedeutung. Theorien sollen konsistent, widerspruchsfrei, derart sein, dass keine zwei Sätze der Theorie als Konjunktion einen Widerspruch darstellen.

So ist beispielsweise die – durchaus heftig kontestierte – Theorie der Märkte, eine Theorie über mutmaßlich wahre Sätze mit Bezug auf Märkte: „Ein Markt entsteht durch das Zusammentreffen von Angebot und Nachfrage.“, „Es herrscht Markträumung: Jedes Angebot schafft sich selbst seine Nachfrage durch die Wahl des entsprechenden Preises.“ – Keynesianer würden bestreiten, dass dieser Satz, welcher impliziert, dass es auf einem Markt keine unverkauften Waren geben kann, zur Theorie der Märkte gehört – „Es gibt ein Existenzminimum: Wirtschaftssubjekte, welche es (das Existenzminimum) durch den Markt für Arbeit (dem Arbeitsmarkt) bestimmten Preis für Arbeitskraft pro Zeiteinheit (den Tageslohn) nicht zu realisieren vermögen, verkaufen ihre Ware (Arbeitskraft) dort nicht (und ziehen sich statt dessen etwa auf Subsistenzwirtschaft zurück, weichen auf die informellen „Märkte“ wie Diebstahl oder Raub aus, oder, realistischer und eine Erkenntnis des Wirtschaftsnobelpreisträgers Amartya Sen aus der Analyse der Bengalischen Hungersnot von 1943, verhungern ohne Aufbegehren ‚in front of well-stocked food shops protected by the state‘, [Sen81])“ – dieser, etwa von John Maynard Keynes vertretene, Satz, wonach es also Märkte geben kann, auf denen Waren nicht verkauft werden, wird von Vertretern der Neoklassischen Schule, welche sich als der Tradition der „Klassiker“ Adam Smith, David Ricardo, Thomas Malthus, folgend verstehen, als nicht zur Theorie der Märkte gehörend abgelehnt.

Im Rahmen einer Theorie kann auf Sätze, nicht notwendigerweise im Sinne der Logischen Folgerung, geschlossen werden, welche – eben dadurch, dass aus der Theorie heraus auf sie geschlossen werden kann – damit ebenfalls als zur Theorie gehörig angesehen werden. Vertreter der keynesianischen Schule würden den Satz, welcher das universelle Gesetz der Markträumung behauptet als nicht zur Theorie der Märkte gehörend ablehnen, da er im Widerspruch zum Satz von nichtnotwendigerweise geräumten Arbeitsmärkten – welcher aus dem Satz der Existenz

eines Existenzminimus folgt, und somit folglich ebenfalls zur Theorie gehört, da Anbieter von Arbeit den Verkaufspreis nicht unter das Existenzminimum absenken, entsteht potentiell ein Angebotsüberhang gegenüber der Nachfrage und der Markt wird nicht geräumt – steht und damit insbesondere der Satz der Markträumung nicht aus dem Satz des Existenzminimums folgt. Nach dem selben Muster, würden Neoklassiker in umgekehrter Richtung argumentieren: Der Satz des Existenzminimums gehört nicht zur Theorie der Märkte, da er im Widerspruch zum Satz der Markträumung steht. Tatsächlich sind die beiden eben genannten Schulen derartig uneinig, dass die jeweils für wahr geltenden Sätze der Schulen wiederum in zwei verschiedenen Theorien münden: Der Theorie der Neoklassik sowie der Theorie, welche Keynesianismus heißt. Die Schnittmenge beider Theorien – die Menge der Sätze, die in beiden Schulen als wahr gilt – ist mit keiner dieser beiden Theorien identisch.

Zwischen den Theorien, welche uns an dieser Stelle im Rahmen der Prädikatenlogik Erster Stufe interessieren – und die wir, um sie von Theorien, der Art der Theorie der Märkte, abgrenzen zu können, *Theorien der Prädikatenlogik Erster Stufe* (im Englischen *First-Order Theories*), kurz *Logische Theorien* nennen wollen – und Theorien im allgemeineren Sinne, können wir folgende Übereinstimmung feststellen. Auch eine Logische Theorie $\mathcal{T}$ ist eine Menge von Sätzen. Die Signatur und Bedeutung der Sätze dieser Theorie ist durch die Angabe eines theoriespezifischen, fixen, Diskursbereichs $\mathbb{D}$, einer Menge an, wie bisher, Funktions- und Prädikaten-, Symbolen Σ und einer fixen – und damit, wohlgemerkt, insbesondere nicht veränderlichen – Interpretation I dieser Symbole. Die zur Beschreibung einer Theorie benötigte Information lässt sich, kurz gesagt, zum Tupel $(\mathbb{D}, \Sigma, I)$ verdichten. Auch, dass alle Sätze, die aus den Sätzen der Theorie, in unserem Fall: logisch, folgen, ebenfalls zur Theorie gehören, ist Teil der Schnittmenge der Logischen Theorien und der Theorien im Allgemeinen.

Die zu klärende Frage ist nun, wie man vom Tupel $(\mathbb{D}, \Sigma, I)$, gemäß welchem die Theorie etabliert werden soll, zu der Menge derjenigen Sätze gelangt, welche die Theorie konstituiert. Die folgende Definition gibt eine Antwort auf die aufgeworfene Frage.

Definition 16. *Sei $\mathcal{A}$ eine Struktur und $\mathcal{T}_\mathcal{A}$ die durch $\mathcal{A}$ induzierte Theorie, dann definieren wir:*

$$\mathcal{T}_\mathcal{A} \equiv \{F \mid (\mathcal{A} \models F)\}$$

Die Definition verwendet die Phrase „die durch $\mathcal{A}$ induzierte Theorie“, welche, nach dem lateinischen *inducere*, „einführen“, einfach besagt, dass die Theorie durch $\mathcal{A}$ „eingeführt“, sprich: erzeugt, wird. Im Kontext Logischer – und insbesondere Mathematischer – Theorien ist es Gepflogenheit von „Struktur“, ein Begriff welchen wir am Eingang dieses Kapitels eingeführt hatten, zugunsten von „Modell“ zu sprechen. Wir wollen dieser Usance hier Rechnung tragen, indem wir den Begriff „Struktur“ im Kontext von Theorien bevorzugt verwenden werden, nicht jedoch ohne noch einmal nachdrücklich darauf hinzuweisen, dass wir die Begriffe „Struktur“ und „Modell“ vollkommen synonym verwenden. Eine Theorie $\mathcal{T}_\mathcal{A}$ ist gemäß

Definition 16 somit vollständig durch die Angabe einer Struktur $\mathcal{A}$ bestimmt und umfasst die Menge aller Formeln, für die $\mathcal{A}$ ein Modell ist. Anders ausgedrückt, ist eine Theorie $\mathcal{T}_\mathcal{A}$ die Menge aller Sätze, welche durch die Struktur $\mathcal{A}$ erfüllt werden und folglich unter der in der Struktur $\mathcal{A}$ spezifizierten Interpretation wahr sind.

Die eventuell befremdlich anmutende aber nicht gänzlich triviale Frage, ob die Struktur $\mathcal{A}$ zugleich auch ein Modell für $\mathcal{T}_\mathcal{A}$ ist, beantwortet der Folgende Satz positiv.

Satz 5. *Sei $\mathcal{A}$ eine Struktur sowie $\mathcal{T}_\mathcal{A}$ die von $\mathcal{A}$ induzierte Theorie. Dann gilt:*

$$(\mathcal{A} \models \mathcal{T}_\mathcal{A})$$

Beweis. Gemäß Definition 16 ist $\mathcal{T}_\mathcal{A} \equiv \{F \mid (\mathcal{A} \models F)\}$, womit $\mathcal{T}_\mathcal{A}$ alle Formeln F enthält, für die $\mathcal{A}$ ein Modell ist. Wir können das auch so schreiben: $\bigwedge_{F \in \mathcal{T}_\mathcal{A}} (\mathcal{A} \models F)$, wonach $\mathcal{A}$ jedes Element, das heißt, jeden Satz, aus $\mathcal{T}_\mathcal{A}$ erfüllt. Dann gilt, nach Satz 4 – gemäß welchem die Modellierung der Konjunkte auch die Modellierung der Konjunktion impliziert – aber auch, dass $\mathcal{A} \models (\bigwedge_{F \in \mathcal{T}_\mathcal{A}} F)$, oder, in Mengenschreibweise, gemäß Definition 14, $\mathcal{A} \models (\{F \mid F \in \mathcal{T}_\mathcal{A}\})$, kurz $(\mathcal{A} \models \mathcal{T}_\mathcal{A})$, das ist, die Behauptung. □

Wir hatten behauptet, dass alle Sätze, die aus einer Theorie $\mathcal{T}_\mathcal{A}$ folgen, ebenfalls zu dieser Theorie gehören. Das können wir wie folgt konkretisieren.

Satz 6. *Sei $\mathbb{D}$ ein Diskursbereich, I eine Interpretation und Σ eine Signatur, $\mathcal{A} := (\mathbb{D}, \Sigma, I)$ eine Struktur sowie $\mathcal{T}_\mathcal{A}$ die von $\mathcal{A}$ induzierte Theorie. Für alle Formeln F, mit $(\mathcal{T}_\mathcal{A} \models F)$, gilt:*

$$(\mathcal{T}_\mathcal{A} \cup \{F\}) = \mathcal{T}_\mathcal{A}$$

Beweis. Auf der Basis von Satz 5 können wir konstatieren, dass $\mathcal{A}$ ein Modell für $\mathcal{T}_\mathcal{A}$ ist. Laut Annahme haben wir, dass $(\mathcal{T}_\mathcal{A} \models F)$, was die abgekürzte Schreibweise ist für den Satz: Jedes Modell für $\mathcal{T}_\mathcal{A}$ ist auch ein Modell für F. Da nun im Speziellen die Struktur $\mathcal{A}$ ein Modell für $\mathcal{T}_\mathcal{A}$ ist, ist $\mathcal{A}$ folglich auch ein Modell für F und es gilt $(\mathcal{A} \models F)$. Damit ist dann aber, gemäß Definition 16, F ein Element der Menge $\mathcal{T}_\mathcal{A}$ und somit $(\mathcal{T}_\mathcal{A} \cup \{F\}) = \mathcal{T}_\mathcal{A}$, was wir zeigen wollten. □

Jeder Satz der aus einer Theorie logisch folgt, ist somit selbst ein Satz dieser Theorie. Die Tatsache, dass Theorien alle Sätze enthalten, die aus ihr jeweils logisch folgen, führt auf eine andere Betrachtungsweise von Theorien. Wir können das eben Gesagte nämlich auch so reformulieren: Eine Theorie ist *abgeschlossen* unter der Relation der Logischen Folgerung. Dabei ist eine Menge von Sätzen $\mathcal{T}$ dann und nur dann abgeschlossen unter der Relation der Logischen Folgerung – künftig sprechen wir hier nur kurz von Abschluss, und meinen stets den Abschluss unter der Relation der Logischen Folgerung –, wenn jeder Satz, der aus $\mathcal{T}$ folgt, bereits ein Element der Menge $\mathcal{T}$ ist, andernfalls ist die $\mathcal{T}$ nicht abgeschlossen. Es lässt sich ein „Theorifizierer“ definieren, der über einer gegebenen Satzmenge $\mathfrak{A}$ den Abschluss bildet, oder, anders ausgedrückt, die Menge $\mathfrak{A}$ in eine abschlossene Menge überführt.

Definition 17. *Sei* $\mathfrak{A}$ *eine Menge von Formeln. Dann ist der Theorieoperator* Th *über* $\mathfrak{A}$ *wie folgt definiert.*

$$\mathrm{Th}(\mathfrak{A}) := \{F \mid (\mathfrak{A} \models F)\}$$

Die Definition 17 offenbart einen alternativen Weg, um zu einer Theorie zu gelangen, sofern eine Signatur Σ und ein Diskursbereich $\mathbb{D}$ – nicht jedoch eine Interpretation I – gegeben ist. Demnach ist es nicht notwendig, eine Theorie $\mathcal{T}$ als $\mathcal{T} := \mathcal{T}_{\mathcal{A}}$ über die Struktur $\mathcal{A}$ zu erzeugen, wenn man von einer Menge – idealerweise widerspruchsfreier und wahrer – aus der Symbolmenge Σ wohlgeformter Sätze $\mathfrak{A}$ ausgehen kann und diese, mittels des Theorieoperators, abschließt, um die Theorie $\mathcal{T} := \mathrm{Th}(\mathfrak{A})$ zu generieren. Man beachte, dass beide Wege zu einer Theorie zu gelangen gleichwertig sind, insofern, als dass Theorien, ihrem Wesen nach, nichts anderes als abgeschlossene Mengen von Sätzen sind.

Die Menge der Sätze $\mathfrak{A}$ bildet den Keim oder Kern der Theorie und sollte bedachtsam zusammengestellt sein, da sie die Aufgabe übernehmen, die in strukturinduzierten Theorien die, in den Strukturen enkapsulierte Menge I der, Interpretation übernimmt: Sie bestimmen die Bedeutung. Die Sätze dieser Menge sind so „würdig“ und „wertvoll“, griechisch ἄξῐος [áxīos], von ἄγω [agō], „schätzen“, dass sie *Axiome* – griechisch ἀξίωμα [axíōma], aus ἀξιόω [axióō], „für würdig halten“ und -μα [-ma], was, als Suffix das Resultat einer Handlung bezeichnet, zusammen also „das Würdige“, „das Wertvolle“ – genannt werden. Die vermittelst der Axiome $\mathfrak{A}$ durch Anwendung des Theorieoperators erzeugte Menge $\mathcal{T} := \mathrm{Th}(\mathfrak{A})$ ist die durch die Axiome induzierte Theorie. Gleichzeitig ist sie die Menge aller Sätze, die in dieser Theorie wahr sind. Man kann nun auch sagen, dass sie wahr sind, eben weil sie aus den Axiomen logisch folgen, denn die Menge ist ja durch den Abschluss unter der Relation der Logischen Folgerung mit Bezug auf die Axiomenmenge erzeugt worden. Ein Satz F, welcher unter einer Theorie $\mathcal{T}$ wahr ist, also $F \in \mathcal{T}$, heißt ein *Theorem* – vom griechischen θεώρημα [theōrēma], dem, wörtlich, „göttlich Geschauten“: θεός [theós], „Gott“, ὁράω [horáō], „sehen“, „schauen“ und -μα [-ma] sind die Komposita des Wortes – dieser Theorie. Man kann das auch so wenden: Theoreme sind die Sätze, welche in der Theorie – entweder direkt aus den Axiomen oder indirekt aus anderen bereits direkt oder indirekt logisch gefolgerten Theoremen – *bewiesen* werden können, wobei ein *Beweis* aus der Angabe der, bei den Axiomen beginnenden, Kette Logischer Folgerungen besteht, an deren Ende das bewiesene Theorem steht. Man beachte, dass auch Axiome $A \in \mathfrak{A}$ Theoreme sind. Ihr Beweis ist die unspektakuläre Eingliedkette A.

Es sollte klar geworden sein, dass jede Theorie $\mathcal{T}$ zusammen mit ihrer „Komplementärmenge“ $\overline{\mathcal{T}} := \{\neg F \mid F \in \mathcal{T}\}$ eine Menge von Sätzen – eine Sprache, englisch *Language*, kurz $\mathcal{L}$ – bildet. Wir wollen für $\mathcal{L}_{\mathcal{T}} := (\mathcal{T} \cup \overline{\mathcal{T}})$ kurz den Ausdruck „Sprache der Theorie $\mathcal{T}$“ wählen. Wir folgen damit Kroening und Strichman, [KS08], – etwa wenn sie von der Expressivität verschiedener Theorien sprechen – bei denen jedoch nicht immer klar ist, dass sie damit neben der Satzmenge $\mathcal{T}$ – welche ja nur die Hälfte dieser Sprache ausmacht – auch die Menge $\overline{\mathcal{T}}$ im Blick haben. Offenkundig ist $\mathcal{L}_{\mathcal{T}}$ – für beliebige Theorien $\mathcal{T}$ – eine Teilmenge derjenigen Sätze, die

durch die Sprache der Prädikatenlogik Erster Stufe im Allgemeinen dargestellt werden können. Die Sprache der Prädikatenlogik ist umfassender und somit prinzipiell ausdrucksmächtiger als die Sprache $\mathcal{L}_{\mathcal{T}}$. Eine weitere betonenswerte Besonderheit von Theorien ist die Tatsache, dass die Bedeutung der nichtlogischen Zeichen – das heißt, der Funktions- und Prädikatssymbole – genau so *feststeht*, wie die Bedeutung der Logischen Zeichen „$\forall$“, „$\neg$“, „$\wedge$“ und so weiter. Ihre Bedeutung ergibt sich für den strukturinduzierten Fall direkt aus der in der Struktur $\mathcal{A} = (\mathbb{D}, \Sigma, I)$ spezifizierten Interpretation I dieser Zeichen. Im Fall der Axiomeninduktion der Theorie, ergibt sich die Bedeutung der nichtlogischen Zeichen aus den Axiomen zusammen mit der im Diskursbereich angegebenen Menge der zur Untersuchung gehörigen Gegenstände: Mit Hilfe der Axiome ließe sich die eindeutige Interpretation I aller Prädikats- und Funktionssymbole, sollte Bedarf dazu aufkommen, rekonstruieren.

5.5 Satisfiability Modulo Theory

Damit liegt bei der Sprache einer Theorie $\mathcal{L}_{\mathcal{T}} := (\mathcal{T} \cup \overline{\mathcal{T}})$ – wohlgemerkt, im Unterschied zur Prädikatenlogik Erster Stufe im Allgemeinen Fall, in der ein nichttautologischer, nichtkontradiktorischer Satz, abhängig von der gewählten Interpretation einmal wahr und einmal falsch sein mag – stets fest, ob ein Satz F dieser Sprache wahr oder falsch ist. Er ist wahr, dann und nur dann, wenn $F \in \mathcal{T}$ und falsch genau dann, wenn, $F \in \overline{\mathcal{T}}$. Im Kontext einer Theorie kann ein Satz F folglich nicht erfüllt oder nicht erfüllt, also „wahr“ oder „falsch gemacht“, werden: Er ist ja bereits wahr, wenn in $\mathcal{T}$, oder falsch, wenn in $\overline{\mathcal{T}}$. Dennoch hat sich im Kontext von Theorien in der Literatur die Formulierung „Erfüllbarkeit bezüglich einer Theorie“ – in der englischen Bezeichnung, die wir im Folgenden stets verwenden wollen, *Satisfiability Modulo Theory*, kurz *SMT*: Erfüllbarkeit *modulo*, lateinischer Ablativ von *modulus*: „nach Maß“, „auf die Weise“, einer gegebenen Theorie – durchgesetzt: Eine – da, wie wir gesehen hatten, die Erfüll*barkeit*, eines Satzes in der Sprache einer Theorie bereits stets feststeht – eklatante Fehlbenennung, der wir uns hier, *nolens volens*, anzuschließen genötigt sehen. Die Frage, die im Rahmen der *Satisfiability Modulo Theory* eigentlich mit Bezug auf einen Satz interessiert, ist die, nicht ob der Satz, vor dem Hintergrund der gegebenen Theorie, erfüllbar, sondern ob er *erfüllt* ist und man hätte vielleicht besser die Bezeichnung „*Satisfied Modulo Theory*“ gewählt.

Doch damit ist die Fehlbenennung noch nicht komplett. Die Bezeichnung *Satisfiability Modulo Theory* ist nicht nur im Hinblick auf den Begriff *Satisfiability* unglücklich gewählt. Auch die Bezeichnung *Theory* bezeichnet in diesem Kontext nicht selten gar keine Theorie, sondern nur einen Bruchteil einer solchen. Es ist nämlich durchaus möglich, die Einschränkung, welche die Sprache einer Theorie, in Relation zur Sprache der Prädikatenlogik Erster Stufe allgemein, ohnehin darstellt, noch weiter zu verschärfen. Die Autoren Kroening und Strichman, welche mit ihrem Buch „Decision Procedures“, [KS08], konkrete Verfahren im Kontext der *Satisfiability Modulo Theory* bezüglich ausgewählter „Theorien“ vorstellen, präzisieren, dass es sich bei letzteren zuweilen, tatsächlich um „fragments of the logic“,

„Logikfragmente“, bezüglich dieser Theorien handelt. Damit ist die Einschränkung der Menge der *Logischen* Zeichen gemeint. Im Kontext einer Theorie $\mathcal{T}$ kann entspechend auch die Bezeichnung *Theoriefragment* gewählt werden. Es unterscheidet sich, anschaulich gesprochen, von der Menge von Sätzen der, in den Logischen Zeichen unbeschränkten, Theorie $\mathcal{T}$ dadurch, dass aus ihm alle Sätze entfernt werden, welche Logische Zeichen enthalten, die im eingeschränkten Zeichensatz Logischer Zeichen nicht vorhanden sind; und analog für $\overline{\mathcal{T}}$. So umfasst die Sprache, das Theoriefragment, welches *Marabou* verwendet – und auf das wir gleich ausführlicher zu sprechen kommen werden –, neben den im Einzelnen noch zu besprechenden nichtlogischen Symbolen, lediglich die Logischen Zeichen der Menge $\{\exists, \wedge\}$, bei der bemerkenswerterweise insbesondere die Negation fehlt. Betrachten wir nun also die Sprache *Marabou*s, in welcher wir das Verifikationsproblem von Eigenschaften Neuronaler Netze zu formulieren haben, im Folgenden genauer.

5.6 Die Sprache Marabous

Die Sprache *Marabou*s ist das Theoriefragment der sogenannten quantorenfreien Linearen Arithmetik der Reellen Zahlen, englisch *quantifier-free linear real arithmetic*, kurz *QFLRA*. Dabei haben wir es wiederum mit einer Fehlbezeichnung zu tun, da, wie wir im Letzten Abschnitt konstatiert hatten, die Sprache wider die Bezeichnung den Existenzquantor enthält. Es lässt sich diese Bezeichnung schwach rechtfertigen durch den Verweis auf den Mathematikunterricht und die Weise wie Kinder in der Schule etwa auf Gleichungen und Ungleichungen in der Theorie der Reellen Zahlen abgerichtet – „Das Lehren der Sprache ist hier kein Erklären, sondern ein Abrichten.“, Wittgenstein, [Wit53] – werden: Die „Formel“ $x+1=2$ hat die einzige Lösung $x=1$. Zwar ist die Formel quantorenfrei, dafür aber nicht eindeutig interpretierbar, da die Variable x ungebunden ist. Wir müssen die Quantifizierung implizit voraussetzen: Der Satz $\forall x\colon x+1=2$ hat sicher nicht die Lösung 1, sondern, wie dies bei falschen Sätzen stets der Fall zu sein pflegt, gar keine. Also muss die Formel $\exists x\colon x+1=2$ gemeint sein und Schulkinder können, gegebenenfalls auch ohne im Besitz einer Lösung zu sein, beherzt erwidern: „Ja, das ist wahr“. Hier werden solche Gleichungen oder Ungleichungen folglich so verstanden, dass alle Variablen existenzquantifiziert sind und es wird so getan *als ob* die entsprechenden Gleichungen oder Ungleichungen quantorenfrei wären.

Tatsächlich ist die Sprache *Marabou*s eine Sprache, welche Gleichungen und Ungleichungen der eben vorgestellten Art zum Gegenstand hat. Die Sprache ist ein Fragment einer Teilmenge der Theorie der Reellen Zahlen. Der Diskursbereich, die Art der Gegenstände, von denen die Rede ist, ist in *Marabou* also die Menge der Reellen Zahlen – und zwar der Gegenstände, nicht der Zeichen – und wir können festhalten, dass in der Sprache *Marabou*s $\mathbb{D}_{\mathit{Marabou}} = \mathbb{R}$ gilt. *Marabou* kennt die jeweils zweistelligen Prädikatssymbole „$\leq$“: „Kleinergleich“ und „$\geq$“: „Größergleich“ und nur diese sowie die jeweils zweistelligen Funktionssymbole „$+$“: „Plus“ und „$\cdot$“: „Mal“. Dazu kommen die überabzählbar unendlich vielen nullstel-

ligen Funktionssymbole, die Konstanten, mit denen wir für gewöhnlich die Reellen Zahlen bezeichnen: $\boxed{0}$, $\boxed{-0.1}$, $\boxed{\pi}$ um nur einige Beispiele zu nennen, wobei wir die Zeichen „eingeboxt“ haben, um sie von den jeweiligen Gegenständen 0, -0.1, π zu unterscheiden; die Gesamtheit aller dieser Symbole bezeichnen wir entsprechend mit $\boxed{\mathbb{R}}$.

Damit können wir die Menge der nichtlogischen Zeichen, das heißt, der Symbole $\Sigma_{\textit{Marabou}} = (\Sigma_{\text{pred}} \cup \Sigma_{\text{func}})$ der Sprache *Marabou*s angeben als

$$\Sigma_{\textit{Marabou}} = (\{\leq, \geq\} \cup \{+, \cdot, \boxed{\mathbb{R}}\}).$$

Marabou verfügt damit über eine Teilmenge des Zeichensatzes, den man gemeiniglich mit der Arithmetik der Reellen Zahlen in Verbindung bringt. Das Logische Fragment, welches die Sprache *Marabou*s darstellt, beschränkt, wie bereits besprochen, die Logischen Zeichen in wohlgeformten *Marabou*-Formeln auf Zeichen der Menge $\{\exists, \wedge\}$. Aufgrund der Abwesenheit des Allquantors – und der Negation, vermittels der wir im Bedarfsfall den Allquantor über die letzte Logische Äquivalenz aus Formel 23 zu emulieren imstande wären – können wir, in der oben, „im Schulunterricht“, eingeführten, saloppen Art, davon sprechen, dass Formeln in *Marabou* stets „quantorenfrei“ sind und somit höchstens Existenzquantoren enthalten.

Bisher haben wir also gesehen, inwiefern die Sprache *Marabou*s „quantorenfrei“ genannt werden kann und wir haben angedeutet, inwiefern diese Sprache syntaktische Ähnlichkeiten mit der Arithmetik Reeller Zahlen aufweist, mit welcher es einen Teil der verwendeten Symbole gemein hat. Vom Theoriefragment der sogenannten quantorenfreien linearen Arithmetik der Reellen Zahlen, *QFLRA*, der Sprache *Marabou*s, konnten wir bisher also *QF* und *RA* syntaktisch herleiten. Es fehlt das *L*. Es geht mit dem Begriff der Linearen Arithmetik, wofür das *L* steht, eine weitere Einschränkung der Sprache einher, die die Sätze der Sprache *Marabou*s auf Konjunktionen Linearer Ungleichungen beschränkt. Eine *Lineare Ungleichung* ist dabei ein atomarer Satz der Sprache *Marabou*s, in welchem in jedem Multiplikations-Term – einem Term, in dem das äußerste Funktionssymbol das „$\cdot$“-Funktionszeichen ist – *mindestens eines* der Argumente zu einer Konstante auswertet und insbesondere zu nichts auswertet, was eine Variable enthält. Dem entspricht etwa der Term $0.1 \cdot (x \cdot -2.3)$, nicht jedoch der Term $((-4.5 \cdot y) \cdot (6 \cdot z))$. Folglich ist etwa $\exists x\colon (0.1 \cdot (x \cdot -2.3) \leq 7)$ Ausdruck der Linearen Arithmetik Reeller Zahlen, und somit ein Satz der Sprache *Marabou*s und $\exists y \exists z\colon ((-4.5 \cdot y) \cdot (6 \cdot z)) \geq -8.9$ kein Ausdruck der Linearen Arithmetik Reeller Zahlen und also auch kein Satz der Sprache *Marabou*s. Die Linearität der Ausdrücke ist eine zur Syntax der Sprache gehörige Eigenschaft. Für die Formalisierung ist diese Definition hilfreich:

Definition 18. *Sei F eine Formel. Unter dem* existenziellen Abschluss *oder* Existenzabschluss *der Formel F verstehen wir die Formel F selbst, im Fall, dass in ihr alle vorkommenden Variablen durch Quantoren gebunden sind, andernfalls den Existenzabschluss der Formel $\exists x F$, falls x in F ungebunden vorkommt. Die vollkommen analoge Definition mittels des Allquantors anstelle des Existenzquantors definiert den* Allabschluss.

Der Existenzabschluss bindet demnach, rekursiv, alle in einer Formel frei vorkommenden Variablen durch jeweils einen eigenen Existenzquantoren. Folgende Definition spezifiziert die Grundlagen der Syntax der Sprache *Marabou*s.

Definition 19. *Sei $\mathbb{D} = \mathbb{R}$, der Diskursbereich* Marabous *und V eine Menge von Variablen. Sei $c \in \mathbb{D}$. Ein atomarer* Proto-Satz *der Sprache* Marabous*, ist eine, tatsächlich, quantorenfreie Formel, die sich auf eine der beiden folgenden Formen – etwa per Äquivalenzumformung – zurückführen lässt.*

$$\Big(\sum_{c_i \in \mathbb{D},\, x_i \in V} c_i \cdot x_i \leq c \Big) \quad \textit{oder} \quad \Big(\sum_{c_i \in \mathbb{D},\, x_i \in V} c_i \cdot x_i \geq c \Big)$$

Jeder atomare Proto-Satz ist in der Sprache Marabous *ein wohlgeformter Proto-Satz. Seien P und Q in der Sprache* Marabous *wohlgeformte Proto-Sätze, dann ist auch $(P \wedge Q)$ ein in der Sprache* Marabous *wohlgeformter Proto-Satz. Nichts sonst ist ein in der Sprache* Marabous *wohlgeformter Proto-Satz.*

Die Bezeichnung „Proto"-Satz, vom griechischen πρό [pró], zeitlich oder als Ausdruck eines Vorzugs: „vor" und τό [tó], „deshalb", bezeichnet wahlweise den mutmaßlich wichtigsten Teil, den „Kern", eines Satzes oder eine Variante eines, noch unvollständigen, „Ur"-Satzes. Man beachte nämlich, dass ein Proto-Satz kein Satz ist, der für eine Aussage steht, da in ihm Variablen vorkommen, die durch keinen Quantor gebunden sind. Wahrheitswertfähig wird ein Proto-Satz aber durch seinen Existenzabschluss. Die gesamte Syntax der Sprache *Marabou*s können wir entsprechend formal folgendermaßen festhalten.

Definition 20. *Der Existenzabschluss eines in der Sprache* Marabous *wohlgeformten Proto-Satzes ist ein in der Sprache* Marabous *wohlgeformter Satz. Nichts sonst ist ein in der Sprache* Marabous *wohlgeformter Satz.*

Widmen wir uns nun der Semantik der quantorenfreien Linearen Arithmetik der Reellen Zahlen, das ist, der Sprache *Marabou*s, zu. Die Menge aller wahren Sätze dieser Sprache ist, wie gesagt, ein Fragment – eine echte Teilmenge – der Theorie der Reellen Zahlen. Wir gelangen von der Theorie der Reellen Zahlen, zum Theoriefragment *QFLRA*, indem wir aus der Theorie alle Sätze entfernen, die Logische oder nichtlogische Zeichen enthalten, die im Theoriefragment nicht definiert sind. Dafür ist es vorteilhaft, die Theorie der Reellen Zahlen zuvor einzuführen, wobei wir uns im Folgenden auf Rudins „Principles of Mathematical Analysis", [Rud76] und Elias Zakons, „Mathematical Analysis", [Zak04], stützen. Insbesondere sind die dies zuwege bringenden Axiome – zusätzlich zu den anschließend einzuführenden – die folgenden Körperaxiome, englisch *field axioms*, welche wir auf den zu induzierenden „Körper" der Reellen Zahlen angepasst haben.

Definition 21. *Sei der Diskursbereich die Menge der zu definierenden Reellen Zahlen, mithin* $\mathbb{D} = \mathbb{R}$*. Die Körperaxiome sind wie folgt definiert.*

$$\begin{aligned}
\textit{(Assoziativität)} \quad & \forall x \forall y \forall z \colon (x + (y + z) = (x + y) + z) \wedge \\
& \qquad (x \cdot (y \cdot z) = (x \cdot y) \cdot z) \\
\textit{(Kommutativität)} \quad & \forall x \forall y \colon ((x + y) = (y + x)) \wedge ((x \cdot y) = (y \cdot x)) \\
\textit{(Distributivität)} \quad & \forall x \forall y \forall z \colon x \cdot (y + z) = (x \cdot y) + (x \cdot z) \\
\textit{(Neutr. Elem. d. Addition)} \quad & \forall x \colon (x + 0) = x \\
\textit{(Neutr. Elem. d. Multiplikation)} \quad & \forall x \colon ((x \cdot 1) = x) \wedge \neg(1 = 0) \\
\textit{(Inverses der Addition)} \quad & \forall x \exists y \colon (x + y) = 0 \\
\textit{(Inverses der Multiplikation)} \quad & \forall x \exists y \colon \neg(x = 0) \Rightarrow ((x \cdot y) = 1)
\end{aligned}$$

Man hüte sich an dieser Stelle davor, vortheoretisches Wissen in die Definition hineinzuprojizieren. Alles was wir vor der Definition 21 etwa über den Gegenstand 1 wissen, ist, dass er ein Element des Diskursbereichs $\mathbb{D}$ – welcher hier die, im Hinblick auf ihre Eigenschaften *erst zu definierenden*, Reellen Zahlen, und nur diese, als Elemente hat – ist. Die Körperaxiome spezifizieren nun unter Anderem einige Eigenschaften dieses Gegenstands 1, beispielsweise, dass der Gegenstand 1 nicht identisch (die implizite Semantik der Identitätsrelation wird indes erst durch die im Folgenden vorgestellten Axiome abschließend komplettiert) ist mit dem Gegenstand 0 und dass jeder Gegenstand des Diskursbereichs, wenn mit dem Gegenstand 1 multipliziert – die Semantik der Multiplikationsfunktion wird ebenfalls erst durch die Körperaxiome implizit festgelegt –, wiederum sich selbst als Wert ergibt. Hinzuzunehmen sind des Weiteren die folgenden Axiome der sogenannten Totalordnung.

Definition 22. *Sei* $\mathbb{D} = \mathbb{R}$*. Die Axiome der Totalordnung sind wie folgt definiert.*

$$\begin{aligned}
\textit{(Reflexivität)} \quad & \forall x \colon (x \leq x) \\
\textit{(Antisymmetrie)} \quad & \forall x \forall y \colon ((x \leq y) \wedge (y \leq x)) \Rightarrow (x = y) \\
\textit{(Transitivität)} \quad & \forall x \forall y \forall z \colon ((x \leq y) \wedge (y \leq z)) \Rightarrow (x \leq z) \\
\textit{(Totalität)} \quad & \forall x \forall y \colon (x \leq y) \vee (y \leq x)
\end{aligned}$$

Es ist die Verträglichkeit der Addition respektive Mulitplikation mit der Totalordnung, wie folgt, axiomatisch zu ergänzen.

Definition 23. *Sei* $\mathbb{D} = \mathbb{R}$*. Die Axiome der Kompatibilität der Totalordnung bezüglich der Addition und Multiplikation sind wie folgt definiert.*

$$\begin{aligned}
\textit{(Additionskompatibilität)} \quad & \forall x \forall y \forall z \colon (x \leq y) \Rightarrow ((x + z) \leq (y + z)) \\
\textit{(Multiplikationskompatibilität)} \quad & \forall x \forall y \colon ((0 \leq x) \wedge (0 \leq y)) \Rightarrow (0 \leq (x \cdot y))
\end{aligned}$$

Um die benötigten Axiome zu komplettieren, wird noch das folgende Axiom zur Vollständigkeit der Ordnung benötigt.

Definition 24. *Sei $\mathbb{D} = \mathbb{R}$. Das Axiom der Ordnungsvollständigkeit ist wie folgt definiert.*

(Ordnungsvollständigkeit) *Jede beschränkte nicht-leere Teilmenge (über $\mathbb{R}$) hat ein Supremum (in $\mathbb{R}$).*

Die Gesamtheit der in Definition 21, Definition 22, Definition 23 und Definition 24 spezifizierten Axiome induziert die Theorie der Reellen Zahlen, aus der sich, durch Streichen aller Sätze mit dort nicht erklärten Zeichen, auch das Theoriefragment der wahren Sätze der Sprache *Marabou*s, der quantorenfreien Linearen Arithmetik der Reellen Zahlen, entnehmen lässt. Die Menge aller falschen Sätze dieser Sprache sind – man sei der Tatsache wohl eingedenk, dass der Sprache insbesondere das Negationszeichen abgeht – alle syntaktisch korrekten, wohlgeformten, Sätze der Sprache, die nicht Element des Theoriefragments sind. Es sei an dieser Stelle nachdrücklich darauf hingewiesen, dass das Theoriefragment, die Menge aller in der Sprache *Marabou*s wahren Sätze, *keine Theorie* ist. Denn die Sprache ist nicht abgeschlossen, wie folgende Überlegung zeigt. Seien F und G wahre Sätze in *QFLRA*, der Sprache *Marabou*s, und somit zwei Elemente des Theoriefragments. Da $\{F, G\} \models (F \Leftrightarrow G)$, müsste, wäre das Theoriefragment zugleich eine Theorie, auch der Satz $(F \Leftrightarrow G)$ Element des Theoriefragments sein. Dies ist jedoch ausgeschlossen, da das Zeichen „$\Leftrightarrow$“ in *QFLRA* nicht definiert ist. Bemerktenswerterweise enthält *QFLRA* noch nicht einmal Tautologien, das Minimum der wahren Sätze, die selbst die – aus einer leeren Axiomenmenge induzierte – Leere Theorie enthält (sofern diese mindestens über ein Prädikat im Zeichensatz sowie gegebenenfalls mindestens einen Gegenstand im Diskursbereich verfügt).

Was die Definitionen anbelangt, beachte man des Weiteren folgende Punkte. Die Identitätsrelation $t_1 = t_2$, welche die Gleichheit zweier durch die Terme t_1 und t_2 vermittelten Gegenstände ausdrückt, ist aus der Sicht der Sprache *Marabou*s „syntaktischer Zucker“ für den Ausdruck $(t_1 \leq t_2) \wedge (t_1 \geq t_2)$ und jener durch diesen in den Axiomen ersetzt zu denken – was einige der Axiome überflüssig macht, etwa das Antisymmetrie-Axiom. Andersherum ist, aus der Sicht der Axiome, die in *Marabou* gegebene Relation $(t_1 \geq t_2)$ verzichtbar, da diese auch als $(t_2 \leq t_1)$ ausdrückbar ist.

Ein einfacherer Weg, um zur Semantik der Sprache *Marabou*s zu gelangen, ist die direkte Spezifikation der Bedeutung aller nichtlogischen Symbole in Form einer Interpretation. Da dies, ohne die vorherige Einführung der Theorie der Reellen Zahlen den Beigeschmack eines Zirkelbeweises – in welchem das vorausgesetzt wird, was erst noch zu zeigen ist, im Englischen wird hier treffend von *begging the question* geprochen – trägt, erfolgt die Vorstellung erst an dieser Stelle.

Definition 25. *Die Interpretation I_{Marabou} der Symbole Σ_{Marabou}, mit zugehöriger Stellig-*

keitsfunktion σ, über dem Diskursbereich $\mathbb{D}_{Marabou}$ sind wie folgt definiert:

$$I_{Marabou}(\boxed{r}) := r, \textit{ für alle Konstanten-Symbole } \boxed{r} \in \Sigma_{Marabou} \textit{ und } r \in \mathbb{D}_{Marabou},$$
$$I_{Marabou}(\boxed{+}) := \textit{Die Additionsfunktion über den Reellen Zahlen},$$
$$I_{Marabou}(\boxed{\cdot}) := \textit{Die Multiplikationsfunktion über den Reellen Zahlen},$$
$$I_{Marabou}(\boxed{\leq}) := \textit{Die Kleiner-Oder-Gleich-Relation über den Reellen Zahlen},$$
$$I_{Marabou}(\boxed{\geq}) := \textit{Die Größer-Oder-Gleich-Relation über den Reellen Zahlen},$$

wobei die Konstanten-Symbole $\boxed{r}$, die Funktionssymbole sind, für die $\sigma(\boxed{r}) = 0$, also die nullstelligen Funktionssymbole.

Der Umstand, dass damit, nach Definition 25, die Semantik der Sprache *Marabou*s in gewisser Hinsicht mächtiger ist als die Syntax – so kann etwa der Satz $(\exists x \exists y \colon (x \cdot x) \leq y)$ semantisch, als wahr, ausgewertet werden, obwohl er, da nichtlinear, in der Sprache der *QFLRA* nicht wohlgeformt ist –, ist hierbei nicht von Bedeutung.

Da wir nun die erforderliche Syntax und Semantik kennen, sind wir somit in Kenntnis der Sprache *Marabou*s. Es sei die Aufforderung gestattet, sich überzeugen zu mögen, dass die Formalisierung Neuronaler Netze in Kapitel 3.3 fast vollständig in der Sprache der quantorenfreien linearen Arithmetik Reeller Zahlen, das heißt, die Sprache *Marabou*s, geschehen ist. Durchaus eine essentielle Frage ist es, ob und inwiefern wir das formalisierte Neuronale Netz N, in Form der Netzformel F_N, als zur Sprache *Marabou*s gehörig betrachten können, wenn F_N, unauflöslich, in dieser Sprache nicht erlaubte Logische Zeichen enthält. Wie wir spätestens in Formel 22 des Satzes 2 gesehen hatten, ist dies etwa das Logische inklusive Oder, „$\vee$“, welches, wie wir eben definiert hatten, zunächst einmal nicht zum Zeichensatz der Sprache der quantorenfreien Linearen Arithmetik Reeller Zahlen, *QFLRA*, gehört.

Es gibt folgenden Kniff, wie dem Sprachumfang *Marabou*s das Logische Oder hinzugefügt werden kann.

Satz 7. *Seien $F_1, \ldots, F_n$, mit $n > 1$, Proto-Sätze der Sprache Marabous und sei $F^{\exists}$ die Formel die durch den Existenziellen Abschluss der Formel $(F_1 \vee \ldots \vee F_n)$ entsteht. Schließlich seien $F_1^{\exists}, \ldots, F_n^{\exists}$ die jeweiligen existenziellen Abschlüsse der Formeln $F_1, \ldots, F_n$. Dann ist der Satz $F^{\exists}$ wahr, genau dann, wenn mindestens ein Satz der Menge $\{F_1^{\exists}, \ldots, F_n^{\exists}\}$ wahr ist.*

Beweis. Man beachte zunächst, dass jeder Satz der Menge $\{F_1^{\exists}, \ldots, F_n^{\exists}\}$ einen wohlgeformten Satz der Sprache *Marabou*s darstellt, denn aus einem wohlgeformten Proto-Satz F_i, wobei $i \in \{1, \ldots, n\}$, entsteht durch den Existenzabschluss der wohlgeformte Satz $F_i^{\exists}$. Des Weiteren sei auf die sogenannte „Oder-Verträglichkeit“ des Existenzquantors verwiesen, nach welcher, gegeben zwei Formeln G und H und Variable x gilt $(\exists x \colon (G \vee H)) \equiv ((\exists x \colon G) \vee (\exists x \colon H))$, welcher somit impliziert, dass der Existenzabschluss der Disjunktion der Disjunkte dasselbe bedeutet wie die Disjunktion der jeweiligen Existenzabschlüsse der Disjunkte im Einzelnen. Dass die

Disjunktion $H_1 \vee \ldots \vee H_n$, wie bisher mit $n > 1$, schließlich dann und nur dann wahr ist, wenn mindestens ein Element der Menge der Disjunkte $\{H_1, \ldots, H_n\}$ wahr ist, ist exakt die Semantik der Wahrheitsfunktion des Logischen Oders „$\vee$“. □

Wir wollen künftig jede Formel $F := (G \vee H)$, bei der G und H wohlgeformte Proto-Sätze der Sprache *Marabou*s sind, ebenfalls zu den wohlgeformten Sätzen der Sprache *Marabou*s zählen und sie somit zur Genese wohlgeformter Sätze dieser Sprache – durch die Bildung des Existenzabschlusses – ebenfalls zulassen. Um ein veranschaulichendes Beispiel davon zu geben, wie eine Disjunktion in der Sprache *Marabou*s ausgewertet wird, betrachten wir das Folgende. Der Existenzabschluss des Proto-Satzes $(x \leq 3) \vee (x \geq 5)$ erzeugt die Behauptung, dass x nicht im offenen Intervall $3 < x < 5$ liegt. Die Wahrheit dieses Satzes kann mit den Mitteln der Sprache *Marabou*s wie folgt entschieden werden. Wenn $\exists x\colon (x \leq 3)$, der Existenzabschluss des Proto-Satzes $(x \leq 3)$, wahr ist oder $\exists x\colon (x \geq 5)$, der Existenzabschluss des Proto-Satzes $(x \geq 5)$, wahr ist, oder beide – was im konkreten Fall schwierig sein dürfte, da die Gespaltenen Zahlen bisher nicht definiert worden sind –, dann ist der Satz $\exists x\colon (x \leq 3) \vee (x \geq 5)$ wahr, andernfalls falsch.

Da wir somit die Menge der Logischen Zeichen, über die die Sprache *Marabou*s verfügt, als die Menge $\{\exists, \wedge, \vee\}$ betrachten können, ist jedes ReLU-Netzwerk N als Formel F_N in der Sprache *Marabou*s formalisierbar. Dass tatsächlich nur diese Logischen Zeichen benötigt werden, beweist etwa die Darstellung von F_N in Disjunktiver Normalform F_N^{DNF}, in Formel 22 des Satzes 2.

Durch die Disposition über die Sprache *Marabou*s vermögen wir fürderhin mit *Marabou* „zu sprechen“, beispielsweise, um ausfindig zu machen, ob ein gegebenes Neuronale Netz eine bestimmte Eigenschaft hat. Der Formulierung von Eigenschaften eines Neuronalen Netzes in der Sprache *Marabou*s ist das nächste Kapitel gewidmet.

6 Eigenschaften Neuronaler Netze

> $1 + 1 = 2$. [...] The above proposition is occasionally useful.
>
> Alfred North Whitehead und Bertrand Russell,
> *Principia Mathematica* [WR27]

> Man kann $\vdash \Phi(A)$ lesen: „A hat die Eigenschaft Φ“. [...] Da in dem Ausdrucke $\Phi(A)$ das Zeichen Φ an einer Stelle vorkommt, und da wir es durch andere Zeichen Ψ, Ξ ersetzt denken können – wodurch dann andere Functionen des Argumentes A ausgedrückt würden –, so kann man $\Phi(A)$ als eine Function des Argumentes Φ auffassen.
>
> Gottlob Frege, 1879,
> *Begriffsschrift* [Fre79]

Der Komplex der Eigenschaften, englisch *Properties*, Plural von *Property*, Neuronaler Netze scheint zum gegenwärtigen Zeitpunkt ein stark untererforschtes Wissensgebiet darzustellen. Eine – überschaubare – Übersicht über die wenigen gegenwärtig in der Forschungsgemeinschaft diskutierten Eigenschaften Neuronaler Netze, liefern Seshia et al., [SDD+18]. Ob die Aufgabe einer systematischen Untersuchung von Eigenschaften Neuronaler Netze bisher je unternommen worden ist, entzieht sich der Kenntnis des Autors der vorliegenden Arbeit. Skizzen zu einer solchen Unternehmung finden sich bei Albarghouthi, [Alb21], die wir an passender Stelle untersuchen werden. Es wird in diesem Kapitel folglich eine weitgehend unkartierte Landschaft betreten werden müssen. Als Kompassersatz wird uns dabei dienen, dass die Formale Sprache, in der wir Eigenschaften formal ausdrücken können müssen, durch den *Marabou*-Verifizierer vorgegeben ist. Die Definition dieser Sprache, war der Gegenstand von Kapitel 5.6. Die Möglichkeiten der Spezifikation von Eigenschaften Neuronaler Netze, die sich nicht im engen Rahmen dieser Formalen Sprache ausdrücken lassen, werden wir nicht untersuchen.

6.1 Eigenschaften von Funktionen

Wir mögen uns erinnern, dass, nach Satz 1, Neuronale Netze Funktionen sind. Eigenschaften Neuronaler Netze sind damit Eigenschaften von Funktionen. Die eingangs zitierte Gleichung $1 + 1 = 2$, die von Whitehead und Russell in den *Principia Mathematica* aufwendig bewiesen wurde, ist uns bei dieser Gelegenheit, als Beispiel einer Funktioneneigenschaft, nützlich. Der Beweis von Russell und Whitehead ist nämlich zugleich auch ein Beweis, dass die Additionsfunktion die Eigenschaft hat, bei einem eins-wertigen ersten Funktionsargument und einem eins-wertigen zweiten Funktionsargument den Wert Zwei anzunehmen. Dieser Sachverhalt lässt sich weiter formalisieren, wenn wir die Additionsfunktion, genau wie die Neuronalen

Netz-Funktionen, als vektor- oder tupelwertige Funktion verstehen, mit der naheliegenden Interpretation des eindimensionalen Vektor-Wertes der Funktion als der Summe der Vektorelemente der Eingabe, wobei wir die Präfixnotation der Addition wählen: $+((x_1, x_2)^\top) = y$, mit $+\colon \mathbb{R}^2 \to \mathbb{R}$. Um einen kurzen Ausdruck zu haben, wählen wir die Bezeichnung $f_{\text{add}} := +((x_1, x_2)^\top)$ für diese Variante der Additionsfunktion.

Die obige Whitehead-Russell-Eigenschaft – die $(1, 1)^\top$-Eingabe wird auf die 2-Ausgabe abgebildet – lässt sich verallgemeinern als eine Eigenschaft, die prinzipiell allen Funktionen $f\colon \mathbb{R}^2 \to \mathbb{R}$ mit $(x_1, x_2)^\top \mapsto y$, zukommen oder auch nicht zukommen kann. Formal können wir dieses „Zukommen“ auch dergestalt ausdrücken, dass die Eigenschaft P (wie *Property*), in Bezug auf eine Funktion f einen Satz darstellt, der wahr ist, wenn f die Eigenschaft P besitzt, sonst aber falsch. Die Behauptung, dass die Funktion f die Eigenschaft P hat, können wir – in einer ersten Näherung, die wir später mit Bezug zu Eigenschaften Neuronaler Netze präzisieren werden – formal schreiben als $P(f)$. In der Tat können wir nun, mit Frege im Eingangszitat, den Ausdruck $P(f)$ lesen als „Die Funktion f hat die Eigenschaft P.“ und dabei beobachten, dass sowohl f eine Funktion verschiedener Ps ist, denn eine Funktion f hat im Normalfall eine Reihe verschiedener Eigenschaften P , als auch, dass P eine Funktion über verschiedene fs ist, da, wie eben festgestellt, die selbe Eigenschaft P mehr als nur einer Funktion f zukommen kann.

In einer ersten Annäherung an die Formalisierung von Funktionseigenschaften, von Funktionen der Art obiger fs, welche von $n > 0$ Elementen des Diskursbereich $\mathbb{D}$ auf $m > 0$ Elemente desselben Bereichs abbilden, wobei wir $\mathbb{B}$ oben als die Menge der booleschen Wahrheitswerte $\{\text{true}, \text{false}\}$ definiert hatten, haben wir somit

$$P\colon (\mathbb{D}^n \to \mathbb{D}^m) \to \mathbb{B}.$$

Eigenschaften P von Funktionen f sind – wie alle Eigenschaften ganz allgemein – also in der Hinsicht lückenhaft, dass der Ausdruck $P(\cdot)$ keine Bedeutung, keinen Wahrheitswert, hat. Erst der mit einer Funktion f gesättigte Satz $P(f)$ ist ein wahrheitswertiger Satz. Die formale Darstellung einer solchen, um eine f-Instanz ergänzungsbedürftigen, Eigenschaft, die Whitehead-Russell-Eigenschaft der Addition, P_{WR}, wäre in erster Näherung diese:

$$\begin{aligned} P_{\text{WR}}(f) := \exists x_1 \exists x_2 \exists y\colon (f \in \{f' \mid f'\colon \mathbb{R}^2 \to \mathbb{R}, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto y\}) \wedge \\ \big((x_1 = 1) \wedge (x_2 = 1) \wedge (y = 2)\big). \end{aligned} \tag{31}$$

Die Eigenschaft P_{WR} kommt auch der Funktion $f_{\text{leftshift}}((x_1, x_2)^\top) := x_1 \cdot 2^{x_2}$ zu, nicht jedoch der Funktion $f_{\text{mul}}((x_1, x_2)^\top) := x_1 \cdot x_2$, wobei für die beiden Funktionen gilt: $f_{\text{leftshift}}, h_{\text{mul}} \in \{h \mid h\colon \mathbb{R}^2 \to \mathbb{R}\}$.

Bevor wir im nächsten Abschnitt die Formalisierung von Eigenschaften Neuronaler Netze im Speziellen untersuchen werden, wollen wir auf einige an der Formalisierung der P_{WR}-Eigenschaft 31 feststellbare Merkmale hinweisen. Die formale Eigenschaftsbescheibung wird mit einer Kaskade von Existenzquantoren „$\exists x_1 \exists x_2 \exists y$“

eingeleitet, welche für die sich anschließenden Protosatz-Teilformeln, alle in ihnen potentiell vorkommenden Verwendungen der Eingabe- und Ausgabeparameter, x_1, x_2 und y, bereits existenziell binden. Es folgt eine Formel, die die Funktion f, den Formalen Parameter von P_{WR}, beschreibt: „$(f \in \{f' \mid f' \colon \mathbb{R}^2 \to \mathbb{R}, (x_1, x_2)^\top \mapsto y\})$“. Zum Schluss folgt die Formel, „$\big((x_1 = 1) \wedge (x_2 = 1) \wedge (y = 2)\big)$“, in welcher der eigentliche „Kern“ der Eigenschaft formuliert ist.

6.2 Albarghouthi-Formalisierung von Eigenschaften Neuronaler Netze

Wenn wir nun die Untersuchung von Eigenschaften von Funktionen im Allgemeinen zu den Eigenschaften von Funktionen, die durch Neuronalen Netze im Speziellen repräsentiert sind, zurücklenken, so werden wir mit der Frage konfrontiert, ob nicht eine andere Herangehensweise an die Formalisierung von Eigenschaften Neuronaler Netze, als die eben – für Funktionen im Allgemeinen – skizzierte, geeigneter wäre. Im Buch *Introduction to Neural Network Verification*, [Alb21], widmet sich dessen Autor, Aws Albarghouthi, im dritten Kapitel dem Problem der Formalisierung von Eigenschaften Neuronaler Netze. Wir wollen, zusammen mit Albarghouthi – der im Vorwort des Buches zumindest selbstbewusst bekundet: „This book offers the first introduction of foundational ideas from automated verification as applied to deep neural networks and deep learning.“ –, davon ausgehen, dass es sich dabei um eine Pionierleistung handelt, die einer gründlichen Untersuchung an dieser Stelle würdig ist. Dieser Abschnitt widmet sich diesem Unterfangen. Alle folgenden Referenzen, sofern nicht explizit anderes angegeben, sind bezüglich des oben angegebenen Werkes von Albarghouthi, [Alb21], insbesondere des dritten Kapitels daraus, zu verstehen.

Die Idee zur Formalisierung von Eigenschaften Neuronaler Netze, wird man vielleicht am besten vor dem Hintergrund der Aussage Albarghouthis verstehen, welche bereits in der Einleitung des Buches zur Sprache kommt:

> [N]eural networks are programs.
>
> — Aws Albarghouthi, *Introduction to Neural Network Verification*, [Alb21]

Die Behauptung leuchtet, *prima facie*, ein: Wie bei einem Computerprogramm, welches zu einer Programmeingabe eine Programmausgabe berechet, versorgt man ein Neuronales Netz mit einem Eingabevektor und liest das berechnete Ergebnis des Netzes am Ausgabevektor ab. Das Gebiet der Programmverifikation trägt auf dem Boden der Arbeiten von Floyd, [Flo67], und Hoare, [Hoa69], bereits seit den 60er Jahren des letzten Jahrhunderts Früchte. Besondere Bedeutung kommt dabei – in Reverenz an den letztgenannten Autor – der sogenannten Hoare-Logik zu, mit deren Hilfe sich Programmeigenschaften zunächst einmal formalisieren lassen, um anschließend deren Vorhandensein oder deren Abwesenheit formal beweisen zu können. Diese Mächtigkeit der Hoare-Logik versucht Albarghouthi für die Verifikation von Eigenschaften Neuronaler Netze nutzbar zu machen.

Für die Beschreibung der Eigenschaften Neuronaler Netze schlägt Albarghouthi vor, etwa die Sprache der Linearen Arithmetik Reeller Zahlen zu verwenden. Diese Sprache umfasst bei ihm indes neben den schwachen Ungleichheitsrelationen „$\leq$“ und „$\geq$“ auch die strengen Ungleichheitsrelationen „$<$“ und „$>$“. Da für zwei Terme t_1 und t_2 gilt $(t_1 < t_2) \equiv \neg(t_1 \geq t_2)$ und $(t_1 > t_2) \equiv \neg(t_1 \leq t_2)$, verfügt diese Sprache somit neben dem Existenzquantor und der Konjunktion implizit auch über die Negation. Dies ist an dieser Stelle bemerkenswert, da sich – wobei wir hier nicht den Raum haben, diese Behauptung zu beweisen und nur auf den (lies: *not-exists-and*) „*nextand*“-Operator, welcher belegt, dass ein einziges Logisches Zeichen für die Prädikatenlogik Erster Stufe letztlich hinreicht, im Artikel „Combinatory Logic“, [Bim20], der *Stanford Encyclopedia of Philosophy* verweisen können – aus der Menge $\{\neg, \exists, \wedge\}$ Logischer Zeichen alle anderen Logischen Zeichen redefinieren lassen. Folgerichtig gehört zu der von Albarghouthi ins Auge gefassten Sprache der volle, uneingeschränkte Satz der Logischen Zeichen.

Die zur Untersuchung stehenden Eigenschaften sind nach Albarghouthi, *bipartit*, zweigeteilt, zu formulieren. Dazu sei wie bisher ein Neuronales Netz gegeben $N\colon (x_1, \ldots, x_n)^\top \mapsto (y_1, \ldots, y_m)^\top$, wobei $n, m > 0$, bei welchem $(x_1, \ldots, x_n)^\top$ die abstrakten Eingabeparameter und $(y_1, \ldots, y_m)^\top$ die abstrakten, von N berechneten, Werte zur Eingabe sind – das heißt $(x_1, \ldots, x_n)^\top$ und $(y_1, \ldots, y_m)^\top$ haben keinen konkreten Wert und vermögen zunächst einmal jeden möglichen Wert anzunehmen, können aber, wie wir gleich sehen werden, in ihrer Allgemeinheit prinzipiell beschränkt werden. Im initialen „Ausgangszustand“ lassen sich in den sogenannten *Preconditions*, den Bedingungen vor der „Ausführung“ der Netzwerkfunktion von N, Beschränkungen der Eingabe und, wohlgemerkt, nur der Eingabe, also den Komponenten von $(x_1, \ldots, x_n)^\top$, formulieren: Etwa die Beschränkung $(x_1 \leq 2 \cdot x_3) \wedge (-4.5 \cdot x_6 + -7 \cdot x_8 > x_9)$. Damit ist der erste Teil der Eigenschaft beschrieben. Nun wird die Funktion N ausgeführt und der Rückgabewert festgehalten: $(y_1, \ldots, y_m)^\top \leftarrow N(\,(x_1, \ldots, x_n)^\top\,)$. Im zweiten Teil der Eigenschaftsbeschreibung können anschließend im Rahmen der sogenannten *Postconditions* Bedingungen an die Elemente des Ausgabevektors und wiederum des Eingabevektors formuliert werden. Schematisch veranschaulicht Albarghouthi dies so.

$$\{\ \text{precondition}\ \} \quad (y_1, \ldots, y_m)^\top \leftarrow N(\,(x_1, \ldots, x_n)^\top\,) \quad \{\ \text{postcondition}\ \}$$

Wenn die Neuronale Netzwerkfunktion N ein Programm ist, dann ist dieses Vorgehen durchaus stimmig und steht im Einklang mit der gängigen Praxis der Programmverifikation. Eine interessierende Eigenschaft der Programmverifikation, welche durch dieses Schema gut eingefangen wird, lautet etwa: Wenn das System vorher in einem sicheren Zustand war – was das genau bedeutet, ist in den *Preconditions* spezifiziert –, dann ist das System auch nach der Ausführung des Programms – hier $N(\,(x_1, \ldots, x_n)^\top\,)$ – in einem sicheren Zustand – die genaue Bedeutung was dies heißt, wird in den *Postconditions* ausgeführt. Wenn das Programm diese Eigenschaft hat, ist es sicher, sonst potentiell unsicher.

Noch ist indes nicht einsichtig, wie dieses Schema bei der Formulierung von Eigenschaften Verwendung finden kann. Wir wissen noch nicht einmal, was das Schema bedeutet oder wie daraus ein wahrheitswertfähiger Satz entstehen kann. Albarghouthi hilft aus.

> The way to read a specification, informally, is as follows: for any values of $x_1, \ldots, x_n$ that make the precondition true, let $[r = N(\,(x_1, \ldots, x_n)^\top\,)]$ [...]. Then the postcondition is true.
>
> — Aws Albarghouthi, *Introduction to Neural Network Verification*, [Alb21]

Es ergibt sich aus dieser Formulierung das Problem, dass der enthaltene „Let"-Ausdruck einen Imperativ darstellt, der – *qua* Imperativ – nicht wahrheitswertfähig ist. Wir halten die Netzwerkformel 21 bereit und verbleiben inquisitiv. Albarghouthi präzisiert wie folgt.

> The formula that we generate [...] looks roughly like this:
>
> $$(\text{precondition} \wedge [F_N]) \Rightarrow \text{postcondition}$$
>
> If this formula is valid, then the correctness property holds.
>
> — Aws Albarghouthi, *Introduction to Neural Network Verification*, [Alb21]

Wäre die Formel $(\text{precondition} \wedge F_N) \Rightarrow \text{postcondition}$, wie im Zitat behauptet, universell gültig, englisch *valid*, so wäre sie eine Tautologie und nicht informativer als der Satz „Wenn P, dann P", $(P \Rightarrow P)$, für beliebige Propositionen P. Die Aussage, dass jene Formel *wahr* ist, ist hingegen durchaus informativ, da sie uns über das Vorhandensein einer möglicherweise nicht trivialen „*correctness*"-Eigenschaft von Netzwerk N in Kenntnis setzt. Auch gibt es im Kontext von Theorien, in welchen – *qua* Theorien – die Interpretation nichtlogischer Symbole fix ist und nicht zur Dispositionsmenge verschiedener modellerzeugender Interpretationen, vor denen der Satz zuweilen wahr oder falsch sein kann – oder im Fall der universellen Gültigkeit stets wahr sein muss –, zählt, keinen Raum für alternative Modelle neben den Modellen, die die Theorie induzieren. Was Albarghouthi folglich meint, ist wohl *die Wahrheit* des Satzes und nicht seine universelle Gültigkeit: Wenn der Satz $(\text{precondition} \wedge F_N) \Rightarrow \text{postcondition}$ wahr ist, dann und nur dann hat das Netz N die durch precondition und postcondition beschriebene Eigenschaft.

Da Albarghouthi – insbesondere was eine explizite Definition einer solche Spezifikation angeht – keine formalere Aufklärung zur Spezifikation von Eigenschaften Neuronaler Netze anbietet, werden wir für die hiesige Untersuchung mit diesen wenigen Bruchstücken vorlieb nehmen müssen. Setzen wir die wesentlichen Informationen aus beiden Zitaten zusammen. Dabei fällt vielleicht zuerst auf, dass die Formel $((\text{precondition} \wedge F_N) \Rightarrow \text{postcondition})$ ein Proto-Satz ist, da alle Variablen ungebunden sind. Das „for any values of $x_1, \ldots, x_n$" des ersten Zitats legt – nach dem Prinzip der fairsten Interpretationsart, englisch *Principle of Charity* – nahe,

dass der Allabschluss der Formel $((\text{precondition} \wedge F_N) \Rightarrow \text{postcondition})$ insgesamt gemeint sein könnte. Damit haben wir, dass, nach Albarghouthi, das Netz N, mit dem n-elementigen Eingabevektor $(x_1, \ldots, x_n)^\top$ und dem m-elementigen Ausgabevektor $(y_1, \ldots, y_m)^\top$, die durch precondition und postcondition spezifizierte Eigenschaft hat genau dann, wenn folgender Satz wahr ist.

$$\forall x_1 \ldots \forall x_n \forall y_1 \ldots \forall y_m \colon (\text{precondition} \wedge F_N) \Rightarrow \text{postcondition} \tag{32}$$

Die Wahrheitsfunktion eines Konditionals $(F \Rightarrow G)$, mit Teilformeln F und G, wertet zu wahr aus dann und nur dann, wenn F falsch, also $\neg F$ wahr, ist oder wenn G wahr ist. Die Formel $(\text{precondition} \wedge F_N) \Rightarrow \text{postcondition}$ ist, nach *D*e Morgan, also äquivalent zu $(\neg\text{precondition} \vee \text{postcondition}) \vee \neg F_N$. Es ist gegenüber Albarghouthi sicher nicht unfair zu konkretisieren, dass die Eigenschaft P durch den Proto-Satz $P := (\neg\text{precondition} \vee \text{postcondition})$ – der ja dasselbe sagt wie $(\text{precondition} \Rightarrow \text{postcondition})$ – spezifiziert ist.

Mit Albarghouthi zu sagen, dass das Netz N die Eigenschaft P, genau dann *hat*, wenn Formel 32 wahr ist, ist demnach gleichbedeutend damit, dass N die durch P beschriebene Eigenschaft *nicht hat* genau dann, wenn folgender Satz – die Negation von Satz 32 – wahr ist.

$$\neg\forall x_1 \ldots \forall x_n \forall y_1 \ldots \forall y_m \colon (\text{precondition} \wedge F_N) \Rightarrow \text{postcondition} \tag{33}$$

Vor dem Hintergrund der Semantik des Konditionals, der Wahrheitsfunktion „⇒“ – nach welcher ein Satz $(F \Rightarrow G)$, mit Teilformeln F und G, falsch ist dann und nur dann, wenn F wahr aber G falsch, das heißt, $\neg G$ wahr, ist – sowie im Hinblick auf die zweite Äquivalenz von Formel 23, ist der durch Formel 33 beschriebene Satz äquivalent zu diesem

$$\exists x_1 \ldots \exists x_n \exists y_1 \ldots \exists y_m \colon (\text{precondition} \wedge F_N) \wedge \neg\text{postcondition}$$

oder, was das Gleiche ist, diesem

$$\exists x_1 \ldots \exists x_n \exists y_1 \ldots \exists y_m \colon F_N \wedge (\text{precondition} \wedge \neg\text{postcondition}). \tag{34}$$

Wir hatten definiert, dass $P \equiv (\neg\text{precondition} \vee \text{postcondition})$, wonach also, mit *De Morgan* $\neg P \equiv \neg(\neg\text{precondition} \vee \text{postcondition}) \equiv (\text{precondition} \wedge \neg\text{postcondition})$. Womit Satz 34 äquivalent ist zu

$$\exists x_1 \ldots \exists x_n \exists y_1 \ldots \exists y_m \colon F_N \wedge \neg P. \tag{35}$$

Zusammengefasst ist der Satz 35 demnach wahr genau dann, wenn N nicht die Eigenschaft P hat. In etwas forcierter Weise können wir sagen, dass das Neuronale Netz die *Un*-Eigenschaft $\neg P$ hat, genau dann, wenn Satz 35 wahr ist. Es ist dabei der erwartete Abschluss des Proto-Satzes, welchen die quantorenfreie Eigenschaftsformel P darstellt, so, wie in Definition 26, einzubeziehen.

Definition 26. *Ein Neuronales Netz N hat die* Un-Eigenschaft*, oder* Uneigenschaft*, $\neg P$, genau dann, wenn, erstens, $\neg P$ eine Teilformel darstellt, die prospektiv universell abgeschlossen wird und N nicht die Eigenschaft P hat, die prospektiv existentiell abgeschlossen wird oder, zweitens $\neg P$ eine Teilformel darstellt, die prospektiv existentiell abgeschlossen wird und N nicht die Eigenschaft P hat, die prospektiv universell abgeschlossen wird.*

Man beachte dabei insbesondere, dass „Es ist nicht der Fall, dass N die Eigenschaft P hat." respektive, was dasselbe sagt, „N hat nicht die Eigenschaft P." *nicht* äquivalent ist zu „N hat die Eigenschaft $\neg P$". Abschließend ist also, nach Albarghouthi, der durch Formel 35 ausgedrückte Satz wahr dann und nur dann, wenn das Netz N – welches durch die Netzwerkformel F_N vollumfänglich beschrieben ist – die durch $\neg P$ ausgedrückte Uneigenschaft hat.

Es ist schwer, die Bedeutung der Formel 35 überzubetonen. Wenn es uns gelingt, eine interessierende Eigenschaft P so zu formulieren, dass, deren Negation ein Ausdruck in der Sprache *Marabou*s darstellt, so ist es im Sinne Albarghouthis in der Sprache *Marabou*s möglich, den Satz zu formulieren, dass ein Neuronales Netz N, die Eigenschaft P hat. Eine solche Eigenschaft könnte etwa die Eigenschaft $P :\equiv ((x \leq 2) \Rightarrow (y > 3))$ sein. Die dem entsprechende Negation, die Uneigenschaft, $\neg P \equiv ((x \leq 2) \wedge (y \leq 3))$ ist ein Proto-Satz der Sprache *Marabou*s. Wäre nun für dieses P der Satz 35 *falsch*, so wüssten wir, dass das Netz N die Eigenschaft P hat.

Es sollen hier aber auch kritische Punkte der Albarghouthi-Formalisierung von Eigenschaften Neuronaler Netze nicht unangesprochen bleiben. Dazu ist nicht zuletzt das Auffahren des Apparates der Hoare-Logik selbst in Frage zu stellen. Ist – denn das scheint der Hauptgrund dieses Auffahrens – ein Neuronales Netz, wie Albarghouthi meint, am besten als ein *Programm* beschrieben, welches als solches einer Programm-Verifikation zugänglich ist? Nehmen wir an, es sei dem so. Ist die Additionsfunktion der Natürlichen Zahlen ein Programm? Dann haben – bisher unbemerkt von Generationen von Geschichtswissenschaftlern – vielleicht die Alten Babylonier, Jahrtausende vor Ada Lovelace, das erste Computerprogramm der Weltgeschichte in Ton gedrückt, und die Wissenschaft der Informatik wäre wohl älter als die Wissenschaft der Astronomie. Möglicherweise konnten dann die Chinesen der Shang-Dynastie in einer Hochsprache programmieren, noch bevor, am anderen Ende des asiatischen Kontinents, überhaupt erst das Alphabet erfunden wurde, und dies zudem tausende Jahre bevor Grace Hopper *ihr* erstes Programm in einer Hochsprache schrieb. Die Maya haben womöglich mehr Zeilen Programmcode in Stein gehauen als Margaret Elaine Hamilton zusammen mit dem Team vom *MIT* später für das Apolloprogramm der *NASA* schrieb – ausgedruckt immerhin ein Papierstapel fast größer als sie selbst. Das Neuronale Netz mit zwei Eingängen x_1, x_2 und einem einzigen Neuronenknoten $N := \mathit{ReLU}(w_1 x_1 + w_2 x_2 + b)$, bei dem $w_1 = w_2 := 1$ und $b := 0$ gewählt wird, *ist* diese Additionsfunktion, und *qua* Neuronales Netz, nach Albarghouthi, ein Programm. Auch größere Neuronale Netze sind bloß Kompositionen aus Funktionen dieser oder ähnlicher Art. Kann denn in Frage stehen, dass ein solches „Programm" terminiert? Vielleicht sind auch, mithilfe

der Additionsfunktion beschreibbare, Zahlen Programme und die Eulersche Zahl $e := \sum_{k=0}^{\infty}(1/k!)$ terminiert nicht, da fortwährend neue Summanden hinzukommen. Kann hier die Hoare-Logik aushelfen? Was – und wo – sind die Programmzeilen, die das „Programm“ Additionsfunktion durchläuft, um zum Ergebnis zu gelangen? Selbst wenn diese und ähnliche Bedenken zufriedenstellend beantwortet würden, wodurch der Einsatz der Hoare-Logik gerechtfertigt erscheinen könnte, so bliebe die Frage, weshalb Albarghouthi letztendlich keinen Gebrauch von ihr zu machen scheint. Wie wir gesehen hatten, kulminiert die Formalisierung des Satzes, dass das Netz N die Eigenschaft P hat, bei Albarghouthi in der Formel 35. Diese Formel kommt aber mit der Prädikatenlogik Erster Stufe allein – genauer sogar lediglich mit einem Teil derselben – aus und bedarf der Hoare-Logik an keiner Stelle.

Das Ergebnis obiger destruktiver *Reductio-ad-Absurdo*-Argumentation – der vorgetragenen Art: „Wer die Prämisse (Neuronale Netze sind Programme) behauptet, behauptet auch, was (absurderweise) daraus folgt.“, durch welche insinuiert werden soll, die Prämisse sei potentiell defizitär; ein Argumentationsmuster mit Tücken, wovon die Wissenschaftsgeschichte an dem oft kolportierten Fall Einsteins, dessen, aus der Theorie der Quantenmechanik als mutmaßlich offensichtlich absurd hergeleiteter, Sachverhalt, dass Gott dann ein mit dem Universum Würfel Spielender sein müsste, später, in Gestalt der gängigen Interpretation der sogenannten „Kopenhagener Deutung der Quantenmechanik“, als Sachverhalt der Existenz „echten“, ontologischen Zufalls, im Unterschied zu bloß, prominent durch Pierre-Simon Laplace, etwa in Form des berühmten „Laplaceschen Dämons“, vertretenen, epistemischem Zufall, als tatsächlich bestehend bestätigt wurde, ein Beispiel gibt –, konstruktiv wendend, sollten wir hier unterscheiden zwischen *einer berechenbaren Funktion* und der *Berechnung dieser Funktion*, durch einen Algorithmus – „ein Programm“, sofern zusätzlich ein Berechnungsmodell spezifiziert wird, welches den Programmcode auf eine klar definierte Weise auszuführen, zumindest theoretisch, etwa als Turingmaschine, in der Lage ist. Ein Neuronales Netz ist zunächst einmal – wie die Additionsfunktion der Natürlichen Zahlen – eine Funktion. Diese Neuronale-Netzwerk-Funktion, genauererweise: eine Approximation dieser Funktion, kann berechnet werden, durch eine konkrete Implementierung des Netzwerks in Software – oder, seltener, in Hardware oder einer Mischung aus beidem. Da Albarghouthi – ebenso wie wir – die Spezifikation von Eigenschaften Neuronaler Netze, für seinen Teil unter nicht fruchtbringender vermeintlicher Zuhilfenahme der Hoare-Logik, aber auf die Eigenschaften der berechenbaren *Funktion* und nicht – bei der Formulierung welcher die Verwendung der Hoare-Logik durchaus fertil erscheinen mag – die Eigenschaften bei der *Berechnung* der Funktion bezieht, ist obige *Reductio*-Argumentation wohl zumindest nicht als unfair zu betrachten.

Was indes noch unglücklicher ist als die Redundanz der Hoare-Logik bei der Formulierung der Eigenschaften Neuronaler Netze, ist die durch jene eingeführte mutwillige Beschneidung der Ausdruckskraft der möglichen zu untersuchenden Eigenschaften P durch den Zwang dieser ihrer Aufteilung in $\mathrm{preconditions}$ und $\mathrm{postconditions}$. Und, um Schlechtem Schlimmes hinzuzufügen, sind dabei die

$\mathrm{preconditions}$ unnötigerweise darauf eingezwängt, ausschließlich lineare Beschränkungen über Variablen des Eingabevektors zu erlauben. Doch auch die Aufhebung dieser Beschränkung bei der Formulierung der $\mathrm{postconditions}$ trägt dem Umstand noch keine Rechnung, dass ein Neuronales Netz, im Regelfall, mehr Knoten als die Knoten der Ein- und Ausgabeschicht umfasst. Warum sollten nicht auch Eigenschaften, die die „inneren" Knoten involvieren, formulierbar sein? „Kein Post-Knoten des vorletzten *Hidden Layers* des Netzes N nimmt jemals einen Wert größer als 100 an." könnte durchaus eine interessierende Eigenschaft des Netzes N sein, welche in der Albarghouthi-Formalisierung von Eigenschaften nicht ausdrückbar ist.

Auf der anderen Seite ist die von Albarghouthi vorgeschlagene Formalisierung zu mächtig. In der Sprache Marabous verfügen wir, anders als in der Sprache, die Albarghouthi für die Formalisierung zu Grunde legt, nicht über die Negation und im Speziellen nicht über die starken Ungleichheitsrelationen, die durch „$<$" und „$>$" ausgedrückt werden.

Ein letzter Kritikpunkt, der uns zum nächsten Abschnitt überleitet, ist eher eine ästhetische Frage. Anstatt umständlich zu sagen, dass ein Neuronales Netz N die Eigenschaft P hat genau dann, wenn es nicht die Uneigenschaft $\neg P$ hat genau dann, wenn Satz 35 falsch ist, erscheint es natürlicher, das Folgende zu sagen.

Definition 27. *Ein Neuronales Netz N mit der Formalisierung F_N und der Knotenvariablenmenge $\{v_1, \ldots, v_\ell\}$ hat die Eigenschaft P genau dann, wenn der Satz*

$$\exists v_1 \ldots \exists v_\ell \colon F_N \wedge P. \tag{36}$$

wahr ist.

Wir betonen zunächst, dass ein wesentlicher Unterschied zur Albarghouthikonzeption darin besteht, dass *alle* Knotenvariablen des Neuronalen Netzes und nicht nur die Variablen des Ein- und Ausgabevektors existenzquantifiziert werden, was *ipso facto* eine Expressivitätserweiterung mit Blick auf die formulierbaren Eigenschaften darstellt, da $\{v_1, \ldots, v_\ell\} \supseteq \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$. Man beachte bei Definition 27 nun aber, dass damit die Umdeutung der Anwesenheit einer Uneigenschaft $\neg P$ in die Anwesenheit einer Eigenschaft $\neg P$ einhergeht. Konkret wird das „N hat nicht P.", in Albarghouthis Sinne, zu „N hat $\neg P$." in unserem Sinne folgendermaßen umgedeutet, wobei das „gdw." für „genau dann, wenn" und das „vs." für *versus*, „im Gegensatz dazu", steht.

$$\overbrace{\left(\neg(N \text{ hat } P) \text{ gdw. } \exists v_1 \ldots \exists v_n \colon F_N \wedge \neg P\right)}^{\text{Albarghouthi-Konzeption, F. 35}} \text{ vs. } \overbrace{\left((N \text{ hat } \neg P) \text{ gdw. } \exists v_1 \ldots \exists v_n \colon F_N \wedge \neg P\right)}^{\text{unsere Konzeption, F. 27}}$$

Was hier als trivialerweise gültige Transformation anmutet, ist indes keine logische Äquivalenzumformung, wie wir wie folgt, im Hinblick auf die Negation der Formel 35 mit vertauschtem P und $\neg P$, sehen.

$$\overbrace{\left(\neg(N \text{ hat } P) \text{ gdw. } \exists v_1 \ldots \exists v_n \colon F_N \wedge \neg P\right)}^{\text{Albarghouthi-Konzeption, F. 35}} \text{ vs. } \overbrace{\left((N \text{ hat } \neg P) \text{ gdw. } \neg\exists v_1 \ldots \exists v_n \colon F_N \wedge P\right)}^{\text{Albarghouthi-Konzeption, Neg. v. F. 35}}$$

$$\overbrace{(\exists v_1 \ldots \exists v_n \colon F_N \wedge \neg P)}^{\text{gdw. } \neg(N \text{ hat } P)} \not\equiv \overbrace{(\neg\exists v_1 \ldots \exists v_n \colon F_N \wedge P)}^{\text{gdw. } (N \text{ hat } \neg P)}$$

Praktisch hat diese leicht unterschiedliche Eigenschaftskonzeption wenig Auswirkungen: Werden bei Albarghouthi die Eigenschaften als universell und die Uneigenschaften existenziell abgeschlossen angenommen, so sind in unserer Konzeption Eigenschaften implizit existenzquantifiziert und Uneigenschaften implizit allquantifiziert.

6.3 Formalisierung von Netzeigenschaften mittels der Sprache Marabous

Die Formalisierung von Eigenschaften P Neuronaler Netze N in der Sprache *Marabou*s, welche in dieser Weise in der Verifikationssoftware für Eigenschaften Neuronaler Netze *Marabou* ziemlich genau so implementiert ist, ist nun gerade die in Formel 36 der Definition 27 angegebene. Konkret bietet das Verifikationsprogramm *Marabou* grob folgenden Dienstumfang. Es erwartet einen Verweis auf die ein spezifisches Netzwerk N beschreibenden Daten. Im einfachsten Fall ist dies ein übliches Neuronales-Netzwerk-Dateiformat, in welchem das Netz ohnehin auf einer Rechenanlage im Sekundärspeicher liegt. Mittels der in dieser Datei enthaltenen Information – über die dem Netz eigene Anzahl an Schichten, der konkreten Bias- und Kantengewichtswerte und so weiter –, erzeugt *Marabou* die Netzwerkformel F_N, welche wir als Formel 21 hergeleitet hatten. Um verifizieren zu können, ob das gegebene Netz N eine Eigenschaft P hat oder nicht, benötigt *Marabou* des Weiteren eine Beschreibung dieser Eigenschaft. Eine Eigenschaft P ist dabei ein Proto-Satz der Sprache *Marabou*s, über welchen später, in Konjunktion mit der Netzwerkformel F_N, der Existenzabschluss gebildet wird. Für die Spezifikation dieser Eigenschaft P steht einer *Marabou* als Verifizierer verwendenden Person die Sprache *Marabou*s zur Verfügung. Nach der Komposition einer Eigenschaftsspezifikation P kann *Marabou* nun auch P übergeben werden, woraufhin *Marabou* etwas der Formel 36 Entsprechendes erzeugen und mit dem Verifizieren beginnen wird. *Wie* dieser Verifikationsprozess *Marabou*s vonstattengeht, werden wir in Kapitel 7 genauer untersuchen.

Für den Moment genügt es uns, zu wissen, dass *Marabou* im Falle, dass das Netz N die Eigenschaft P hat, als Ergebnis des Verifikationsprozesses die „Antwort“ SAT zurückgibt, was signalisiert, dass die Formel 36 für das gegebene Netz N und die gegebenen Eigenschaft P wahr ist. In diesem Fall gibt *Marabou* desweiteren – sozusagen als Beweis der Wahrheit der in Formel 36 ausgedrückten Existenzbehauptung, welche durch eine einzige Instanz bewiesen werden kann – ein Beispiel, in Form einer vollständigen Belegung aller Variablen des Netzwerks, für die der durch Formel 36 ausgedrückte Satz wahr ist. Hat *Marabou* andernfalls festgestellt, dass N nicht die Eigenschaft P hat, was gleichbedeutend damit ist, dass der durch Formel 36 ausgedrückte Satz für das gegebene N und P falsch ist, so signalisiert es dies, ohne nähere Begründung, durch die schlichte Rückgabe von UNSAT.

Wir wollen eine einfache Eigenschaft für den in Abschnitt 2.5 vorgestellten Klassifizierer N, welcher in Abbildung 6 dargestellt ist, spezifizieren. Da wir vermuten, dass N die Eigenschaft hat, jedes Eingabebild, für das der erste Pixel nicht dunkler ist als der zweite, formal $(x_2 \geq x_1)$, nicht als $c = 1$ zu klassifizieren, also $(y_2 > y_1)$, könnten wir geneigt sein, *Marabou* zu fragen, ob N tatsächlich diese Eigenschaft hat. Dazu müssen wir diese Eigenschaft aber zunächst einmal in der Sprache *Marabou*s reformulieren und werden mit einem Problem konfrontiert: Die entsprechende Formalisierung P'

$$(\forall x_1 \forall x_2 \forall y_1 \forall y_2 \colon) \overbrace{((x_2 \geq x_1) \Rightarrow (y_2 > y_1))}^{:= P'}$$

enthält das in der Sprache *QFLRA* nicht definierte materiale Konditional „$\Rightarrow$“ und ist darüber hinausgehend implizit allquantifiziert und daher so nicht in der Sprache *Marabou*s ausdrückbar. Allerdings lässt sich die implizit existenzquantifizierte Eigenschaft $P := \neg P'$, als Negation des Konditionals, bei welcher der Vordersatz wahr aber der Nachsatz falsch, seine Negation wahr, ist, durchaus wie folgt in der Sprache *Marabou*s ausdrücken.

$$(\exists x_1 \exists x_2 \exists y_1 \exists y_2 \colon) \overbrace{((x_2 \geq x_1) \wedge (y_2 \leq y_1))}^{:= P}$$

Damit haben wir eine Eigenschaft P spezifiziert und können diese zusammen mit dem Netzwerk N nun *Marabou* zur Verifikation übergeben. Antwortet *Marabou* mit UNSAT, so wissen wir, dass N die Uneigenschaft P' hat, da es die Eigenschaft P nicht hat. Antwortet es hingegen SAT, so können wir schließen, dass N die Uneigenschaft P' nicht hat, da es ja die Eigenschaft P hat. Dass N die Uneigenschaft P' tatsächlich nicht besitzt, ist von *Marabou* in diesem Fall durch ein Beispiel, etwa $[x_1 \leftarrow 0, x_2 \leftarrow 0, y_1 \leftarrow 0, y_2 \leftarrow 0]$, gezeigt – eine allquantifizierte Aussage wird durch ein einziges Gegenbeispiel falsifiziert –, da N die $(x_1, x_2) = (0, 0)$-Eingabe auf die Ausgabe $(y_1, y_2) = (0, 0)$ abbildet, was insbesondere impliziert, dass $(x_2 \geq x_1) \wedge (y_2 \leq y_1))$. Bei der Annahme, dass das Netz alle Eingaben, bei denen $(x_2 \geq x_1)$ gegeben ist, nicht als $c = 1$ klassifiziert, haben wir möglicherweise den Fall übersehen, dass alle Eingaben, bei denen $(x_1 = x_2)$, und somit $(y_1 = y_2)$, gilt, durch das Netz zwar durchaus als $c = 2$ klassifiziert angenommen werden können, da $(y_2 \geq y_1)$, aber eben auch, da $(y_1 \geq y_2)$, als $c = 1$ klassifiziert gedacht werden können.

Eine in der Praxis häufig interessantere Eigenschaft ist die Eigenschaft der *Mindestrobustheit* des Netzes N mit Bezug auf ein festes Eingabebild $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2)^\top$ auf die wir nun zu sprechen kommen wollen. Anschaulich gesprochen handelt es sich dabei um die Frage, wie stark man die Eingabeelemente, die Grauwerte der Pixel $\hat{x}_1$ und $\hat{x}_2$, ändern müsste, damit sich die Klassifikation von N für dieses Bild umkehrt. Die dazu nötige Veränderung der Eingabeelemente kann durch ein Abstandsmaß quantifiziert werden, wobei wir uns im Speziellen stets nur des Tschebyscheff-Distanzmaßes d_∞, welches aus der Maximumsnorm, L^∞, abgeleitet ist, bedienen.

Es ist für Vektoren $\boldsymbol{u} = (u_1, \ldots, u_n)^\top$ und $\boldsymbol{v} = (v_1, \ldots, v_n)^\top$ gleicher Dimension wie folgt definiert.

$$d_\infty(\boldsymbol{u}, \boldsymbol{v}) := \|\boldsymbol{u} - \boldsymbol{v}\|_\infty = \max(|u_1 - v_1|, \ldots, |u_n - v_n|) \tag{37}$$

Die Tschebyscheffdistanz d_∞ ergibt sich demnach als der größte Abstand – gemessen in der L^1-Norm, des Absolutwertes, $|\cdot|$, der Differenz – zweier Vektorkomponentenpaare u_i und v_i als $|u_i - v_i|$, wobei $i \in \{1, \ldots, n\}$.

In Bezug auf die Mindestrobustheit von Bildklassifizierern bezüglich eines Bildes $\hat{\boldsymbol{x}}$ – wohlgemerkt ein konstanter Vektor –, welches vom Netz N als l klassifiziert wird, wobei $l \in \{1, \ldots, m\}$ ein Index eines Elements des Ausgabevektors derart ist, dass $N(\hat{\boldsymbol{x}}) = \hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_m)^\top$ und $\arg\max(\hat{\boldsymbol{y}}) = l$, findet das Tschebyscheffdistanzmaß wie folgt Verwendung. Wenn für alle Eingabevektoren $\boldsymbol{x}$ – ein variabler Vektor dessen Elemente $x_1, \ldots, x_n$ allesamt Variablen sind – und zugehörigen Ausgabevektoren $N(\boldsymbol{x}) = \boldsymbol{y} = (y_1, \ldots, y_m)^\top$ gilt, dass $d_\infty(\hat{\boldsymbol{x}}, \boldsymbol{x}) \leq \epsilon$ und $\arg\max(\boldsymbol{y}) = l$, dann besitzt N eine Mindestrobustheit von ϵ bezüglich Bild $\hat{\boldsymbol{x}}$. Doch wir wollen kleinschrittig vorgehen. Ein konkretes Bild $\hat{\boldsymbol{x}}$ stellt als n-dimensionaler Vektor einen Punkt, eine Koordinate, im n-dimensionalen Raum dar. Zu jedem dieser Koordinaten gehört eine Klassifikation $c = l = \arg\max(N(\hat{\boldsymbol{x}}))$. Betrachten wir, in Abbildung 15, noch einmal unser Beispiel aus Abbildung 7, bei dem $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2)^\top = (0.25, 0.75)$ und $c = \arg\max(N(\hat{\boldsymbol{x}})) = 2$.

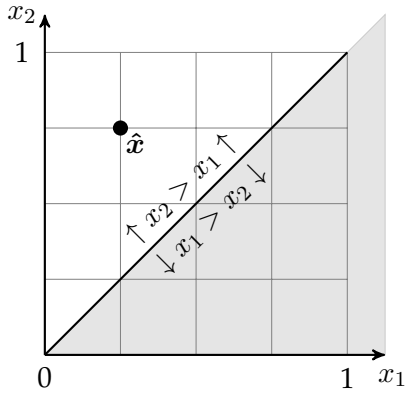


Abbildung 15: Eingabevektoren $\boldsymbol{x}$ mit n Vektorkomponenten stellen Koordinaten im n-Dimensionalen Raum dar. Abgebildet ist der Eingabevektor $(x_1, x_2)^\top = (0.25, 0.75)^\top$ aus Abbildung 7. Alle nichtnegativen Koordinaten auf weißem Untergrund, inklusive der x_2-Achse werden als 2 klassifiziert, alle nichtnegativen Koordinaten auf grauem Untergrund, inklusive der x_1-Achse werden als 1 klassifiziert. Alle Koordinaten auf der $(x_1 = x_2)$-Diagonalen können sowohl als 1 als auch als 2 klassifiziert angenommen werden.

Wenn wir nun, als Eigenschaft der Mindestrobustheit, fordern, dass für alle Bilder $\boldsymbol{x}$ gelten möge, dass $d_\infty(\hat{\boldsymbol{x}}, \boldsymbol{x}) \leq \epsilon$ und dabei für $\boldsymbol{x}$ die ursprüngliche Klassifikation

l gilt, also $\arg\max(\boldsymbol{y}) = l$, so definieren wir einen Bereich um die ursprüngliche Koordinate $\hat{\boldsymbol{x}}$ und fordern dass überall in diesem Bereich nur Koordinaten liegen, die, wie $\hat{\boldsymbol{x}}$, als l klassifiziert werden. Dieser Bereich ist, für alle n-dimensionalen Eingabevektoren, ein n-dimensionaler Hyperwürfel, englisch *hypercube*, in dessen Zentrum die Koordinate $\hat{\boldsymbol{x}}$ liegt und der einen „Radius" von ϵ besitzt, genauer: dessen Kantenlänge 2ϵ ist. Für unser Beispiel aus Abbildung 7 ist dieser Hyperwürfel ein Quadrat. Der entsprechende Sachverhalt ist in Abbildung 16 dargestellt.

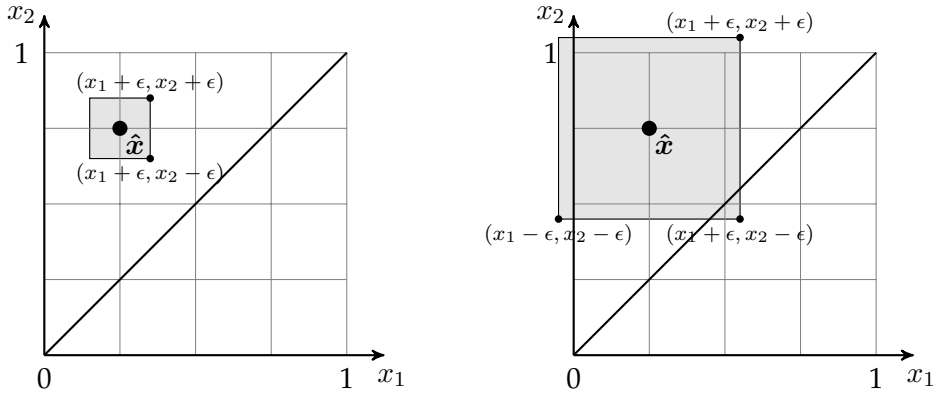


Abbildung 16: Die Eigenschaft der Mindestrobustheit definiert einen n-dimensionalen Hyperwürfel mit Kantenlänge 2ϵ um die n-dimensionale Eingabekoordinate $\hat{\boldsymbol{x}}$. Das Netz N besitzt bezüglich Vektor $\hat{\boldsymbol{x}}$ und bezüglich ϵ die Eigenschaft der Mindestrobustheit, falls alle Koordinaten – inklusive der Kanten – des Hyperwürfels, hier grau dargestellt, durch N dieselbe Klassifikation l (und nur diese) erfahren, wie $\hat{\boldsymbol{x}}$. Links: Abgebildet ist die Situation für $\epsilon = 0.1$. Das Netz N ist für das Bild $\hat{\boldsymbol{x}}$ mindestens so robust wie ϵ, da alle Koordinaten des Hyperwürfels auf der Seite der Diagonale liegen, die als 2 klassifiziert wird. Rechts: Abgebildet ist die Situation für $\epsilon = 0.3$. Das Netz N ist für das Bild $\hat{\boldsymbol{x}}$ nicht robust für dieses ϵ, da beispielsweise die Koordinate $(x_1 + \epsilon, x_2 - \epsilon) = (0.25 + 0.3, 0.75 - 0.3) = (0.55, 0.45)$, welche sich im Hyperwürfel befindet, auf derjenigen Seite der Diagonale liegt, die als 1 klassifiziert wird. Man beachte, dass die Koordinaten $(x_1 + \epsilon, x_2 + \epsilon) = (0.55, 1.05)$ und $(x_1 - \epsilon, x_2 - \epsilon) = (-0.05, 0.45)$ außerhalb der Grenzen für legale Pixelwerte v, nämlich $0 \leq v \leq 1$, liegen.

Um die auf der rechten Seite von Abbildung 16 exemplifizierte Verletzung der Pixelwert-Norm zu remedieren, zu beheben, werden die entsprechenden Größen nach oben zur 1 und nach unten zur 0 begrenzt. Für ein gegebenes ϵ stellt dies für

einen ursprünglichen Pixelwert $\hat{x}_i$ des $\hat{\boldsymbol{x}}$-Vektors die Formalisierung

$$\max(0, (\hat{x}_i - \epsilon)) =: \hat{x}_i^l \;\leq\; x_i \quad \wedge \quad x_i \;\leq\; \hat{x}_i^u := \min((\hat{x}_i + \epsilon), 1) \tag{38}$$

sicher. Man beachte, dass, da sowohl $\hat{x}_i$ als auch ϵ konstante Werte sind, auch die Werte für $\hat{x}_i^l$ und $\hat{x}_i^u$ Konstanten sind, die vor einer möglichen Anfrage an *Marabou* ausgewertet werden können und ausgewertet werden müssen.

Damit können wir nun den Versuch unternehmen, die Eigenschaft der Mindestrobustheit in der Sprache *Marabou*s zu formalisieren. Nach dem bisher Zusammengetragenen, werden wir wieder von einer Uneigenschaft ausgehen müssen, da wir ja ausdrücken wollen, dass *alle* Vektoren $\boldsymbol{x}$, welche höchstens um ϵ Einheiten pro Pixel von $\hat{\boldsymbol{x}}$ nach unten oder oben abweichen, genauso wie $\hat{\boldsymbol{x}}$, nur als l klassifiziert werden. Letzteres ist gleichbedeutend damit, dass der Wert von y_l für jeden so ϵ-veränderten – wir sprechen auch von einem *perturbierten* – Vektor $\boldsymbol{x} = (x_1, \ldots, x_n)^\top$ im Ergebnisvektor $\boldsymbol{y} = N(\boldsymbol{x}) = (y_1, \ldots, y_m)^\top$ einen Wert annimmt, der größer ist als die Werte alle anderen $\boldsymbol{y}$-Vektorelemente, kurz $y_l > y_i$ für alle $i \in \{1, \ldots, m\} \setminus \{l\}$. Die implizit allquantifizierte Mindestrobustheits-Uneigenschaft P' lautet demnach wie folgt.

$$(\forall x_1 \cdots \forall y_m \colon) \Big(\bigwedge_{i \in \{1,\ldots,n\}} \big((\hat{x}_i^l \leq x_i) \wedge (x_i \leq \hat{x}_i^u)\big) \Big) \;\Rightarrow\; \Big(\bigwedge_{j \in \{1,\ldots,m\} \setminus \{l\}} \big(y_l > y_j\big) \Big) \tag{39}$$

Da die Negation von $\bigwedge_{j \in \{1,\ldots,m\} \setminus \{l\}} \big(y_l > y_j\big)$, aufgrund der Gesetze von *De Morgan*, die Formel $\bigvee_{j \in \{1,\ldots,m\} \setminus \{l\}} \big(y_l \leq y_j\big)$ ist, ist die implizit existenzquantifizierte Eigenschaft P zur Uneigenschaft P' diese:

$$(\exists x_1 \cdots \exists y_m \colon) \Big(\bigwedge_{i \in \{1,\ldots,n\}} \big((\hat{x}_i^l \leq x_i) \wedge (x_i \leq \hat{x}_i^u)\big) \Big) \;\wedge\; \Big(\bigvee_{j \in \{1,\ldots,m\} \setminus \{l\}} \big(y_l \leq y_j\big) \Big) \tag{40}$$

Vor einer Verifikationsanfrage an *Marabou* sind die $m - 1$ Disjunkte der Disjunktion $\bigvee_{j \in \{1,\ldots,m\} \setminus \{l\}} \big(y_l \leq y_j\big)$ jeweils einzeln durch eine Konjunktion mit der Formel $\bigwedge_{i \in \{1,\ldots,n\}} \big((\hat{x}_i^l \leq x_i) \wedge (x_i \leq \hat{x}_i^u)\big)$ zu verknüpfen. Die daraus entstehenden $m - 1$ Eigenschaftsformeln P_1 bis P_{m-1} werden jeweils *Marabou* zur Verifikation übergeben. Besitzt das Netz keine der Teileigenschaften P_i, mit $i \in \{i, \ldots, m-1\}$, so besitzt das Netz nicht die Eigenschaft P und folglich hat es die Uneigenschaft der Mindestrobustheit P': Keine Perturbation der Pixel um höchstens ϵ Einheiten führt dazu, dass sich die ursprüngliche Klassifikation mit Bezug auf ein so perturbiertes Bild ändert. Besitzt andererseits das Netz mindestens eine der Teileigenschaften P_i, dann hat das Netz die Eigenschaft P und somit nicht die Uneigenschaft der Mindestrobustheit P'. Die von Marabou in diesem Fall ebenfalls berechnete Belegung aller Variablen, $[x_1 \leftarrow d_1, \ldots, x_n \leftarrow d_n, y_1 \leftarrow d_{n+1}, \ldots, y_m \leftarrow d_{n+m}, \ldots]$, bei der $d_i \in \mathbb{D}$, für alle $i \in \{1, \ldots, m+n, \ldots\}$, erzeugt ein Bild $\hat{\boldsymbol{x}}^{\text{adv}} =: (d_1, \ldots, d_n)^\top$, welches ein Gegebenbeispiel, englisch *adversarial example*, für die Robustheit exemplifiziert:

Das Bild $\hat{\boldsymbol{x}}^{\text{adv}}$ wird höchstens um ϵ Einheiten perturbiert, aber es wird, anders als $\hat{\boldsymbol{x}}$, nicht als l klassifiziert.

Es lässt sich die Disjunktion $\bigvee_{j\in\{1,\dots,m\}\setminus\{l\}}(y_l \leq y_j)$ als Konsequenzteilformel aus Formel 40, auch so interpretieren, dass das pixelweise um höchstens ϵ Einheiten perturbierte Bild $\boldsymbol{x}$ in *keine* andere Klasse fallen soll als l. Zuweilen ist aber auch die Eigenschaft interessant, dass – *ceteris paribus*, „alles andere bleibt gleich" – das perturbierte Bild $\boldsymbol{x}$ *nicht als eine spezifische Klasse* und zwar die Zielklasse t, für englisch *target*, mit $t, l \in \{1,\dots,m\}$ und $t \neq l$, klassifiziert wird. Anschaulich gesprochen dürfen sich im Hyperwürfel, im Zentrum von welchem $\hat{\boldsymbol{x}}$ steht, Vektoren aller möglichen Klassifikationen befinden, insbesondere nicht nur der ursprünglichen Klassifikation l – solange diese Klassifikation nicht die Klasse t ist. Letzeres findet formal seinen Ausdruck darin, dass mindestens ein Vektorelement der Ausgabe mit von t verschiedenem Index größer als y_t sein muss. Die entsprechende Uneigenschaft ist folglich diese

$$(\forall x_1 \cdots \forall y_m :)\Big(\bigwedge_{i\in\{1,\dots,n\}} \big((\hat{x}_i^l \leq x_i) \wedge (x_i \leq \hat{x}_i^u)\big)\Big) \Rightarrow \Big(\bigvee_{j\in\{1,\dots,m\}\setminus\{t\}} (y_j > y_t)\Big) \quad (41)$$

und die Eigenschaft entsprechend folgende

$$(\exists x_1 \cdots \exists y_m :)\Big(\bigwedge_{i\in\{1,\dots,n\}} \big((\hat{x}_i^l \leq x_i) \wedge (x_i \leq \hat{x}_i^u)\big)\Big) \wedge \Big(\bigwedge_{j\in\{1,\dots,m\}\setminus\{t\}} (y_j \leq y_t)\Big). \quad (42)$$

Man beachte, dass auch die Formel 41 ein Ausdruck für die Mindestrobustheit von Netz N bezüglich des Bildes $\hat{\boldsymbol{x}}$ ist, nämlich der Mindestrobustheit dieses Bildes einer Klassifikation als t zu widerstehen. Wir haben damit zwei – in der Formel 39 respektive Formel 41 spezifizierte – verschiedene Ausdrücke der Mindestrobustheit. In der vorliegenden Arbeit werden wir uns auf die Letztere, welche in Formel 42 ihren Ausdruck als Eigenschaft, im in Definition 27 angegebenen Sinne, findet, der beiden fokussieren, wenngleich erstere die verbreitetere sein dürfte. Für unseren Klassifizierer aus Abbildung 6, der nur zwei Klassen – l und t, falls $t \neq l$ – kennt, sind die in Formel 40 und die in Formel 42 spezifizierten Eigenschaften zwar dieselben, dies gilt jedoch nicht mehr für *MNIST*-Klassifizierer mit ihren zehn verschiedenen Ziffern-Klassen.

Bisher haben wir nur von *Mindest*-Robustheit gesprochen. Nicht selten von größerem Interesse ist die tatsächliche Robustheit eines Netzes N mit Bezug auf ein Eingabebild $\hat{\boldsymbol{x}}$. *Marabou*, als ein Entscheider, kann anschaulich gesprochen nur „Ja-Nein-Fragen" beantworten aber keine Fragen nach konkreten Quantitäten, wie einem konkreten ϵ, etwa dem größtem Mindestrobustheits-ϵ, das ist, der Robustheit von N bezüglich $\hat{\boldsymbol{x}}$. Allerdings kann man sich diesem Wert, der tatsächlichen Robustheit, mittels der Binärsuche beliebig nahe annähern, wie wir im Folgenden darlegen wollen. Für unseren Beispielklassifizierer N aus Abbildung 6 ist für das Eingabebild $\hat{\boldsymbol{x}} = (0.25, 0.75)^\top$ aus Abbildung 7 der Ablauf der Binärsuche nach der größten Mindestrobustheit von N bezüglich $\hat{\boldsymbol{x}}$ in Abbildung 17 dargestellt.

Im Zuge der Binärsuche wird eine Invariante – ein Satz, für dessen Wahrheit man initial garantiert, und dessen Wahrheit während des gesamten Unternehmens der Binärsuche stets eingehalten wird – aufgestellt. Die Invariante lautet:

> Das Netz N ist bezüglich des, als l klassifizierten, Bildes $\hat{\boldsymbol{x}}$ mindestens so robust wie der in ϵ^l hinterlegte Wert, jedoch nicht so robust wie ϵ^u, wofür ein Gegenbeipiel $\hat{\boldsymbol{x}}^{\mathbf{adv}}$, bei welchem $d_\infty(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}^{\mathbf{adv}}) \leq \epsilon^u$ und gleichzeitig $\arg\max(N(\hat{\boldsymbol{x}}^{\mathbf{adv}})) \neq l$, bürgt.

Initial setzen wir zum Einen die Epsilonuntergrenze auf $\epsilon^l := 0$, was der Invariante genüge tut, da ein Bild x, welches *gar nicht* perturbiert werden darf genauso als l klassifiziert wird, wie das ursprüngliche Bild $\hat{\boldsymbol{x}}$. Zum Anderen setzen wir die Epsilonobergrenze auf $\epsilon^u := 1$, was jedes beliebige Eingabebild impliziert, da insbesondere jeder 0-wertige Pixel einen Wert bis inklusive $1 = 0 + \epsilon^u$ und jeder 1-wertige Pixel einen Wert bis inklusive $0 = 1 - \epsilon^u$ erreichen kann. Sofern also das Netz nicht *jedes Bild* als l klassifiziert, wird *Marabou* ein Gegebenbeispiel für die Mindestrobustheit von $\epsilon = 1$ finden, welches wir als unser initiales $\hat{\boldsymbol{x}}^{\mathbf{adv}}$ festhalten, womit wir insgesamt die Invariante im Ausgangszustand erfüllen.

Für einen Schritt der Binärsuche gehen wir nun wie folgt vor. Zunächst bestimmen wir den mittleren Wert zwischen ϵ^l und ϵ^u als $\epsilon^m := (\epsilon^l + \epsilon^u)/2$. Mit Bezug auf diesen Wert ϵ^m gibt es nun zwei Möglichkeiten. Die erste Möglichkeit ist die, dass der Verifizierer, *Marabou*, bescheinigt, dass das Netz N über eine Mindesrobustheit von ϵ^m bezüglich des Bildes $\hat{\boldsymbol{x}}$ verfügt. In diesem Fall wird, ohne dass dabei die Invariante verletzt würde, die Epsilonuntergrenze ϵ^l auf ϵ^m angehoben. Die zweite Möglichkeit besteht darin, dass die Antwort SAT *Marabou*s, zu verstehen gibt, dass es eine Variablenbelegung – und im Speziellen eine Variablenbelegung des $\boldsymbol{x}$-Vektors, welche wir als $\hat{\boldsymbol{x}}^{\mathbf{adv}}$ festhalten – gibt, die zeigt, dass N bezüglich Bild $\hat{\boldsymbol{x}}$ nicht die Mindestrobustheit von ϵ^m besitzt. Wir können, da wir das entsprechende Gegenbeispiel bereits festgehalten haben, demnach die Epsilonobergrenze ϵ^u auf ϵ^m absenken, ohne die Invariante zu verletzen. Da in beiden der geschilderten Möglichkeiten die Invariante eingehalten wird, können wir das Verfahren, wohlgemerkt bei verkleinerter Distanz von $\epsilon^u - \epsilon^l$, fortsetzen und einen weiteren Schritt der Binärsuche tätigen. Prinzipiell kann die Binärsuche – bei stets steigender Präzision der aktuellen Epsilonuntergrenze als Approximation der Robustheit als der größten bisher bekannten Mindestrobustheit ϵ^l – beliebig lange fortgesetzt werden. In der Praxis wird man sich mit einer Mindestpräzision $\epsilon_{\text{precision}}$ begnügen und die Binärsuche abbrechen, sobald der Abstand zwischen Epsilonober- und Epsilonuntergrenze die Mindestpräzision erreicht hat: $\epsilon^u - \epsilon^l \leq \epsilon_{\text{precision}}$.

Das Verfahren der Binärsuche zur Approximation der Robustheit eines Netzes in Bezug auf ein Eingabebild ist auch für *MNIST*-Klassifizierer und handgeschriebene Ziffernbilder geeignet, wovon wir in Kapitel 9 bei der Falsifikation *Marabou*s Gebrauch machen werden. Im nächsten Kapitel soll jedoch zunächst untersucht werden, wie, auf welche Weise, *Marabou* bei der Verifikation Neuronaler Netze zu Werke geht.

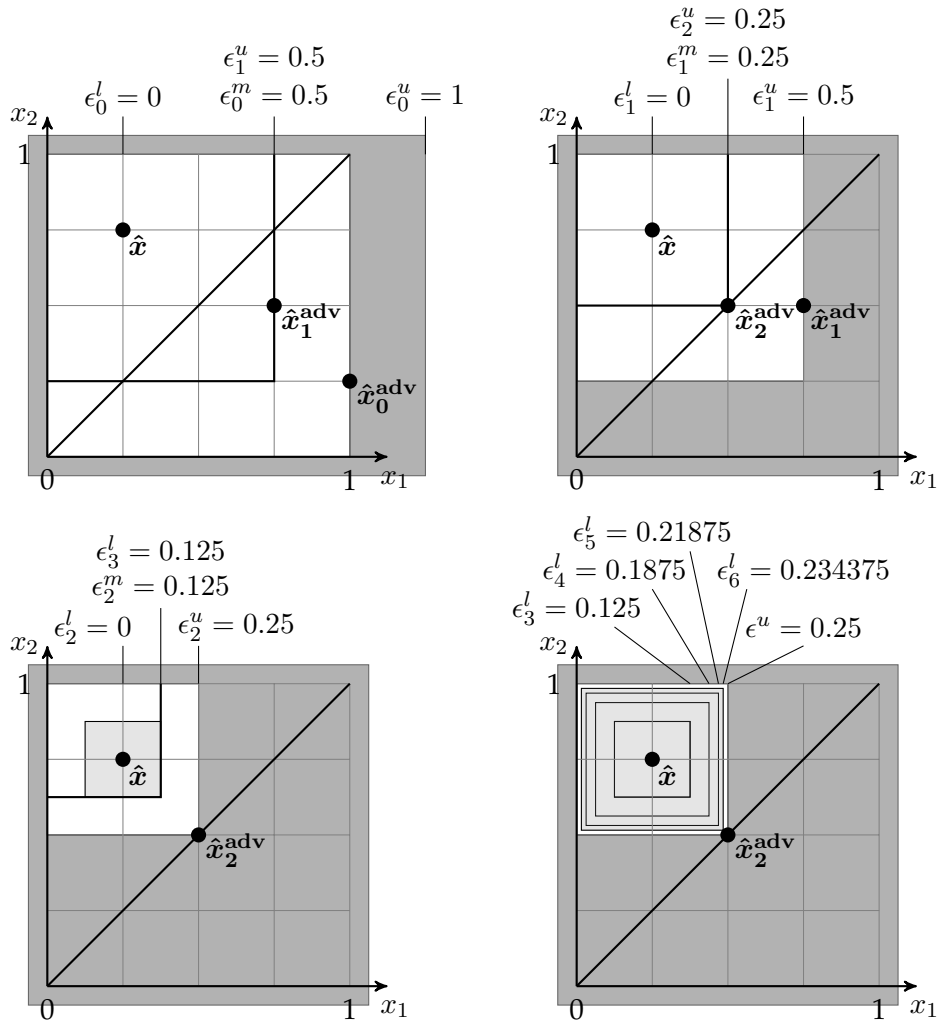


Abbildung 17: Veranschaulichung der Binärsuche nach der größten Mindestrobustheit. Das ursprüngliche Bild $\hat{\boldsymbol{x}}$ wird als $l = 2$ klassifiziert. Bereiche für sicher unrobuste Epsilonwerte sind in dunklem Grau dargestellt, Bereiche für sicher robuste Epsilonwerte in hellem Grau, für weiße Bereiche ist die Robustheit der Epsilonwerte (noch) nicht entschieden. Oben links: Im Ausgangszustand ist $\epsilon_0^l = 0$ kleinstmöglich und $\epsilon_0^u = 1$ größtmöglich gewählt. Für $\epsilon_0^u = 1$ gibt es das Gegenbeispiel $\hat{\boldsymbol{x}}_{\mathbf{0}}^{\mathbf{adv}}$, welches sicher nicht als l klassifiziert wird. Für das mittlere ϵ_0^m zwischen ϵ_0^l und ϵ_0^u, gibt es das Gebenbeispiel $\hat{\boldsymbol{x}}_{\mathbf{1}}^{\mathbf{adv}}$, weshalb $\epsilon_1^u = \epsilon_0^m$ gesetzt wird. Oben rechts: Wieder gibt es ein Gegenbeispiel $\hat{\boldsymbol{x}}_{\mathbf{2}}^{\mathbf{adv}}$ für das mittlere ϵ_1^m auf dessen Wert daraufhin ϵ_2^u angepasst wird. Unten links: Da N bezüglich $\hat{\boldsymbol{x}}$ für das mittlere ϵ_2^m die Uneigenschaft der Mindestrobustheit besitzt, wird das untere ϵ_3^l auf den Wert von ϵ_2^m angehoben. Unten rechts: Weitere Schritte der Binärsuche ergeben eine kontinuierliche Anhebung der Epsilonuntergrenze ϵ^l, welche sich der Epsilonobergrenze ϵ^u immer weiter annähert.

7 Simplex, Reluplex und Marabou

> The term "simplex" evolved from an early geometrical version in which (like in game theory) the variables were nonnegative and summed to unity. In that formulation a class of "solutions" was considered which lay in a simplex.
>
> George Bernard Dantzig, Alex Orden und Philip Starr Wolfe, *The Generalized Simplex Method for Minimizing a Linear Form under Linear Inequality Restraints* [DOW55]

> My proposal [the simplex method] served as a kind of trigger—ideas that had been brewing all through World War II but had never found expression burst forth like an explosion.
>
> George Bernard Dantzig, *Origins of the Simplex Method* [Dan90b]

> We call the algorithm Reluplex, for "ReLU with Simplex".
>
> Guy Katz, Clark Barrett, David Dill, Kyle Julian und Mykel Kochenderfer, *Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks* [KBD+17]

Im Jahr 1937 – Amerika und dem Rest der Erde sind noch zwei Jahre relativen Friedens gewährt, bevor es von den aus Europa kommenden und schnell die ganze Welt umspannenden Ereignissen des zweiten großen Krieges dieses Jahrhunderts erschüttert werden wird – tritt der junge George Dantzig eine, durch den Weggang Milton Friedmans an die Universität von Chicago, vakant gewordene Stelle als Statistiker im Büro für Arbeitsstatistik, *Bureau of Labor Statistics*, in Washington an. In der Abteilung für *Urban Study of Consumer Purchases* erlangte er zum ersten Mal Kenntnis des von, dem dort ebenfalls beschäftigten, Jerry Cornfield aufgeworfenen Problems, welches als „The Diet Problem", bekannt wurde. Aus Dantzigs Erinnerungen jener Zeit, welche Jahrzehnte später in einem Aufsatz gleichen Namens, „The Diet Problem", [Dan90a], erschienen und aus dem die obige Anekdote entnommen ist, geht hervor, dass die erste Anwendung der von ihm, Dantzig, während des Krieges im Dienste der *U. S. Air Force* stehend, neu ersonnenen Simplex-Methode eben jenes Problem des „optimalen Speiseplans" gewesen ist: Gesucht war demnach, in den, von Dantzig ebendort, [Dan90a], zitierten, Worten Jerry Cornfields, „a low cost diet that would meet the nutritional need of a GI soldier."

In diesem Kapitel werden wir in groben Zügen die Arbeitsweise des Verifikationsprogramms von Eigenschaften Neuronaler Netze *Marabou* vorstellen, welches auf sehr enge Weise mit dem durch Dantzig berühmt gewordenen Simplexverfahrens verbunden ist. Dabei übernimmt *Marabou*, ganz allgemein betrachtet, die Rol-

le eines Entscheiders, der feststellt, ob ein, in der Sprache *Marabou*s – der Sprache der quantorenfreien Linearen Arithmetik Reeler Zahlen, kurz *QFLRA* – vorliegender Satz wahr oder falsch ist. *Marabou* fungiert somit als ein „*SMT*-Solver“ der ermittelt, ob ein gegebener Satz zur „Theorie“ – genauer dem Fragment einer Theorie, den wahren Sätzen dieser Sprache – *Marabou*s gehört. Dies bedeutet, ausgedrückt als Problem der *Satisfiability Modulo Theory*, wofür *SMT* schließlich abgekürzt steht, dass das *Marabou*-Programm die Antwort *SAT*, für englisch *satisfiable*, „erfüllbar“, berechnet, wenn der in Frage stehende Satz, gegeben („modulo“) die Theorie der (wahren Sätze der) quantorenfreien Linearen Arithmetik Reeler Zahlen, wahr ist und *UNSAT*, für englisch *unsatisfiable*, „unerfüllbar“, berechnet, wenn der übergebene Satz falsch ist. Die wesentlichen Konturen, in welchen *Marabou* diese Berechnungen durchführt, werden in diesem Kapitel gezeichnet werden. Der Ausgangspunkt und ein Herzstück *Marabou*s ist hierbei Dantzigs Simplex-Algorithmus, auf dem der *Reluplex*-Algorithmus *Marabou*s zum wesentlichen Teil basiert. Allerdings stellt sich in diesem Kontext, um im Bild des „Diet Problems“ zu bleiben, nicht die Frage, was die *konstengünstigste* Möglichkeit ist, das „Regierungseigentum“ – die Kurzform „*GI*“ in „*GI soldier*“ ist die U. S.-amerikanische Abkürzung für *government issue*, eine Bezeichnung die ursprünglich nur den an die Truppe „von der Regierung ausgegebenen“ Ausrüstungsgegenständen zukam – unter einer Reihe nur beschränkt zur Verfügung stehender Lebensmitteltypen auf eine nachhaltig gesunde Art – was die Mindestaufnahme bestimmter Vitamine, Mineralien, Nährstoffe umfasst sowie die Höchstmenge an bedenklichen Stoffen wie Fetten, bestimmten Metallen und dergleichen – zu ernähren, sondern *ob dies*, gegeben die Beschränkungen, *überhaupt möglich ist*. Dantzigs Gruppe bei der *Air Force* interessierte sich eher für das erste Problem, welches ein Optimierungsproblem beschreibt. Im Zuge des mit *Marabou* und der Verifikation von Eigenschaften Neuronaler Netze verbundenen *SMT*-Problems interessiert uns hier die zweite Art von Problemen, die Erfüllbarkeitsprobleme.

7.1 Von Reluplex zu Marabou

Marabou hat einen Wegbereiter. Das Vorläuferprojekt der Personen, welche der Öffentlichkeit das von ihnen entwickelte *Marabou*, als frei zugängliche Software, zum Geschenk machten, hieß und heißt *Reluplex*. Es realisierte bereits die *SMT*-basierte Verifikation von Eigenschaften derjenigen Neuronalen Netzwerke, bei denen die Aufgabe der Aktivierungsfunktion durch *ReLU*-Funktionen übernommen werden. Der Name *Reluplex* bezeichnet sowohl das Verifikationsprogramm als auch den diesem Programm eigenen Algorithmus, der die Verifikation berechnet und ist ein Amalgam aus „*ReLU*“ – der Aktivierungsfunktion sowie der Emphase der besonderen Fähigkeit von *Reluplex* mit den Schwierigkeiten, die diese Funktion bei der Verifikation aufwirft, umgehen zu können – und einer Variante von Dantzigs „*Simplex*“-Algorithmus.

Der Aufsatz, in welchem das *Reluplex*-Programm zur Verifikation von Eigen-

schaften Neuronaler Netze – im Englischen „*Deep Neural Networks*“ oder kurz „*DNN*s“ genannt – vorgestellt wird, „Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks“, [KBD+17], ist wohl *stante pede* allein dadurch zum Klassiker der jüngsten Geschichte der Informatik geworden, dass im Anhang I, „Verifying Properties in DNNs with ReLUs is NP-Complete“, des Aufsatzes zum ersten Mal der formale Beweis erbracht wird, dass die Verifikation von Eigenschaften eines Neuronalen Netzes mit *ReLU*-Aktivierungen ein *NP*-vollständiges Problem ist. Letzteres könnte eine ernüchternde Nachricht für alle diejenigen darstellen, die auf die Möglichkeit zur Verifikation von Eigenschaften „sehr großer“ Netze gehofft hatten. Die *NP*-Vollständigkeit der Verifikation von Eigenschaften Neuronaler *ReLU*-Netze impliziert nämlich, dass die Verifikation einer um eine „Einheit“ – hier beschrieben in der Anzahl der *ReLU*-Knoten des Netzes – größeren Probleminstanz potentiell mit der Ver-x-fachung, wobei $x \in \mathbb{R}$ und $x > 1$, des Lösungsaufwandes einhergeht, was einem mit der Problemgröße asymptotisch exponentiell steigendem Verifikationsaufwand entspricht. Nichtsdestominder ist die Verifikation von Eigenschaften von Netzen mit einer, in Summe, immerhin vierstelligen Dezimalzahl an *ReLU*-Neuronenknoten mittels *Reluplex* durchgeführt und im selben Aufsatz, [KBD+17], präsentiert worden.

Der vielleicht wichtigste Grund für die Ablösung von *Reluplex* durch *Marabou* bestand in dem Ziel die Effizienz der Verifikation zu erhöhen, um dadurch Eigenschaftsverifikationen von Netzen mit einer noch größeren Knotenzahl innerhalb einer vertretbaren Zeitspanne durchführen zu können. Wie dem Aufsatz „The Marabou Framework for Verification and Analysis of Deep Neural Networks“, [KHI+19], zur Verifikationssoftware *Marabou*, zu entnehmen ist, sind die Hauptverbesserungen gegenüber *Reluplex* dementsprechend auch die folgenden. Durch einen neuen *divide-and-conquer*-Modus, der es *Marabou* insbesondere erlaubt, bei Bedarf eine Verifikationsaufgabe rekursiv in leichter zu lösende Teilaufgaben zu zerlegen, wird es möglich, die Berechnung des Verifikationsergebnisses, unter Verwendung mehrerer Zentraler Recheneinheiten, zu parallelisieren und dadurch massiv zu beschleunigen. Der Kernel von *Marabou* umfasst des Weiteren nun eine integrierte Version des Simplex-basierten Lösungsverfahrens, was den aufwändigen Kontextwechsel zwischen *GLPK* – dem insbesondere nicht auf die Anforderungen der Verifikation spezialisierten „*GNU Linear Programming Kit*“ – und Hauptprogramm, welcher noch in *Reluplex* vorherrschte, unnötig macht. Weitere Effizienzsteigerungen rühren von verfeinerten Deduktionsmöglichkeiten her, welche es *Marabou* erlauben, vereinfachende Umformungen der Problemformulierung basierend auf der gesamten Topologie des Netzes vorzunehmen, die den Suchraum – die Verifikation einer Eigenschaft ist die Suche nach einer Variablenbelegung, unter der die Formel 36 wahr ist – einschränken und die Suche dadurch effizienter gestalten. Diese und sonstige effizienzverbessernde Maßnahmen sind für unser Projekt hier nicht von Belang, da wir letztlich an der Korrektheit und nicht der Geschwindigkeit des Verifikationsprozesses durch *Marabou* interessiert sind. Da wir uns außerdem nur auf diejenige Art von Netzarchitektur beschränken wollen, welche auch bei *Relu-*

plex Verifikationsgegenstand ist, nämlich Neuronale *ReLU*-Netze, so nehmen wir die architekturbezüglichen Neuerungen *Marabou*s – welche nun die Verwendung beliebiger Aktivierungsfunktionen, solange sie, wie die *ReLU*-Funktion, als stückweise Lineare Funktionen dargestellt werden können, gestattet und auch sogenannte *Max-Pooling*-Layer, das sind Knotenverbünde, von denen nur der jeweilige Maximalwert einen Nachfolgeknoten erreicht, erlaubt, was bestimmte Formen sogenannter *Convolutional Neural Networks*, welche eine Netzeingabe zusätzlich „zu filtern" in der Lage sind, zur Verifikation zulässt – hier nur zur Kenntnis, ohne von ihnen Gebrauch machen zu wollen.

Eine Neuerung *Marabou*s gegenüber *Reluplex*, der wir uns indes durchaus zu bemüßigen nicht abgeneigt gesehen haben, ist das Angebot, *Marabou* in eine Umgebung der Programmiersprache *Python* einzubetten sowie die Möglichkeit, in *Python*, mittels der Funktionen der Programmbibliothek *TensorFlow*, trainierte Neuronale Netze, welche im sogenannten *Protocol-Buffer*-Format gespeichert wurden, direkt in den *Marabou*-Verifizierer einzulesen, welcher daraufhin ein Pendant der Netzwerkformel 21 erstellt. Auch die Spezifikation einer Eigenschaft P kann in *Python* erfolgen, sodass schließlich die gesamte Verifikationskette von Eigenschaften Neuronaler Netze – von der Erstellung und dem Training, der Netze, der Spezifikation der Eigenschaften und endlich der Verifikation selbst – in einer einheitlichen Programmierumgebung, dem *Python*-Interpreter, erfolgen kann.

Da, wie eben erläutert, die durch *Marabou* eingeführten Neuerungen keinen wesentlichen Mehrwert – was die Grundzüge der Funktionsweise der durch *Marabou* vollzogenen Verifikation angeht – gegenüber *Reluplex* leisten, konzentrieren wir uns für die Beschreibung des Verifikationsprozesses durch *Marabou* im Folgenden auf das, was auch in *Marabou* das Zentrum der Verifikation darstellt, den *Reluplex*-Algorithmus.

Wir hatten angedeutet, dass *Reluplex* – „Simplex mit ReLUs" – auf dem Simplexverfahren aufbaut, dieses aber zusätzlich um eine algorithmische Komponente – wenn man so will, eine „*ReLU*-Komponente" – anreichert, die eine Behandlung der *ReLU*-Kodierungen, Formel 16, der Netzwerkformel 21 dergestalt ermöglicht, dass die Wahrheit des Satzes, der aus dem Existenzabschluss der Konjunktion von Netzwerkformel und der Eigenschaftskodierung entsteht – lies: von Formel 36 –, vergleichsweise „kostengünstig" entschieden werden kann. Zwar ist nämlich die Formel 36 durch den Simplexalgorithmus allein prinzipiell entscheidbar, dies aber, so werden wir sehen, nur zu sehr hohen Laufzeitkosten. Die Hinzufügung einer eigens dafür konzipierten *ReLU*-Komponente zum Simplexalgorithmus dient also maßgeblich der Effizienzsteigerung des Verifikationsverfahrens von Eigenschaften Neuronaler *ReLU*-Netze. In diesem und den folgenden Abschnitten soll der *Reluplex*-Algorithmus vorgestellt werden. Wir sehen uns dabei genötigt, diesen in recht groben Zügen darzustellen, da eine gebührlich ausführliche Repräsentation der Problematik – der Linearen Programmierung, wie die Lineare Optimierung eines Systems von Linearen Ungleichungen unter einer Zielfunktion alternativ genannt wird – im Allgemeinen und selbst des Simplexverfahrens zur Lösung dieses Problemes

im Besonderen, den Rahmen dieser Arbeit überanspruchen müsste. Eine vollumfängliche Darstellung speziell des *Reluplex*-Algorithmus würde darüberhinaus eine Inspektion des quelloffenen *Marabou*-Programmcodes erforderlich machen, da der *Reluplex*-Aufsatz, [KBD+17], viele diesbezüglich notwendige Details unspezifiziert – und der Aufsatz zu *Marabou*, [KHI+19], diese gänzlich undokumentiert – lässt, was den Rahmen dieser Arbeit endgültig durchbrechen würde.

7.2 Die Grundidee des Simplexalgorithmus

Um zu verstehen, welchen Anteil der Simplex-Algorithmus an der Entscheidung der Frage haben kann, ob ein Satz der Sprache *Marabou*s, der quantorenfreien Linearen Arithmetik Reeller Zahlen, *QFLRA*, wahr oder falsch ist, erinnern wir uns daran, dass die „elementaren Bausteine" dieser Sprache die in Definition 19 dargelegten atomaren Proto-Sätze

$$\Big(\sum_{c_i \in \mathbb{D},\, x_i \in V} c_i \cdot x_i \bowtie c \Big), \text{ mit } \bowtie \in \{\leq, \geq\},$$

sind, bei denen der Diskursbereich $\mathbb{D}$ der Sprache *Marabou*s durch die Reellen Zahlen $\mathbb{R}$ konstituiert ist, die c_i dieses Diskursbereichs auch *Koeffizienten* heißen, c, ebenfalls ein Element aus $\mathbb{D}$, *Skalar* genannt wird und bei denen die Summe der durch ihre jeweiligen Koeffizienten c_i gewichteten Variablen x_i aus der Menge aller Variablen V überhaupt, also $(\sum_{c_i \in \mathbb{D},\, x_i \in V} c_i \cdot x_i)$, in derjenigen Relation zum Skalar c stehen, welche durch $\leq$ oder $\geq$ beschrieben ist.

Diese Proto-Sätze werden auch, wie wir in Abschnitt 5.6 dargelegt hatten, Lineare Ungleichungen oder *Lineare Beschränkungen*, im Englischen *linear constraints*, geheißen, was daher rührt, dass sie eine schwache Ungleichheit – der Summations-Term $(\sum_{c_i \in \mathbb{D},\, x_i \in V} c_i \cdot x_i \bowtie c)$ auf der linken Seite ist „höchstes (mindestens)" so groß wie der Skalar-Term c auf der rechten Seite, genau dann, wenn der linksseitige Term zum rechtsseitigen Term in der $\leq$ ($\geq$)-Relation steht – der beiden, durch die Terme auf der jeweiligen Seite bezeichneten, Gegenstände ausdrücken, bei denen der Term auf der linken Seite der Linariätseigenschaft gehorcht. Die Variablen x_i auf der linken Seite der Ungleichung werden „in Summe", $(\sum_{c_i \in \mathbb{D},\, x_i \in V} c_i \cdot x_i)$, durch den Skalar c der rechten Seite *von oben beschränkt*, falls $(\sum_{c_i \in \mathbb{D},\, x_i \in V} c_i \cdot x_i) \leq c$, da diese Summe nicht größer sein darf als c, oder *von unten beschränkt*, da die Summe in diesem Fall nicht kleiner als c sein darf. Wenn wir im Folgenden von Linearen Ungleichungen reden, meinen wir stets Formeln, welche die schwachen Ungleichheitsrelationen, „$\leq$" und „$\geq$", involvieren, niemals jedoch die starken, durch „$<$" sowie „$>$" ausgedrückten, Ungleichheiten.

Eine nichtleere Menge $\{F_1, \ldots, F_n\}$ von atomaren Proto-Sätzen dieser Art – wir legen fest, dass $F_{\text{any}} := (F_1 \wedge \cdots \wedge F_n)$, das englische any erinnere uns daran, dass wir zunächst „irgendeine" Lösung suchen, die Konjunktion dieser Proto-Sätze bezeichnen möge –, bei der $F_j := (\sum_{c_{ji} \in \mathbb{D},\, x_{ji} \in V} c_{ji} \cdot x_{ji} \bowtie c_j)$, wobei $j \in \{1, \ldots, n\}$ sowie $c_j \in \mathbb{D}$ und, wie gehabt, $\bowtie \in \{\leq, \geq\}$ und schließlich V eine Variablenmenge

ist, wird ein *System Linearer Ungleichungen* genannt. Dieses System *besitzt eine* sogenannte *Lösung*, genau dann wenn der Satz

$$\exists x_1 \cdots \exists x_m \colon F_{\text{any}},$$

bei dem $\{x_1, \ldots, x_m\} = V$, – also der Existenzabschluss der Konjunktion der Linearen Beschränkungen der Menge $\{F_1, \ldots, F_n\}$ – wahr ist. Konkret *ist eine Lösung* eine Zuweisung jeweils eines Gegenstandes d_m des Diskursbereichs $\mathbb{D}$ an jeweils eine Variable x_i der Variablenmenge V in Form der Zuweisungsvorschrift $[x_1 \leftarrow d_1, \ldots, x_m \leftarrow d_m]$ so, dass der Satz

$$F_{\text{any}}[x_1 \leftarrow d_1, \ldots, x_m \leftarrow d_m]$$

wahr ist. Das Interesse der Linearen Programmierung, welche ihr Dasein dem Lösen von Linearen Ungleichungssystemen schuldet – wobei diese Schuld etwa der Simplex-Algorithmus einlöst –, liegt nun nicht allein in der Frage nach *irgendeiner* Lösung eines solchen Systems, sondern *der optimalen*. Dazu wird, falls es mehr als eine Lösung gibt, unter allen möglichen Lösungen wahlweise die (genauer: eine) „kostengünstigste“ Lösung, das ist eine Lösung die den Wert einer Summe der Art

$$\sum_{p_i \in \mathbb{D}, x_i \in V} p_i x_i,$$

welche die „Kosten“ beschreibt, minimiert, wobei p_i etwa die „Preise“ der Gegenstände i, deren Quantitäten x_i die Teilkosten $p_i x_i$ generieren, ausdrücken. Alternativ wird unter allen möglichen Lösungen eine „nützlichste“ – zeitgenössische Ökonomen haben bemerkenswerterweise bis zum heutigen Tage Schwierigkeiten damit, eine zirkelfreie Definition des, zumindest für die Neoklassik fundamentalen, auf Präferenzen basierenden, Nutzen-Begriffs zu geben; eine Peinlichkeit, die sie mit der Biologie hinsichtlich ihres Fundamentalbegriffs, „Leben“, teilen – gesucht, welche also einen Summationswert der Art

$$\sum_{u_i \in \mathbb{D}, x_i \in V} u_i x_i,$$

der den Gesamt-„Nutzen“, als Summe der „Nützlichkeiten“, englisch *utilities*, $u_i x_i$, welche die jeweiligen Mengen x_i des „Gutes“ i jeweils „stiften“, maximiert.

Sei, wie oben $\{F_1, \ldots, F_n\}$ eine nichtleere Menge von atomaren Proto-Sätzen, wobei diesmal $F_j := (\sum_{c_{ji} \in \mathbb{D},\, x_{ji}^{\text{opt}} \in V^{\text{opt}}} c_{ji} \cdot x_{ji}^{\text{opt}} \bowtie c_j)$, mit $j \in \{1, \ldots, n\}$ und $c_j \in \mathbb{D}$ sowie $\bowtie \in \{\leq, \geq\}$, wobei V^{opt} eine Variablenmenge ist, deren Elementen später der optimale Lösungswert zugewiesen werden wird. Daneben sei $\{F'_1, \ldots, F'_n\}$ wiederum eine nichtleere Menge derselben atomaren Proto-Sätzen, jedoch mit alternativer Variablenmenge wie folgt: $F'_j := (\sum_{c_{ji} \in \mathbb{D},\, x'_{ji} \in V'} c_{ji} \cdot x'_{ji} \bowtie c_j)$, mit $j \in \{1, \ldots, n\}$ und $c_j \in \mathbb{D}$ sowie $\bowtie \in \{\leq, \geq\}$, bei der V' eine zwar gleichmächtige, ($|V^{\text{opt}}| = |V'|$) aber

disjunkte Variablenmenge ist, $(V^{\text{opt}} \cap V') = \emptyset$, sodass die Variablenmenge insgesamt $V := (V^{\text{opt}} \cup V')$ umfasst. Als Abkürzungen legen wir fest, dass

$$F_{\min} := (F_1 \wedge \cdots \wedge F_n) \wedge (F'_1 \wedge \cdots \wedge F'_n) \quad \wedge \\ \Big((\sum_{p_i \in \mathbb{D}, x_i^{\text{opt}} \in V^{\text{opt}}} p_i x_i^{\text{opt}}) \leq (\sum_{p_i \in \mathbb{D}, x'_i \in V'} p_i x'_i) \Big),$$

einen, die Quintessenz des Minimierungsproblems kapselnden quantorenfreien Protosatz bezeichnet eben so, wie der folgende die Quintessenz des Maximierungsproblems enkapsuliert:

$$F_{\max} := (F_1 \wedge \cdots \wedge F_n) \wedge (F'_1 \wedge \cdots \wedge F'_n) \quad \wedge \\ \Big((\sum_{u_i \in \mathbb{D}, x_i^{\text{opt}} \in V^{\text{opt}}} u_i x_i^{\text{opt}}) \geq (\sum_{u_i \in \mathbb{D}, x'_i \in V'} u_i x'_i) \Big).$$

Das Simplex-Verfahren, welches das jeweilige Optimierungsproblem zu lösen in der Lage ist, entscheidet demnach, ob, im Falle des Minimierungsproblems, folgender Satz

$$\exists x_1^{\text{opt}} \cdots \exists x_m^{\text{opt}} \forall x'_1 \cdots \forall x'_m \colon F_{\min}$$

– den man lesen kann als die Behauptung „Das durch die Menge $\{F_1, \ldots, F_n\}$ beschriebene Lineare Ungleichungssystem, hat, vor dem Hintergrund der Zielfunktion $(\sum_{p_i \in \mathbb{D}, x_i^{\text{opt}} \in V^{\text{opt}}} p_i x_i^{\text{opt}})$, die Lösung $[x_1^{\text{opt}} \leftarrow d_1, \ldots, x_m^{\text{opt}} \leftarrow d_m]$ so, dass die Kosten $(\sum_{p_i \in \mathbb{D}, x'_i \in V'} p_i x'_i)$ jeder anderen Lösung $[x'_1 \leftarrow d_{m+1}, \ldots, x'_m \leftarrow d_{2m}]$ mindestens so hoch wie die Kosten jener Lösung sind." – respektive, im Fall des Maximierungsproblems der Satz

$$\exists x_1^{\text{opt}} \cdots \exists x_m^{\text{opt}} \forall x'_1 \cdots \forall x'_m \colon F_{\max}$$

– lies: „Das durch die Menge $\{F_1, \ldots, F_n\}$ beschriebene Lineare Ungleichungssystem, hat, vor dem Hintergrund der Zielfunktion $(\sum_{u_i \in \mathbb{D}, x_i^{\text{opt}} \in V^{\text{opt}}} u_i x_i^{\text{opt}})$, die Lösung $[x_1^{\text{opt}} \leftarrow d_1, \ldots, x_m^{\text{opt}} \leftarrow d_m]$ so, dass der Nutzen $(\sum_{u_i \in \mathbb{D}, x'_i \in V'} u_i x'_i)$ jeder anderen Lösung $[x'_1 \leftarrow d_{m+1}, \ldots, x'_m \leftarrow d_{2m}]$ höchstens so groß ist wie derjenige jener Lösung." – wahr oder falsch ist. Ist der Satz falsch, so kann es auch keine Lösung des Problems geben. Ist der Satz hingegen wahr, so ist eine Lösung des Optimierungsproblems wiederum eine Belegung $[x_1^{\text{opt}} \leftarrow d_1, \ldots, x_m^{\text{opt}} \leftarrow d_m, x'_1 \leftarrow d_{m+1}, \ldots, x'_m \leftarrow d_{2m}]$ der Variablen aus V mit Gegenständen $\{d_1, \ldots, d_{2m}\} \subseteq \mathbb{D}$ so, dass der Satz

$$F_{\min}[x_1^{\text{opt}} \leftarrow d_1, \ldots, x_m^{\text{opt}} \leftarrow d_m, x'_1 \leftarrow d_{m+1}, \ldots, x'_m \leftarrow d_{2m}],$$

beziehungsweise der Satz

$$F_{\max}[x_1^{\text{opt}} \leftarrow d_1, \ldots, x_m^{\text{opt}} \leftarrow d_m, x'_1 \leftarrow d_{m+1}, \ldots, x'_m \leftarrow d_{2m}],$$

wahr ist.

7.3 Geometrische Deutung des Simplexalgorithmus

Es gibt eine geometrische Veranschaulichung des Problems der Optimierung – wir wollen uns ohne Beschränkung der Allgemeinheit auf den Fall der Minimierung beschränken – Linearer Ungleichungssysteme. Die Veranschaulichung der Lösung eines Minimierungsproblems Linearer Ungleichungssysteme, etwa mithilfe des Simplexalgorithmus, als die Lösung eines geometrischen Problems bringt dabei jedoch ähnliche Schwierigkeiten mit sich, wie die Moral einer Anekdote, die zuweilen unter Studierenden der Mathematik kursiert. Dieser Erzählung zufolge wird demnach eine Topologie-Professorin im Anschluss an eine ihrer Vorlesungen von Studierenden befragt, wie man sich Gegenstände im n-dimensionalen Raum denn veranschaulichen könnte. „Oh, das ist einfach." antwortet diese, während sie $\mathbb{R}^n$ an die Tafel schreibt. „Stellen Sie sich einen Gegenstand im dreidimensionalen Raum vor.", sie wischt das n von der Tafel und ersetzt es durch eine 3, sodass nun $\mathbb{R}^3$ an der Tafel steht. Die Studierenden nicken einsichtsvoll. „Nun", während sie die 3 wieder wegwischt, „generalisieren Sie einfach", sie schreibt erneut n, „zum allgemeinen Fall n," an der Tafel steht, wie zu Anfang, $\mathbb{R}^n$, „dann haben Sie es".

Die geometrischen Gegenstände des n-dimensionalen Raumes werden in unserem Fall durch den Schnitt von *Halbräumen* bestimmt. Jeweils ein solcher Halbraum wird durch jeweils eine Lineare Ungleichung des Systems beschrieben. Die Ungleichung $(\sum_{c_i \in \mathbb{D},\, x_i \in V} c_i \cdot x_i \leq c)$, bei der $c \in \mathbb{D}$, mit $\mathbb{D} = \mathbb{R}$, sei und $V := \{x_1, \ldots, x_n\}$ hier als endliche Variablenmenge angenommen sei, wird dabei als die Beschreibung eines n-dimensionalen Halbraums verstanden. Dieser Halbraum besteht demnach nun aus einer Menge von Koordinaten $(d_1, \ldots, d_n)$, deren Tupelkomponenten d_i Elemente des Diskursbereichs $\mathbb{D}$ sind. Konkret sind dies alle Koordinaten $(d_1, \ldots, d_n)$ derjenigen Art, die die Belegung $[x_1 \leftarrow d_1, \ldots, x_n \leftarrow d_n]$ so formen, dass der Satz $(\sum_{c_i \in \mathbb{D},\, x_i \in V} c_i \cdot x_i \leq c)[x_1 \leftarrow d_1, \ldots, x_n \leftarrow d_n]$ wahr ist. Diese Koordinatenmenge, lässt sich bequem in einer Notation angeben, die im Englischen als *set-builder*-Notation bezeichnet wird, aber im Deutschen keine einheitliche Bezeichnung zu haben scheint. Demnach wird eine Menge $M := \{e \mid C\}$ dadurch „aufgebaut", dass e, ein potentielles Element, genau dann in M enthalten ist, wenn es die Bedingungen, welche durch C spezifiziert sind, erfüllt. Die *via* $(\sum_{c_i \in \mathbb{D},\, x_i \in V} c_i \cdot x_i \leq c)$ induzierte, den Halbraum konstituierende, Koordinatenmenge ist demnach durch die Menge $\{(d_1, \ldots, d_n) \mid (d_1 \in \mathbb{D} \wedge \cdots \wedge d_n \in \mathbb{D}) \wedge ((c_1 d_1 + \cdots + c_n d_n) \leq c)\}$ umfassend beschrieben. Ein *Schnitt* zweier Halbräume bezeichnet den (Mengen-)Schnitt zweier solcher Koordinatenmengen. Seien etwa $(\sum_{c_{1,i} \in \mathbb{D},\, x_{1,i} \in V} c_{1,i} \cdot x_{1,i} \bowtie_1 c_1)$ und $(\sum_{c_{2,i} \in \mathbb{D},\, x_{2,i} \in V} c_{2,i} \cdot x_{2,i} \bowtie_2 c_2)$ zwei Ungleichungen des Systems, mit $c_1, c_2 \in \mathbb{D}$ und $\bowtie_1, \bowtie_2 \in \{\leq, \geq\}$, so ist der Schnitt der induzierten Halbräume die Menge folgender Koordinaten.

$$\{(d_1, \ldots, d_n) \mid (d_1 \in \mathbb{D} \wedge \cdots \wedge d_n \in \mathbb{D}) \quad \wedge \\ ((c_{1,1} d_1 + \cdots + c_{1,n} d_n) \bowtie_1 c_1) \wedge ((c_{2,1} d_1 + \cdots + c_{2,n} d_n) \bowtie_2 c_2)\ \}$$

Verallgemeinern wir den Fall auf den Schnitt aller durch die Ungleichungen des

Systems induzierten n-dimensionalen Halbräume, so wird dadurch ein geometrischer (höchstens) n-dimensionaler Gegenstand definiert, der aus Menge der Koordinaten – n-Tupeln – besteht, die in der Schnittmenge aller dieser Halbräume liegen. Im zweidimensionalen Fall ist dies „das Innere“ – wobei „innen“ diejenige Hälfte des Raumes meint, welche durch die vermittels der entsprechenden Ungleichung induzierte Koordinatenmenge konstituiert ist und „außen“ die andere Hälfte – eines, nicht notwendigerweise geschlossenen, aber konvexen („konvex“ bedeutet hier, dass für alle zwei Koordinaten die im Inneren liegen, auch jeder Punkt auf der Strecke, die diese zwei Koordinaten als Start- und Endpunkt besitzt im Inneren liegen muss), Polygonzuges. Im verallgemeinerten n-dimensionalen Fall ist der durch den Schnitt der Halbebenen charakterisierte geometrische Gegenstand nun ein konvexes sogenanntes *Polytop*. Man beachte, dass „die Ränder“, also die Ebenen, welche den einen vom anderen jeweiligen Halbraum abtrennen, dieses Polytops stets zum Gegenstand dazugehören, da wir es invariant mit schwachen und nie mit starken Ungleichungsrelationen zu tun haben. Wenn wir künftig von „inneren“ Koordinaten eines Polytops reden, sind immer auch die Koordinaten auf diesen Rändern mitgemeint. Für die Minimierung kommt die durch $\sum_{p_i \in \mathbb{D}, x_i \in V} p_i x_i$ induzierte Ebene hinzu, bei der die Werte der Belegung der Koordinatenkomponenten x_i so gewählt werden – was anschaulich gesprochen einer, hinsichtlich der möglichen Richtungen durch die nichtvariablen Koeffizienten p_i festgelegten, Verschiebung dieser Ebene entspricht –, dass sie obiges Polytop so schneidet, oder, genauer: berührt, dass keine andere Wahl der Belegung der Variablenkomponenten x_i, welche das Polytop ebenfalls schneidet, die Summe $\sum_{p_i \in \mathbb{D}, x_i \in V} p_i x_i$ zu einem kleineren Wert führt.

Betrachten wir beispielsweise die Produktion eines Pfefferkuchenhauses, wobei wir der Gefahr –

> Was soll man nun dazu sagen, wenn jemand [...] in die Kinderstube geht [...], um dort wie J. St. Mill etwa eine Pfefferkuchen- oder Kieselsteinarithmetik zu entdecken!
>
> — Gottlob Frege, *Die Grundlagen der Arithmetik*, [Fre84]

– eine Pfefferkuchen-Geometrie zu entdecken stets, den n-dimensionalen allgemeinen Fall nicht aus den Augen lassend, eingedenk sein wollen. Dieses Haus habe ein Flachdach und neben einer Grundfläche des Weiteren vier Wände und bestehe aus (infinitisimal dünnen) Pfefferkuchenplatten, deren gedankliche Erweiterung zu einer unendlichen Fläche eine Ebene beschreiben möge. Jede der sechs Platten sei zusätzlich zur Funktion der Platte – „Bodenplatte“, „Wand links“, „Wand hinten“, „Wand rechts“, „Wand vorn“, „Dachplatte“– auf einer Seite mit „innen“ und auf der Rückseite mit „außen“ beschriftet. Des Weiteren verfügen wir über einen Tisch mit ebener Oberseite. Alle Pfefferkuchenplattenseiten, die mit „innen“ beschrieben sind definieren, als ideale Ebenen, einen Halbraum, und zwar die Menge aller Raumkoordinaten, die auf dieser Seite der Platte liegen. Indem wir das Pfefferkuchenhaus nun Platte für Platte zusammensetzen, die Bodenplatte zeigt mit der „innen“-Seite nach oben, die „innen“-Seiten der Wände zeigen ins Innere des im Entstehen be-

findlichen Hauses, das Dach zeigt mit der „innen“-Seite nach unten – bringen wir Halbraum für Halbraum zum Schnitt, bis schließlich, sofern das Haus anleitungskonform assembliert wurde, ein Polyeder, ein dreidimensionales Polytop, eben das (Innere des) Pfefferkuchenhaus entstanden ist.

Eine optimale Lösung des Minimierungsproblems können wir in unserem Beispiel veranschaulichen, indem wir als Beschreibung der Kostenfunktion die „Ebene“ der Tischplatte wählen, wobei es günstig ist, wenn der Tisch auf dem Fußboden steht und immer teurer wird, je näher die Ebene der Tischplatte der Zimmerdecke kommt. Nehmen wir an, das Pfefferkuchenhaus schwebt irgendwo in der Mitte des Raumes, wobei die Bodenplatte unseres Pfefferkuchenhauses parallel zur Tischplatte liegt und der Tisch anfangs auf dem Fußboden, mit allen seinen Teilen klar unterhalb der Bodenplatte des Pfefferkuchenhauses, steht. Da sich das Innere des Pfefferkuchenhaus und die Tischplatte nicht schneiden, und ein weiteres Absenken der Tischplatte, und damit der Kosten, in Richtung Fußboden nicht nur nicht erlaubt ist, sondern den Abstand zwischen jedem beliebigen Punkt im Inneren des Hauses und der Tischplatte nur noch weiter erhöht, muss der Tisch, bei steigenden Kosten dieser Operation, in Richtung der Zimmerdecke angehoben werden, solange, bis die Tischoberseite die Pfefferkuchenhausunterseite genau berührt und der Tisch somit exakt unterhalb des Hauses untergesetzt wird. Nun gibt es einen Schnitt der durch die Kostenfunktion beschriebenen Ebene, der Oberseite der Tischplatte, mit der, wohlgemerkt nichtleeren, Schnittmenge der durch das System der Linearen Ungleichungen induzierten Halbräume, dem Inneren des Pfefferkuchenhauses, nämlich genau die Menge aller Koordinaten, die auf der Innenseite der Bodenplatte im Inneren des Hauses liegen. Jede dieser, aus dem Schnitt der Kostenfunktionsebene mit dem Schnitt der Schnitte der durch die Ungleichungen des Systems induzierten Halbebenen resultierenden, Koordinaten ist eine kostenoptimale Lösung unseres Minimierungsproblems. Sicherlich hätte man auch das Haus in Richtung der Tischplatte bewegen können, was aber anschaulich nicht so „teuer“ gewesen wäre, wie den Tisch in Richtung des im Raum schwebenden Pfefferkuchenhauses zu wuchten.

Betrachten wir, um das Beispiel zu einem Abschluss zu bringen, noch den Fall eines „fehlerhaft“ zusammengebauten Pfefferkuchenhauses, bei welchem konkret alle mit „außen“ beschrifteten Seiten der Pfefferkuchenplatten ins Innere des Hauses zeigen. Dieses „Haus“ hat keine „Substanz“: Die es konstituierende Koordinatenmenge ist leer, da der Schnitt etwa des durch die Dachplatte beschriebenen Halbraums, welcher alle Koordinaten oberhalb dieser Platte beinhaltet, mit dem durch die Bodenplatte beschriebenen Halbraum, welcher alle Koordinaten unterhalb dieser Platte beinhaltet, die leere Menge ist, und der Schnitt der leeren Menge mit einer beliebigen Menge immer nur wiederum die leere Menge erzeugt. Ein so beschaffenes System Linearer Ungleichungen hat keine Lösung, auch keine kostenminimale.

7.4 Eine schlankere Version des Simplexalgorithmus

Wir hatten konstatiert, dass der ursprüngliche Simplexalgorithmus das Potential hat, kostenminimale Lösungen eines Sytems Linearer Ungleichungen zu bestimmen, sofern eine solche Lösung überhaupt existiert. Der Simplexalgorithmus allein ist damit bereits hinreichend, um die Wahrheit von Sätzen in der Sprache *Marabou*s zu entscheiden, wie folgende Überlegung zeigt. Eine durch den Simplexalgorithmus berechnete *optimale* Lösung eines Linearen Ungleichungssystems ist, *a fortiori*, eine *Lösung* eines Linearen Ungleichungssystems. Insbesondere ist jede Belegung $[x_1^{\text{opt}} \leftarrow d_1, \ldots, x_m^{\text{opt}} \leftarrow d_m, x'_1 \leftarrow d_{m+1}, \ldots, x'_m \leftarrow d_{2m}]$, die $F_{\max}$ respektive $F_{\min}$ in einen wahren Satz überführt, zugleich – wenn wir uns x_i^{opt} durch x_i ersetzt denken – eine Belegung, die den Proto-Satz F_{any} als einen wahren Satz zurücklässt. Weiters besitzt ein solches System *keine* Lösung, wenn es *keine optimale* Lösung besitzt. Folglich errechnet auch der Simplexalgorithmus, dass es keine Lösung gibt.

Ein System Linearer Ungleichungen, so hatten wir gleich zu Anfang unserer Untersuchung festgestellt, ist nichts anderes als eine Menge wohlgeformter atomarer Proto-Sätze der Sprache *Marabou*s, welche implizit als untereinander durch Konjunktionen verknüpft angenommen werden. Die resultierende Konjunktion wohlgeformter atomarer Proto-Sätze, ist, mit Hinblick auf Definition 19, wiederum ein wohlgeformter Proto-Satz. Indem der Simplexalgorithmus nun eine (optimale) Lösung des Linearen Ungleichungssystems berechnet, vorausgesetzt, eine solche existiert, erzeugt er zugleich eine Variablenbelegung welche beweist, dass der, durch den Existenzabschluss über die Konjunktion der atomaren Proto-Sätze des Linearen Ungleichungssystems, gewonnene Satz wahr ist. Gibt es andererseits keine Lösung, so gibt es auch keine Variablenbelegung so, dass dieser existenziell über der Konjunktion der atomaren Proto-Sätze des Linearen Ungleichungssystems abgeschlossene Satz wahr ist, was die Wahrheit der Negation des existenziell abgeschlossenen Satzes, oder, äquivalent, die Falschheit des Existenzabschlusses über der Konjunktion der atomaren Proto-Sätze des Linearen Ungleichungssystems beweist. Da nun aber der Existenzabschluss eines wohlgeformten Proto-Satzes, nach Definition 20, ein wohlgeformter Satz der Sprache *Marabou*s ist, entscheidet der Simplexalgorithmus die Wahrheit oder Falschheit von Sätzen der Sprache *Marabou*s.

Man beachte, dass der Simplexalgorithmus auch Sätze der Sprache *Marabou*s entscheiden kann, die mindestens ein Logisches Oder, „$\vee$“, enthalten: Aufgrund der Distributivität der Konjunktion über die Disjunktion, für beliebige Formeln F, G und H gilt $(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H))$. Die rekursive Anwendung dieser Logischen Äquivalenz auf einen Ausgangssatz ermöglicht ihre logisch äquivalente Darstellung in Disjunktiver Normalform, eine Formel die eine Disjunktion von Konjunktionen ist. Entscheidet der Simplexalgorithmus nun, für den Proto-Satz, der aus der Entfernung aller (Existenz-)Quantoren der Formel in Disjunktiver Normalform resultiert, dass mindestens eines der Disjunkte – welches nunmehr eine Konjunktion von wohlgeformten atomaren Proto-Sätzen ist – eine Lösung hat, dann, aufgrund der oben, im Beweis zu Satz 7, angesprochenen Oder-Verträglichkeit der Existenzquantoren, und nur dann ist der Ausgangssatz wahr.

Der ursprüngliche Simplexalgorithmus leistet bei der Entscheidung der Wahrheit oder Falschheit von Sätzen der Spache *Marabou*s dabei aber mehr als nötig ist. Wir benötigen ja, wie bereits angesprochen, nicht den Nachweis einer optimalen Lösung, sondern irgendeiner Lösung. Innerhalb der geometrischen Veranschaulichung interessieren wir uns lediglich für das System Linearer Ungleichungen, welches geometrisch als Schnitte von Halbräumen interpretiert wird, welche einen geometrischen Körper, das Polytop, definieren, der, wie besprochen, eine Menge von Koordinaten ist. Gibt es keine Lösung für das Lineare Ungleichungssystem, so ist diese Menge leer. Andernfalls erzwingt die Optimalitätsbedingung die Auswahl aus einer Teilmenge des Lösungsraums, welcher alle Elemente, Koordinaten, der Lösungsmenge umfasst, nämlich derjenigen, welche zugleich das Optimalitätkriterum erfüllen. Diese Einschränkung ist, für die Entscheidung ob ein Satz der Sprache *Marabou*s wahr oder falsch ist, redundant und – da die Einschränkung höchstens mit einer Verkleinerung des Raumes in welchem Lösungen liegen können einhergeht – impliziert einen potentiellen Mehraufwand hinsichtlich der Berechnung des Ergebnisses.

Konsequenterweise verwendet der *Reluplex*-Algorithmus eine Version des Simplexalgorithmus, welche nicht auf das Auffinden einer optimalen, sondern auf eine beliebige Lösung hin konzipiert wurde. Für die folgende Darstellung stützen wir uns gänzlich auf den Aufsatz, der den *Reluplex*-Algorithmus in den wissenschaftlichen Diskurs eingebracht hat, [KBD+17]. Betrachten wir das System Linearer Ungleichungen – wir wollen uns, ohne Beschränkung der Allgemeinheit, auf einen der beiden Ungleichungstypen, und zwar die Kleiner-Oder-Gleich-Beziehung „$\leq$“ beschränken, da die Argumentation für die Größer-Oder-Gleich-Relation zumeist analog verläuft – unter einem andern Blickwinkel. Jede Lineare Ungleichung der Art

$$\Big(\overbrace{\sum_{c_j \in \mathbb{D},\, x_j \in X} c_j x_j}^{t_\ell} \leq \overbrace{c}^{t_r}\Big)$$

drückt zugleich eine Beziehung, eine Relation, zweier Gegenstände des Diskursbereichs aus. Man beachte, dass die Variablenmenge X genau die Variablen enthält, die in den Linearen Ungleichungen des Systems vorkommen und $V \supseteq X$ eine potentiell echte Supermenge von Variablen bezeichnen möge. Für den Term auf der rechten Seite der Ungleichheit, den wir mit t_r bezeichnet haben, ist der Gegenstand einfach aufzufinden: Es ist der Gegenstand $c \in \mathbb{D}$, Punkt. Der Gegenstand, welcher mit t_ℓ bezeichnet auf der linken Seite referenziert wird, ist nicht zu ermitteln, da wir keinen Hinweis haben, an welche Gegenstände des Diskursbereichs die in t_ℓ enthaltenen Variablen gebunden werden sollen.

Aber wir wissen, dass der Term t_ℓ, wie alle Terme, zu einem, noch unbekannten, Gegenstand auswerten muss. Folglich können wir für diesen Gegenstand einen neuen, wohlgemerkt, *Variablen*-Namen $z \notin X$ vergeben, wobei wir die Variablenmenge

zu $V := X \cup \{z\}$ erweitern. Es gilt demnach

$$\left(z = \sum_{c_j \in \mathbb{D},\, x_j \in X} c_j x_j\right) \quad \text{und} \quad (z \leq c).$$

Die Identität drückt dabei aus, dass z und der Term t_l denselben Gegenstand bezeichnen. Die Ungleichheit $(z \leq c)$ macht die Aussage, dass der mit z bezeichnete noch unbekannte Gegenstand höchstens so groß sein kann, wie der konstante und somit bekannte Gegenstand c des Diskursbereichs. Der Gegenstand z ist also *nach oben beschränkt*, er besitzt die Obergrenze, englisch *upper bound*, $u = c$. Hätten wir indes im Ausgangsfall $t_\ell \geq t_r$, so würde z mit c in der Größer-Oder-Gleich-Relation stehen und es wäre $(z \geq c)$, womit z demnach *nach unten beschränkt* wäre und somit die Untergrenze, englisch *lower bound*, $l = c$ besäße. Besitzt z eine Obergrenze $u = c$ aber keine Untergrenze, so darf z, prinzipiell, beliebig klein sein und wir können die untere Grenze von z entsprechend – schematisch, denn die Unendlichkeit bezeichnet offenkundig keinen Gegenstand – als $\ell = -\infty$, das heißt, als negative Unendlichkeit, festhalten. Hat, analog, z keine Obergrenze, so ist $u = \infty$ wahr.

Indem wir das eben beschriebene Verfahren verallgemeinern und somit auf alle $n > 1$ Linearen Ungleichungen des Systems, wobei $\bowtie_1, \ldots, \bowtie_n \in \{\leq, \geq\}$, anwenden, erhalten wir

$$\begin{aligned}
\left(z_1 = \sum_{c_{1j} \in \mathbb{D},\, x_j \in X} c_{1j} x_j\right) \quad &\text{und} \quad (z_1 \bowtie_1 c_1), \\
&\vdots \\
\left(z_i = \sum_{c_{ij} \in \mathbb{D},\, x_j \in X} c_{ij} x_j\right) \quad &\text{und} \quad (z_i \bowtie_i c_i), \\
&\vdots \\
\left(z_n = \sum_{c_{nj} \in \mathbb{D},\, x_j \in X} c_{nj} x_j\right) \quad &\text{und} \quad (z_n \bowtie_n c_n).
\end{aligned} \tag{43}$$

Damit haben wir insbesondere n – die Anzahl der Ungleichungen des Ungleichungssystems – verschiedene Gleichungen der Art $(z_k = \sum_{c_{ki} \in \mathbb{D},\, x_j \in X} c_{ki} x_j)$, jeweils eine für jedes $k \in \{1, \ldots, n\}$, welche insgesamt $m = |X|$ unterscheidbare Variablen umfassen. Dieser Sachverhalt lässt sich kompakt in Form einer Tabelle ausdrücken, welche in Abbildung 18 dargestellt ist.

Das „Innere“ dieser matrixartigen Struktur wird *Tableau* genannt und mit T abgekürzt. Jede Zeile i steht dabei implizit für die i-te Gleichung, das heißt, konkret für $z_i = c_{i1} x_1 + \cdots + c_{ij} x_j + \cdots + c_{im} x_m$, wobei im Tableau in jeder Zeile jedoch nur die Koeffizienten $c_{i1}, \ldots, c_{n1}$, in dieser Reihenfolge, festgehalten sind. Der j-te Koeffizient in Zeile i, welcher zunächst c_{ij} ist, wird stets vermittelst der Notation T_{ij} referenziert. Wir wollen bereits an dieser Stelle betonen, dass die Bestückung der

	(x_1)	$\ldots$	(x_j)	$\ldots$	(x_m)
$(z_1 =)$	c_{11}	$\ldots$	c_{1j}	$\ldots$	c_{1m}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(z_i =)$	c_{i1}	$\ldots$	c_{ij}	$\ldots$	c_{im}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(z_n =)$	c_{n1}	$\ldots$	c_{nj}	$\ldots$	c_{nm}

Abbildung 18: Die Gesamtheit der Ungleichungen eines Systems Linearer Ungleichungen lässt sich in Gleichungen überführen, welche in einem *Tableau* T darstellbar sind. Jede Zeile steht für den Wert einer Basisvariablen, welcher sich aus der Linearkombination der impliziten Nichtbasisvariablen aller Spalten ergibt. Das Tableau T selbst beinhaltet dabei explizit nur die entsprechenden Koeffizienten der jeweiligen Linearkombination jeder Zeile. Der j-te Koeffizient in der i-ten Zeile, das heißt hier, c_{ij}, wird über T_{ij} adressiert.

Tableau-Tabelle *ephemer*, flüchtig, ist: Der sich an Stelle T_{ij} befindliche Wert des Tableaus, ist aktualisierbar und damit insbesondere revidierbar, überschreibbar. So wird es das Gleichungssystem ja nicht ändern, wenn man die Reihenfolge der Gleichungen ändert, etwa die erste und die n-te Gleichung vertauscht oder alle letzten Summationsglieder an die erste und stattdessen die ersten an die letzte Stelle setzt und entsprechend $z_1 = c_{nm}x_m + \ldots + c_{nj}x_j + \ldots + c_{n1}x_1$ erhält. Man beachte demnach, dass T_{ij} stets die i-te Zeile an der j-ten Stelle des Tableaus bezeichnet, der Wert des Koeffizienten, der sich an dieser Stelle befindet, jedoch verschieden sein kann. Aus zunächst $T_{11} = c_{11}$ wird, nachdem die letzte und erste Zeile sowie die erste und die letzte Spalte des Tableaus getauscht wird – was, so hatten wir festgestellt, nichts an der Aussage ändert, die implizit durch das System der Linearen Ungleichungen ausgedrückt ist –, der Wert $T_{11} = c_{nm}$ an derselben Tableau-Adresse.

Die impliziten „Zeilen"-Variablen, die initial durch die z_i, mit $i \in \{1, \ldots, n\}$ bestimmt sind und welche als, mit den entsprechenden Koeffizienten des Tableaus gewichtete, Linearkombination implizit für den Wert einer Zeile stehen, heißen *Basisvariablen*. Alle anderen Variablen, konkret alle impliziten „Spalten"-Variablen, deren entsprechende Koeffizienten einen Eintrag im Tableau besitzen, heißen *Nichtbasisvariablen*; ursprünglich sind dies x_1 bis x_m. Was Basisvariable und was Nichtbasisvariable ist, bestimmt sich dabei ausschließlich aus ihrer impliziten Lage im Hinblick auf das Tableau: Zeilenvariablen sind basal, Spaltenvariablen sind nichtbasal. Für eine gegebene Variable kann diese Rolle, wie wir später sehen werden, durchaus wechseln: Eine Basisvariable kann zu einer Nichtbasisvariablen werden und umgekehrt. Diesem Umstand werden wir dadurch Rechnung tragen, dass wir als Datenstrukturen zwei, aktualisierbare, Tupel definieren. Das n-stellige Tupel B

beinhaltet, in der Zeilenreihenfolge des Tableaus T von oben nach unten, alle Variablen, die zum jeweils aktuellen Zeitpunkt die Rolle der Basisvariablen einnehmen und das m-elementige Tupel N beinhaltet entsprechend, in der Reihenfolge der Spalten des Tableaus von links nach rechts, die Variablen, die zum selben Zeitpunkt die Rolle der Nichtbasisvariablen spielen. Die i-te Komponente von B, die aktuell i-te Basisvariable, deren Wert durch die i-te Zeile des Tableaus implizit beschrieben ist, wird über B_i referenziert so, wie über N_j die aktuell j-te Nichtbasisvariable des Tableaus, deren zugehörige Koeffizienten sich in der Spalte j für jede Zeile abgelistet finden. Die Ausdrücke $B := (B_1, \ldots, B_{i-1}, N_j, B_{i+1}, \ldots, B_n)$ und $N := (N_1, \ldots, N_{j-1}, B_i, N_{j+1}, \ldots, N_m)$ beschreiben beispielsweise die Aktualisierung der Tupel B und N, während der Rollentausch von Basisvariable B_i und Nichtbasisvariable N_j hin zu Nichtbasisvariable B_i und Basisvariable N_j, durchgeführt wird. Im Ausgangspunkt haben wir – man zögere nicht, hierzu noch einmal die Zeilen- und Spaltenbeschriftungen des in Abbildung 18 dargestellten initialen Tableaus zu betrachten –, dass $B := (z_1, \ldots, z_n)$ und $N := (x_1, \ldots, x_m)$. Schließlich wollen wir die Tupel B und N zuweilen wie Mengen behandeln, was uns die Verwendbarkeit von Mengenoperationen, wie „$\in$“, „$\subseteq$“, und dergleichen, erschließen soll. Hiermit ist dann stets die Menge $B := \{B_i \mid i \in \{1, \ldots, n\}\}$ beziehungsweise $N := \{N_j \mid j \in \{1, \ldots, m\}\}$ gemeint.

Damit sind wir nun insbesondere in die Lage versetzt, die in Abbildung 18, im Tableau, impliziten Gleichungen auf eine Weise ausdrücken zu können, die der volatilen Struktur des Tableaus T und der Veränderlichkeit der, die Basisvariablen und Nichtbasisvariablen repräntierenden, Tupel B und N Rechnung trägt. Die im Tableau T impliziten Gleichungen, mit $n = |B|$ Basisvariablen und $m = |N|$ Nichtbasisvariablen, sind damit explizit die folgenden.

$$\begin{aligned} B_1 &= T_{1,1}N_1 + \cdots + T_{1,j}N_j + \cdots + T_{1,m}N_m \\ &\vdots \\ B_i &= T_{i,1}N_1 + \cdots + T_{i,j}N_j + \cdots + T_{i,m}N_m \\ &\vdots \\ B_n &= T_{n,1}N_1 + \cdots + T_{n,j}N_j + \cdots + T_{n,m}N_m \end{aligned} \tag{44}$$

Ausgehend von der Umbenennung der jeweils linken Seiten der Ungleichungen des Systems Linearer Ungleichung, welche uns zu der Formelmenge 43 geführt hatte, haben wir nun die Identitäten $(z_i = \sum_{c_{ij} \in \mathbb{D},\, x_j \in X} c_{ij} x_j)$, die aus dieser Transformation herrührten, in eine Datenstruktur, nämlich das Tableau, überführt. Unter anderem für die Ungleichheiten, $(z_i \bowtie_i c_i)$, dieser Formelmenge 43, legen wir eine weitere Datenstruktur fest, in der wir die Obergrenze u, die Untergrenze l und den aktuellen, sozusagen „vorläufig“ festgelegten, Wert α – eine konkrete Zuweisung, englisch *assignment*, woran das α, der Buchstabe der im griechischen Alphabet die Rolle des römischen „a“ vertritt, erinnern möge –, eines Gegenstandes aus dem Dis-

kursbereich, zu jeder Variablen, egal ob Basis- oder Nichtbasisvariable, festhalten. Das können wir wie in Abbildung 19 dargestellt veranschaulichen.

	x_1	$\ldots$	x_m	z_1	$\ldots$	z_i	$\ldots$	z_n
u	∞	$\ldots$	∞	d_{1u}	$\ldots$	d_{iu}	$\ldots$	d_{nu}
α	0	$\ldots$	0	0	$\ldots$	0	$\ldots$	0
l	$-\infty$	$\ldots$	$-\infty$	d_{1l}	$\ldots$	d_{il}	$\ldots$	d_{nl}

Abbildung 19: Datenstruktur, welche für die Berechnung des Reluplexalgorithmus Buch führt über die Obergrenzen $u(v)$, die Untergrenzen $l(v)$ sowie die aktuelle Wertzuweisung $\alpha(v)$ bezüglich der Gesamtheit der Variablen $v \in V = \{x_1, \ldots, x_m, z_1, \ldots, z_n\}$. Die Ober- und Untergrenzen $d_{iu} \in \mathbb{D} \cup \{\infty\}$ respektive $d_{il} \in \mathbb{D} \cup \{-\infty\}$ für alle z_i, mit $i \in \{1, \ldots, n\}$, ergeben sich aus den Beschränkungen $(z_i \bowtie_i c_i)$ der Formelmenge 43 beziehungsweise aus dem jeweiligen Nichtvorhandensein einer solchen. Die abgebildete initiale Zuweisung für alle Variablen $v \in V$ ist $\alpha(v) := 0$.

Worüber wir mit der in Abbildung 19 dargestellten Datenstruktur somit insbesondere verfügen, sind die drei aktualisierbaren *Funktionen* u, l und α, welche uns zu jeder Variable $v \in V$ den Wert ihrer Obergrenze als $u(v)$ oder den Wert ihrer Untergrenze $l(v)$ angeben, sowie jederzeit, *via* $\alpha(v)$, den Wert der aktuellen Zuweisung, der Belegung, der Variable v zur Verfügung stellen. Da für die im ursprünglichen Ungleichungssystem vorhandenen Variablen $x \in X$ keine Ober- oder Untergrenzen bekannt sind, werden diese jedes Mal als $u(x) := \infty$ und $l(x) := -\infty$ angenommen. Für die von uns neu eingeführte Variablenmenge $Z := \{z_1, \ldots, z_i, \ldots, z_m\}$ ergibt sich die Obergrenze von z_i, das ist, $u(z_i)$, dann als $u(z_i) = d_{iu}$, falls die Ungleichung, die Beschränkung, $(z_i \leq c_i)$, bei der $c_i = d_{iu}$, in der Formelmenge 43 auftaucht; analog ergibt sich $l(z_i) = d_{il}$ falls die Ungleichung $(z_i \geq c_i)$, bei der $c_i = d_{il}$ gilt, dort erscheint.

Die initiale Belegung aller Variablen $v \in V = (X \cup Z)$ mit 0, also $\alpha(v) := 0$, welche der Auswertungsvorschrift $[x_1 \leftarrow 0, \ldots, x_n \leftarrow 0, z_1 \leftarrow 0, \ldots, z_m \leftarrow 0]$ entspricht, erzeugt eine *Invariante*. Unter einer Invariante verstehen wir einen Satz der ursprünglich wahr ist und der – was auch immer wir sonst „in der Welt“ anstellen mögen – unverändert, „invariant“, wahr bleiben soll. Dieser Satz lautet hier nun wie folgt:

> Jeder der durch die Gleichungen 44 jeweils ausgedrückten Sätze ist unter der Belegung, welche in α spezifiziert ist, wahr.

Oder – die rhetorische Figur eines Fliesenkleberherstellers, varierend, dessen Nutzungshinweis zu diesem Produkt bemerkt, dass der Fliesenkleber 24 Stunden zum Trocknen braucht und der Fliesenkleber 24 Stunden zum Trocknen braucht, um zu

betonen, dass der Kleber 24 Stunden zum Trocknen braucht –, was exakt dasselbe sagt, die Invariante lautet:

> Jeder der durch die Gleichungen 44 jeweils ausgedrückten Sätze ist unter der Belegung, welche in α spezifiziert ist, wahr.

Dass dies für die Initialbelegung gilt, erhellt aus einer beliebigen Gleichung, welche in einer i-ten Zeile des Start-Tableaus kompaktifiziert dargestellt ist:

$$\underbrace{\overbrace{z_i}^{\alpha(z_i):=0}}_{0} = \underbrace{\overbrace{c_{i1}}^{\alpha(x_1):=0} \quad \overbrace{x_1}^{\alpha(x_1):=0} + \ldots + \overbrace{c_{ij}}^{\alpha(x_j):=0} \quad \overbrace{x_j}^{\alpha(x_j):=0} + \ldots + \overbrace{c_{im}}^{\alpha(x_m):=0} \quad \overbrace{x_m}^{\alpha(x_m):=0}}_{0}$$

Führt nun die initiale Variablenbelegung α bereits dazu, dass für alle Variablen v, mit $v \in V = (X \cup Z)$, gilt, dass durch diese Zuweisung keine Ober- oder Untergrenze verletzt wird, womit also $l(v) \leq \alpha(v) \leq u(v)$ wahr ist, so ist diese Belegung zugleich eine Lösung des ursprünglichen Systems der Linearen Ungleichungen, da unter dieser Belegung insbesondere alle Gleichungen und Ungleichungen in der Formelmenge 43 wahr sind, welche ja lediglich die äquivalente Reformulierung des Ausgangsproblems darstellt. Allerdings dürfte sich dieser Umstand in den seltensten Fällen bereits nach der Anfangszuweisung ereignen, im Besonderen da für alle Variablen z_i der Menge Z, etwa eine, von ∞ verschiedene, Obergrenze d_{iu} von der konkreten Ungleichung, $(\sum_{c_i \in \mathbb{D},\, x_i \in V} c_i \cdot x_i \leq c)$, bei der $c =: d_{iu}$, des Systems abhängig ist. Es kann nämlich durchaus c und damit d_{iu} kleiner als 0 sein, und, wegen $\alpha(z_i) := 0$ und $u(z_i) := d_{iu}$, somit $\neg(\alpha(z_i) \leq u(z_i))$, womit die Zuweisung größer als die erlaubte Obergrenze ist. Analoge Überlegungen gelten für eine Verletzung der Belegung mit Bezug auf eine Untergrenze größer als 0, sodass für die initiale Zuweisung von 0 an alle Variablen $v \in V$ im Normalfall nicht sichergestellt ist, dass $l(v) \leq \alpha(v) \leq u(v)$ gilt.

Die Aufgabe, welche erfolgreich zu lösen sich der gleich vorzustellende Simplexalgorithmus anschickt, lässt sich demnach so pointieren: Es ist, für den allgemeinen Fall, eine Belegung für α, falls eine solche existiert, so zu finden, dass für sie die Invariante – die durch α beschriebene Variablenzuweisung führt zur Wahrheit aller im Tableau kodierten Gleichungen – gilt *und* durch die zugleich keine Variable einen Wert zugewiesen bekommt, der ihre jeweilige Ober- oder Untergrenze verletzen würde. Dies – so hatten wir im Spezialfall für eine dies schon gewährleistende Initialbelegung bereits dargelegt – gibt uns, in allen Fällen, wo eine solche existiert, eine Lösung für das, durch das System Linearer Ungleichungen beschriebene, ursprüngliche Erfüllungs- respektive *SMT*-Problem, oder, falls keine Lösung existiert, die Information der Unerfüllbarkeit dieses Problems.

7.5 Die Funktionsweise des Simplexalgorithmus

Zum Zwecke ihrer Instrumentalisierung ist es sachdienlich, sich einiger Grundwahrheiten der Theorie, kurz: Theoreme, der Reellen Zahlen, zu entsinnen, wel-

che wir hier als bekannt vorausgesetzt sehen wollen. Wir gehen für das Folgende stets davon aus, dass der Gegenstandsbezug aller, in den Termen möglicherweise vorkommenden, Variablen, etwa durch eine entsprechende Belegung, feststeht. Statt, exakterweise, „$\models_{\mathbb{R}}$“ und „$\equiv_{\mathbb{R}}$“, der Logischen Implikation respektive Logischen Äquivalenz *modulo* der Theorie der Reellen Zahlen, „$\mathbb{R}$“, schreiben wir nur kurz „$\models$“ und „$\equiv$“. Aus den Axiomen der Reellen Zahlen folgt – wobei wir an dieser Stelle auf den nichttrivialen Nachweis dieser Tatsache aus Platzgründen verzichten wollen –, dass wenn zwei Terme t_1 und t_2 denselben Gegenstand referenzieren, also der Satz $t_1 = t_2$ wahr ist, dann wiederum $s \cdot t_1$ und $s \cdot t_2$, wobei s ebenfalls einen Term und „$\cdot$“ die Multiplikationsfunktion bezeichnet, denselben, nicht notwendigerweise ursprünglichen, Gegenstand referenzieren: $(t_1 = t_2) \models (st_1 = st_2)$. Dasselbe gilt nun auch für den durch $(1/s)$ bedeuteten inversen Gegenstand, welcher nur definiert ist, wenn s nicht die 0 bezeichnet:

$$((t_1 = t_2) \wedge \neg(s = 0)) \quad \models \quad ((1/s) \cdot t_1 = (1/s) \cdot t_2). \tag{45}$$

Des Weiteren, wiederum ohne Nachweis von unserer Seite, folgt aus den Axiomen, dass

$$(t_1 = t_2) \equiv (s + t_1 = s + t_2), \text{ beziehungsweise } (t_1 = t_2) \equiv (s - t_1 = s - t_2), \tag{46}$$

wobei $-t_1$ und $-t_2$, die, bezüglich der Additionsfunktion, „$+$“, inversen Gegenstände zu den durch t_1 respektive t_2 bezeichneten Gegenständen sind und die Additionszeichen im Ausdruck $(s + -t_1 = s + -t_2)$ weggelassen wurden. Eines weiteren Theorems der Theorie der Reellen Zahlen hatten wir uns früher bereits bedient. Demnach bezeichnet ein Term t, der einen, nicht notwendigerweise echten, Subterm s enthält, der denselben Gegenstand wie ein weiterer Term s' referenziert, für den also gilt $s = s'$, denselben Gegenstand, wie der Term t', der aus t dadurch hervorgeht, dass in ihm ein Vorkommen von s durch s' ersetzt wurde. Insbesondere gilt für alle Terme r, s, s' und $t, t_1, \ldots, t_n$, wobei $\bowtie \in \{\leq, =, \geq\}$ und $\star, \star_1, \ldots, \star_n \in \{+, \cdot\}$, dass

$$((t \bowtie (t_1 \star_1 \cdots (s \star r) \cdots \star_n t_n) \wedge (s = s')) \quad \models \quad (t \bowtie (t_1 \star_1 \cdots (s' \star r) \cdots \star_n t_n)). \tag{47}$$

Dadurch sind insonderheit auch termvereinfachende Zusammenfassungen möglich. Beispielsweise ist der Satz, sofern eine Variablenbelegung für x, y und z bekannt ist, $(2x + 3x) \cdot y \leq 4z$ wahr genau dann, wenn der Satz $5x \cdot y \leq 4z$, der die, in der Theorie der Reellen Zahlen tautologische, Identität von $2x + 3x = 5x$ implizit supponiert, wahr ist. Diese Termvereinfachungen, kurz „Rechenregeln“, setzen wir als bekannt und gültig voraus. Es sei schließlich betont, dass die durch Formel 45 und Formel 46 beschriebenen Implikationen respektive Tautologien, modulo der Theorie der Reellen Zahlen, als *wahrheitserhaltende Termumformungen der Identität* und die durch Formel 47 beschriebene Gesetzmäßigkeit als *wahrheitserhaltende Termersetzungen* bezeichnet werden können.

Für den Simplexalgorithmus ist im Speziellen der Wahrheitserhalt der Wahrheit aller Gleichungen 44 unter der Belegung α – der Invariante – von zentraler Bedeutung, welche, wie gesehen, für die Initialbelegung $\alpha(v) := 0$ für alle Variablen $v \in V$, gilt. Eine die Wahrheit der Invariante erhaltende Transformation in diesem Sinne ist der oben bereits angedeutete *Basiswechsel* der Basis $B = (B_1, \ldots, B_n)$ und nicht Nichtbasis $N = (N_1, \ldots, N_m)$ hin zur Basis $B := (B_1, \ldots, B_{i-1}, N_j, B_{i+1}, \ldots, B_n)$ und neuer Nichtbasis $N := (N_1, \ldots, N_{j-1}, B_i, N_{j+1}, \ldots, N_m)$, in der die Basisvariable B_i zu einer Nichtbasisvariable und die Nichtbasisvariable N_j zu einer Basisvariable wird. Dieser, die Invariante erhaltende, Basiswechsel – der von den Autoren des Reluplexalgorithmus, [KBD+17], *Pivotisierung*, englisch *pivoting*, genannt wird, womöglich da die Variable, um die Würdigkeit zu haben, von einer Nichtbasisvariable zu einer Basisvariable werden zu dürfen, hervorstechend und wichtig, englisch *pivotal*, sein muss – geht mit einer Aktualisierung der Tableau-Belegung einher. Es folgen die, möglicherweise weniger unterhaltsamen, Details.

Ein Basiswechsel der Nichtbasisvariablen N_j in die Basis anstelle der Basisvariablen B_i ist nur möglich, wenn der Koeffizient T_{ij} von 0 verschieden ist. Dies setzen wir im Folgenden stets als gegeben voraus. Der Wert der Variablen B_i ist durch die entsprechende Gleichung in den Gleichungen 44 angegeben als

$$B_i = T_{i,1}N_1 + \cdots + T_{i,j-1}N_{j-1} + T_{i,j}N_j + T_{i,j+1}N_{j+1} + \cdots + T_{i,m}N_m, \tag{48}$$

woraus, mit der die Wahrheit der Identität erhaltenden additiven Termerweiterung gemäß Formel 46, folgt, dass

$$0 = T_{i,1}N_1 + \cdots + T_{i,j-1}N_{j-1} + -1 \cdot B_i + T_{i,j+1}N_{j+1} + \cdots + T_{i,m}N_m + T_{i,j}N_j. \tag{49}$$

Die divisive Termerweiterung um den Term $(1/(-T_{ij}))$, welche die Wahrheit der Identität 49 nach Implikation 45 erhält, da nach Voraussetzung $\neg(-T_{ij} = 0)$ gilt, impliziert, dass

$$0 = \frac{T_{i,1}}{-T_{i,j}}N_1 + \cdots + \frac{T_{i,j-1}}{-T_{i,j}}N_{j-1} + \frac{1}{T_{i,j}} \cdot B_i + \frac{T_{i,j+1}}{-T_{i,j}}N_{j+1} + \cdots + \frac{T_{i,m}}{-T_{i,j}}N_m + -N_j \tag{50}$$

und somit, unter erneuter Ausnutzung der Implikationsrelation in Formel 46 im Hinblick auf Additionsterm $+N_j$, dass

$$\overbrace{N_j}^{B_i':=} = \frac{\overbrace{T_{i,1}}^{T_{i,1}':=}}{-T_{i,j}}\overbrace{N_1}^{N_1':=} + \cdots + \frac{\overbrace{T_{i,j-1}}^{T_{i,j-1}':=}}{-T_{i,j}}\overbrace{N_{j-1}}^{N_{j-1}':=} + \frac{\overbrace{1}^{T_{i,j}':=}}{T_{i,j}} \cdot \overbrace{B_i}^{N_j':=} + \frac{\overbrace{T_{i,j+1}}^{T_{i,j+1}':=}}{-T_{i,j}}\overbrace{N_{j+1}}^{N_{j+1}':=} + \cdots + \frac{\overbrace{T_{i,m}}^{T_{i,m}':=}}{-T_{i,j}}\overbrace{N_m}^{N_m':=}. \tag{51}$$

Man beachte, dass – wohlgemerkt bisher nur – für die Zeile i der Basiswechsel bereits wahrheitserhaltend stattgefunden hat, da N_j die Position in B einnimmt, die zuvor B_i hatte und B_i in N wiederum die Stelle, die zuvor N_j innehatte. Um den Basiswechsel für alle anderen der Gleichungen 44 wahrheiterhaltend zu gestalten, benötigen wir die Äquivalenz 47, die uns die Substitution des Terms N_j in allen

anderen, die Basisvariablen B_k, für $k \neq i$, beschreibenden, Gleichungen durch die rechte Seite der in Formel 51 beschriebenen Gleichung erlaubt. Konkret gilt, dass in

$$B_k = T_{k,1}N_1 + \cdots + T_{k,j-1}N_{j-1} + T_{k,j}N_j + T_{k,j+1}N_{j+1} + \cdots + T_{k,m}N_m, \tag{52}$$

der Term N_j entsprechend durch

$$\frac{T_{i,1}}{-T_{i,j}}N_1 + \cdots + \frac{T_{i,j-1}}{-T_{i,j}}N_{j-1} + \frac{1}{T_{i,j}} \cdot B_i + \frac{T_{i,j+1}}{-T_{i,j}}N_{j+1} + \cdots + \frac{T_{i,m}}{-T_{i,j}}N_m$$

ersetzt werden darf, was zu

$$\begin{aligned} B_k = {} & T_{k,1}N_1 + \cdots + T_{k,j-1}N_{j-1} + \\ & T_{k,j}\overbrace{\left(\frac{T_{i,1}}{-T_{i,j}}N_1 + \cdots + \frac{T_{i,j-1}}{-T_{i,j}}N_{j-1} + \frac{1}{T_{i,j}} \cdot B_i + \frac{T_{i,j+1}}{-T_{i,j}}N_{j+1} + \cdots + \frac{T_{i,m}}{-T_{i,j}}N_m\right)}^{=N_j} \\ & + T_{k,j+1}N_{j+1} + \cdots + T_{k,m}N_m, \end{aligned} \tag{53}$$

führt. Ausmultiplizieren bring uns zu

$$\begin{aligned} \overbrace{B_k}^{B'_k:=} = {} & \overbrace{\left(T_{k,1} + \frac{T_{k,j}}{-T_{i,j}}T_{i,1}\right)}^{T'_{k,1}:=}\overbrace{N_1}^{N'_1:=} + \cdots + \overbrace{\left(T_{k,j-1} + \frac{T_{k,j}}{-T_{i,j}}T_{i,j-1}\right)}^{T'_{k,j-1}:=}\overbrace{N_{j-1}}^{N'_{j-1}:=} + \\ & \overbrace{\frac{T_{k,j}}{T_{i,j}}}^{T'_{k,j}:=}\overbrace{B_i}^{N'_j:=} + \\ & \overbrace{\left(T_{k,j+1} + \frac{T_{k,j}}{-T_{i,j}}T_{i,j+1}\right)}^{T'_{k,j+1}:=}\overbrace{N_{j+1}}^{N'_{j+1}:=} + \cdots + \overbrace{\left(T_{k,m} + \frac{T_{k,j}}{-T_{i,j}}T_{i,m}\right)}^{T'_{k,m}:=}\overbrace{N_m}^{N'_m:=}, \end{aligned} \tag{54}$$

was die ehemalige Basisvariable B_i in allen anderen, für $k \neq i$, die Basisvariablen B_k beschreibenden Gleichungen an die Position setzt, welche vorher die, nunmehr zur Basis zu wechseln sich anschickende, Variable N_j innehatte, was den, die Wahrheit der Invariante erhaltenden, Basiswechsel komplettiert, sofern die Datenstrukturen B und N und auch das Tableau aktualisiert wird, indem, für alle $i \in \{1, \ldots, n\}$ und alle $j \in \{1, \ldots, m\}$, überall B_i und N_j so aktualisiert werden, wie in Gleichung 51 und Gleichung 54 angegeben sowie an jeder Stelle der Wert $T_{i,j}$ durch den Wert $T'_{i,j}$ so, wie in Gleichung 51 und Gleichung 54 spezifiziert ersetzt wird.

Um eine Kurzform zu haben für den eben beschriebenen, die Wahrheit der Invariante konservierenden, Basiswechsel von Basisvariable B_i und Nichtbasisvariable N_j zu haben, definieren wir die Funktion `pivot(T, i, j)`, welche den Basiswechsel

vollzieht, indem sie die entsprechenden Aktualisierungen, (T', B', N'), des Tableaus T sowie der Tupel B und N vornimmt:

$$\texttt{pivot}(\texttt{T}, \texttt{i}, \texttt{j}) = (T', B', N')$$

Die Anwendung der `pivot`-Funktion ist im Hinblick auf die Invariante also jederzeit harmfrei.

Eine zweite in dieser Hinsicht harmlose Manipulation betrifft den zweiten kritischen Punkt der Invariante, der für die Gleichungen 44 potentiell wahrheitswertverändernd ist: Eine Änderung der Zuweisung α. Für alle Fälle, in denen diese Aktualisierung auf eine Inkrementierung der Belegung $\alpha(N_j)$ einer Nichtbasisvariablen N_j auf $\alpha'(N_j) := \alpha(N_j) + \delta$, wobei $\delta \in \mathbb{D}$ nicht notwendigerweise nichtnegativ sein muss, zielt, beschreibt die Funktion

$$\texttt{update}(\alpha, \texttt{N}_\texttt{j}, \delta) = \alpha'$$

kompakt, wie diese Änderung der Zuweisung bei gleichzeitigem Erhalt der Invariante vonstatten geht. Die diesbezüglichen Details, welche auch beschreiben, wie sich dadurch die Belegung der Basisvariablen ändert, sind die Folgenden.

Betrachten wir die i-te der Gleichungen 44, bei denen alle Variablen durch die in α spezifizierte gegenwärtige Variablenzuweisung substituiert wurden; der dadurch ausgedrückte Satz ist wahr, da die Invariante für die Belegung α wahr ist.

$$\alpha(B_i) = T_{i,1}\alpha(N_1) + \cdots + T_{i,j} \overbrace{\alpha(N_j)}^{=\alpha'(N_j)-\delta} + \cdots + T_{i,m}\alpha(N_m), \tag{55}$$

wobei man beachte, dass wenn der zukünftige Belegungswert von N_j der Wert $\alpha'(N_j) = \alpha(N_j) + \delta$ sein soll, der alte Zuweisungswert natürlich $\alpha(N_j) = \alpha'(N_j) - \delta$ sein muss, was uns eine Instanz für die Äquivalenz 47 zur Termsubstitution gibt. Die entsprechende wahrheitskonservierende Termersetzung liefert

$$\alpha(B_i) = T_{i,1}\alpha(N_1) + \cdots + T_{i,j}\alpha'(N_j) + \cdots + T_{i,m}\alpha(N_m) - \delta T_{i,j}, \tag{56}$$

und im Hinblick auf die in Formel 46 beschriebene Äquivalenz, bezüglich Additionsterm $\delta T_{i,j}$, dass

$$\overbrace{\alpha(B_i) + \delta T_{i,j}}^{\alpha'(B_i):=} = T_{i,1} \overbrace{\alpha(N_1)}^{\alpha'(N_1):=} + \cdots + T_{i,j} \overbrace{\alpha'(N_j)}^{(\alpha(N_j)+\delta)=:} + \cdots + T_{i,m} \overbrace{\alpha(N_m)}^{\alpha'(N_m):=}, \tag{57}$$

was bedeutet, dass auch unter der neuen Zuweisung α', wie in Gleichung 57 beschrieben, die Wahrheit aller Gleichungen 44 und damit die Invariante gilt.

Mit den Funktionen `pivot` und `update` stehen dem Simplexalgorithmus damit zwei Instrumente zur Verfügung mithilfe derer er, im Falle von `pivot`, beliebige Basisvariablen $v \in V$ aus der Basis heraustauschen darf, um in einem anschließenden Schritt, mittels `update`, eine Obergrenzenüberschreitung, $\alpha(v) > u(v)$, oder Untergrenzenunterschreitung, $\alpha(v) < l(v)$, durch ein „Update“ von α zu $\alpha'(v) = \alpha(v) + \delta$

so zu korrigieren, dass $l(v) \leq \alpha'(v) \leq u(v)$, womit die Zuweisung dieser Variablen in ihren legalen Grenzen liegt – wohlgemerkt ohne dass dabei die Invariante verletzt werden könnte.

Eine solche `update`-Operation, die auf eine Wertzuweisung an Nichtbasisvariable v dergestalt zielt, dass deren Inkrementierung um δ Einheiten zu einem Wert zwischen Ober- und Untergrenze führt, ist, wie man sich klar macht, offenbar nur sinnvoll, wenn die mit dieser Aktualisierung einhergehende Veränderung der Zuweisung der Basisvariablen B_i um $+\delta T_{i,j}$ Einheiten – man vergleiche die linke Seite der Gleichung, welche Formel 57 darstellt – dem eigentlichen Ziel, nämlich dafür zu sorgen, dass unter Beachtung der Invariante, *alle* Variablen ihre jeweiligen Ober- und Untergrenzen einhalten, dienlich ist. Bevor also eine Basisvariable B_i, deren gegenwärtige Zuweisung ihre Untergrenze unterschreitet, $\alpha(B_i) < l(B_i)$ aus der Basis gegen die Nichtbasisvariable N_j herausgetauscht wird, muss diese Nichtbasisvariable N_j – jede Basisvariable ist ja die Linearkombination aller Nichtbasisvariablen –, die nach dem Heraustauschen anvisierte Inkrementierung um $\delta > 0$, zumindest prinzipiell verkraften könnte, ohne selbst ihrerseits ihre Ober- oder Untergrenze zu verletzen, was nur in zwei Fällen gewährleistet ist. Der erste Fall lässt sich so formalisieren:

$$\overbrace{\alpha(B_i)}^{<l(B_i)} = T_{i,1}\alpha(N_1) + \cdots + \overbrace{T_{i,j}}^{>0}\overbrace{\alpha(N_j)}^{<u(N_j)} + \cdots + T_{i,m}\alpha(N_m). \tag{58}$$

Demnach ist der Koeffizient $T_{i,j}$ von Nichtbasisvariable N_j positiv, $T_{i,j} > 0$, und die Wertzuweisung von N_j könnte noch erhöht werden, da die zu N_j gehörende Obergrenze noch nicht erreicht ist: $\alpha(N_j) < u(N_j)$. Im zweiten Fall,

$$\overbrace{\alpha(B_i)}^{<l(B_i)} = T_{i,1}\alpha(N_1) + \cdots + \overbrace{T_{i,j}}^{<0}\overbrace{\alpha(N_j)}^{>l(N_j)} + \cdots + T_{i,m}\alpha(N_m). \tag{59}$$

ist der Nichtbasisvariablenkoeffizient $T_{i,j}$ negativ, aber die Untergrenze noch nicht erreicht – man beachte, dass die Vergrößerung des Wertes $\alpha(N_j)$ in diesem Fall eine Verkleinerung des Produkts $T_{i,j}\alpha(N_j)$ impliziert. In beiden Fällen gibt es eine Variable, N_j, die eine Vergrößerung der Zuweisung von B_i prinzipiell verkrafteten könnte, ohne dabei – durch eine entsprechende Änderung ihrer Zuweisung – notwendigerweise eine ihrer Grenzen verletzen zu müssen.

Die Menge aller Nichtbasisvariablen N_j, die, bezogen auf die Basisvariable B_i dieses Kriterium – den Bedarf von B_i, den Wert der ihr zugewiesenen Belegung zu erhöhen – erfüllen, ist die durch die Funktion slack^+ beschriebene Menge

$$\mathrm{slack}^+(B_i) = \{N_j \in N \mid \big((T_{i,j} > 0) \wedge (\alpha(N_j) < u(N_j))\big) \vee \big((T_{i,j} < 0) \wedge (\alpha(N_j) > l(N_j))\big)\}.$$

Analog ist die Menge aller Nichtbasisvariablen N_j, die, bezogen auf die Basisvariable B_i das Kriterium, den Bedarf von B_i, den Wert der ihr zugewiesenen Belegung zu verringern, erfüllen, die durch die Funktion slack^- beschriebene Menge

$$\mathrm{slack}^-(B_i) = \{N_j \in N \mid \big((T_{i,j} < 0) \wedge (\alpha(N_j) < u(N_j))\big) \vee \big((T_{i,j} > 0) \wedge (\alpha(N_j) > l(N_j))\big)\},$$

wie aus der Betrachtung des, analog zu Formel 58 und Formel 59, wie folgt beschriebenen Sachverhalts hervorgehen sollte.

$$\overbrace{\underbrace{\alpha(B_i)}_{>u(B_i)}}^{>u(B_i)} = T_{i,1}\alpha(N_1) + \cdots + \overbrace{\underbrace{T_{i,j}}_{>0}\underbrace{\alpha(N_j)}_{>l(N_j)}}^{<0\ <u(N_j)} + \cdots + T_{i,m}\alpha(N_m). \tag{60}$$

Sollte – dieselbe Tatsache unter dem entgegengesetzten Blickwinkel betrachtend – eine Basisvariable B_i Bedarf zur Vergrößerung haben, da $\alpha(B_i) < l(B_i)$, und es gibt keine Nichtbasisvariable N_j, die diesen Bedarf kompensieren könnte, ohne dabei eine eigene Grenze zu verletzen, kurz: $\text{slack}^+ = \emptyset$, so können nicht alle Variablenbelegungen gleichzeitig ihre Ober- und Untergrenzen einhalten, ohne dass die Invariante verletzt werden würde. Analoge Überlegungen führen zum selben Ergebnis für $\alpha(B_i) > u(B_i)$ und den Fall, dass $\text{slack}^- = \emptyset$. In beiden Fällen kann das ursprüngliche System Linearer Ungleichungen keine Lösung haben und der Simplexalgorithmus wird mit dem Ergebnis UNSAT terminieren, welches uns über die Unlösbarkeit informiert. Ist hingegen der Fall eingetreten, dass, nach einer Reihe von `pivot`- und `update`-Operationen, alle Variablen in ihren Grenzen liegen, also $l(v) \leq \alpha(v) \leq u(v)$ für alle Variablen $v \in V = \{x_1, \ldots, x_m, z_1, \ldots, z_n\}$ gilt, so wird – da die, seit der Initialisierung aller Variablen auf den Zuweisungswert 0 gültige, Invariante durch diese Operationen schlechterdings nicht verletzt werden kann – diese α-Belegung als Beweis der Lösbarkeit des ursprünglichen Systems Linearer Ungleichungen zusammen mit dem Hinweis SAT vom Simplexalgorithmus zurückgegeben.

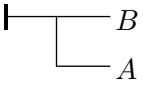
Der Simplexalgorithmus wird nun, zusammenfassend, durch die Anwendung einer Reihe von Regeln beschrieben, die das oben erläuterte Vorgehen kompakt zusammenfassen. Durch ebendiese Anwendung werden die Elemente des Tupels $S = (B, N, T, l, u, \alpha)$ – welches im Reluplexaufsatz, [KBD+17], *configuration*, „Konfiguration“, heißt, wobei S somit den gegenwärtigen „Zustand“, englisch *state*, beschreibt – potentiell aktualisiert. Die Anfangskonfiguration $S = (B, N, T, l, u, \alpha)$ umfasst die, wie oben beschriebenen, Initialwerte für B und N, das initiale Tableau für T, wie in Abbildung 18 dargelegt und die Anfangszuweisung α, welche allen Variablen den Wert 0 zukommen lässt. Die Ober- und Untergrenzen u und l sind genau diejenigen, welche initial in die durch Abbildung 19 beschriebene Datenstruktur insertiert wurden.

Eine Regel namens R leitet aus einer Konfiguration S, die mit dem Bedingungsteil C nicht im Widerspruch steht – mit dieser „verträglich“ ist –, diejenige Konfiguration S' ab, die im Konsequenzteil der Regel steht. In Kurznotation – in Reverenz an Gottlob Frege, der in seiner 1879 erschienenen „Begriffsschrift“, [Fre79], diese Art der Notation, die das Zeichen „$-$“, den „Inhaltsstrich“, das eine „blosse Vorstellungsverbindung“, also *einen Gedanken* bezeichnet, mit dem bejahenden Urteil, *der Behauptung des Gedankens*, welche, durch einen senkrechten Strich „$|$“ ausgedrückt wird und noch vor dem Gedankenstrich steht, verbindet, einführte – schreiben wir dies als $\mathsf{R}\colon (S, C) \vdash S'$, was wie folgt gelesen werden kann: „Besteht der durch

die Konfiguration S beschriebene Sachverhalt, welcher die in C dargelegten Bedingungen erfüllt, so darf auf das Bestehen des durch die Konfiguration S' beschriebenen Sachverhalts geschlossen werden.“ oder, kürzer: „Die Regel R *leitet* aus S und C die Konfiguration S' *ab*.“, weshalb Regeln der Art von R auch „Ableitungsregeln“, englisch *derivation rules*, genannt werden. Dieser Vorgang, das „Ableiten“, englisch *derivation*, bezeichnet dabei konkret die dementsprechende, durch Regeln vermittelte, Überführung einer Konfiguration S, in eine Konfiguration S' oder die Ableitung einer Konfiguration S'', die, gegebenenfalls rekursiv, eine Konfiguration S' ableitet. Für die großflächige Repräsentation dieser Regel, schreiben wir, bei Wegfall des Urteilsstrichs, den *Bedingungsteil einer Regel*, welcher ursprünglich links des „$\vdash$“-Zeichens steht, über den Gedankenstrich, welcher dadurch dazu neigt etwas länger auszufallen, und den *Konsequenzteil einer Regel*, ursprünglich rechts von „$\vdash$“ stehend, unter den Gedankenstrich:

$$\mathsf{R} \quad \frac{S, C}{S'}$$

Frege hätte diese, durch R ausgedrückte, Bedingtheit – wonach, in annähernd den Worten der Begriffschrift, [Fre79], „das Urteil, dass die Möglichkeit, dass $A := S, C$ bejaht wird und $B := S'$ verneint wird, nicht stattfindet“ gefällt wird –, der historischen Korrektheit willen, übrigens – der weitere senkrechte Strich ist der „Bedingungsstrich“ („B, unter der Bedingung, dass A bejaht wird, wird bejaht“) – lieber als



ausgedrückt gewusst, was hier jedoch nicht weiter vertieft werden soll.

Kommen wir endlich zu den Regeln, welche für den Simplexalgorithmus maßgeblich sind. Die Regeln, welche den Basiswechsel realisieren, heißen im Reluplexaufsatz, [KBD$^+$17], Pivot_1 und Pivot_2. Sie sind wie folgt beschrieben.

$$\mathsf{Pivot}_1 \quad \frac{S = (B, N, T, l, u, \alpha), \quad B_i \in B, \quad \alpha(B_i) < l(B_i), \quad N_j \in \mathrm{slack}^+(B_i)}{\texttt{pivot(T, i, j)} = (T', B', N'), \quad S' = (B', N', T', l, u, \alpha)} \tag{61}$$

Der Bedingungsteil der Regel 61, Pivot_1, beschreibt exakt den oben diskutierten Fall, in dem es in der gegenwärtigen Konfiguration S eine Basisvariable B_i gibt, deren gegenwärtige α-Zuweisung, die Untergrenze von B_i verletzt. Gleichzeitig existiert eine Nichtbasisvariable N_j, die eine positive Inkrementierung der B_i-Zuweisung, im Hinblick auf die Invariante, kompensieren könnte, denn N_j liegt in $\mathrm{slack}^+(B_i)$, der Menge aller, bezüglich B_i, so beschriebenen Nichtbasisvariablen. Im Konsequenzteil der Regel, welcher unterhalb des Urteilsstrichs steht, wird, durch die Anwendung der `pivot`-Funktion, der Basiswechsel von B_i und N_j vollzogen, wobei die Basis- und Nichtbasisvariablentupel B und N zu B' und N' ebenso wie die Koeffizienten des Tableaus T zu denen in T' aktualisiert werden. Man beachte auch, dass die Anwendung dieser Regel die Zuweisung α und die Ober- und Untergrenzen u

und l unverändert lässt. Analog beschreibt die Regel Pivot_2 den durch eine Überschreitung der Obergrenze einer Basisvariablen B_i nötig werdenden Basiswechsel mit einer Nichtbasisvariablen N_j, die eine Zuweisungsdekrementierung prinzipiell verkraften könnte.

$$\mathsf{Pivot}_2 \quad \frac{S = (B, N, T, l, u, \alpha), \quad B_i \in B, \quad \alpha(B_i) > u(B_i), \quad N_j \in \text{slack}^-(B_i)}{\texttt{pivot}(\texttt{T}, \texttt{i}, \texttt{j}) = (T', B', N'), \quad S' = (B', N', T', l, u, \alpha)} \tag{62}$$

Die Regel, welche die beschriebene, durch die Funktion `update` realisierte, „Zuweisungswertkorrektur“ einer Nichtbasisvariablen vornimmt, heißt im Aufsatz, welcher diejenige Version des Simplexalgorithmus erläutert, auf welcher der Reluplexalgorithmus aufbaut, [KBD+17], ebenfalls Update und kann, nach dem oben Diskutierten, wie folgt dargestellt werden.

$$\mathsf{Update} \quad \frac{\begin{array}{c} S{=}(B,N,T,l,u,\alpha), \quad N_j{\in}N, \quad (\alpha(N_j) < l(N_j)) \vee (\alpha(N_j) > u(N_j)), \\ l(N_j) \leq (\alpha(N_j){+}\delta) \leq u(N_j) \end{array}}{\texttt{update}(\alpha, \texttt{N}_\texttt{j}, \delta) = \alpha', \quad S' = (B, N, T, l, u, \alpha')} \tag{63}$$

Dass `update`-Operationen nur auf Nichtbasisvariablen $N_j \in N$ durchgeführt werden dürfen, deren Zuweisung zusätzlich dazu eine Verletzung der Untergrenze, $(\alpha(N_j) < l(N_j))$, oder der Obergrenze, $(\alpha(N_j) > u(N_j))$, aufweisen muss, wobei ein Wert δ dieser Verletzung abhelfen würde, $(l(N_j) \leq (\alpha(N_j) + \delta) \leq u(N_j))$, ist im Bedingungsteil der Update-Regel spezifiziert. Die, als Konsequenz des Vorliegens der Bedingung, zu applizierende Funktion `update` führt diese Amelioration, welche eine Änderung der Zuweisungsfunktion α zu α' impliziert, herbei, was im Konsequenzteil der Regel formal festgehalten ist.

Die folgenden Terminationsregeln legen für den Simplexalgorithmus fest, wann die Berechnung einer Lösung abgeschlossen ist oder die Berechnung abgebrochen werden kann, da ruchbar geworden ist, dass es keine Lösung geben kann. Den ersten Fall konkretisiert die Success genannte Regel so.

$$\mathsf{Success} \quad \frac{S = (B, N, T, l, u, \alpha), \quad \forall v \colon (v \in (B \cup N)) \Rightarrow (l(v) \leq \alpha(v) \leq u(v))}{\text{SAT}} \tag{64}$$

Wenn keine Basisvariable und keine Nichtbasisvariable durch die Zuweisung α keine ihrer Ober- oder Unterschranken verletzt sieht, während „im Hintergrund“ stillschweigend weiterhin die Invariante gilt, dann ist das Problem, welches durch das ursprüngliche System Linearer Ungleichungen beschrieben ist, lösbar, was durch die Regelkonsequenz SAT ausgedrückt wird, und α beinhaltet eine Lösung in Form einer konkreten Variablenzuweisung. Man beachte, dass SAT eine, sonderbare, Konfiguration ist, für die kein Bedingungsteil irgendeiner der vorgestellten Regeln erfüllt werden kann, was der „Terminierung“ des Algorithmus entspricht.

Wenn andererseits eine Konfiguration abgeleitet worden ist, aus der hervorgeht, dass die Bedingungen, welche die Anwendbarkeit der Success-Regel erlauben, nicht mehr erreicht werden können, das System also, anders ausgedrückt, keine Lösung

haben kann, sollte der Simplexalgorithmus diesem Umstand Rechnung tragen. Dies geschieht mittels der Failure-Regel.

$$\textsf{Failure}\quad \frac{\begin{array}{c} S{=}(B,N,T,l,u,\alpha),\quad B_i{\in}B, \\ \Big((\alpha(B_i){<}l(B_i)) \wedge (\mathrm{slack}^{+}(B_i){=}\emptyset)\Big) \vee \Big((\alpha(B_i){>}u(B_i)) \wedge (\mathrm{slack}^{-}(B_i){=}\emptyset)\Big) \end{array}}{\mathrm{UNSAT}} \tag{65}$$

Wie wir gesehen hatten, kann es im Besonderen dann keine Lösung geben, wenn die Zuweisung einer Basisvariablen einerseits entweder die Untergrenze verletzt, $\alpha(B_i) < l(B_i)$, aber keine Nichtbasisvariable diesem Problem abhelfen könnte, ohne, durch eine entsprechende Änderung der Zuweisung, selbst ihrerseits eine ihrer Grenzen zu verletzen: die Menge der dies bezüglich B_i leistenden Nichtbasisvariablen $\mathrm{slack}^{+}(B_i)$ ist leer, oder andererseits die Obergrenze verletzt, $\alpha(B_i) > u(B_i)$, aber die Menge $\mathrm{slack}^{-}(B_i)$ leer ist, deren Elemente andernfalls eine Verringerung des Zuweisungwertes an B_i hätten rekompensieren können. Genau diese beiden Fälle sind durch den Bedingungsteil der Regel Failure ausgedrückt, deren Konsequenzteil die einzigartige UNSAT-Konfiguration ist, welche wiederum die „Terminierung“ des Algorithmus impliziert.

Der Simplexalgorithmus selbst besteht nun darin, nach der Herstellung der Initialkonfiguration, die Regeln $\textsf{Pivot}_1$, $\textsf{Pivot}_2$ und Update so lange anzuwenden, bis durch die finale Anwendung der Regel Success entweder die Lösbarkeit des, durch das System der Linearen Ungleichungen beschriebenen, Erfüllungsproblems oder dessen Unlösbarkeit, durch letztmalige Anwendung der Failure-Regel, festgestellt wird.

Aus unserer Diskussion geht hervor, dass der so beschriebene Simplexalgorithmus, als Logisches – modulo der Theorie der Reellen Zahlen – Kalkül, *korrekt*, englisch *sound*, ist: Wann immer aus der Initialkonfiguration S, welche insbesondere die Wahrheit der Invariante etabliert, durch eine Kette der Art

$$S \vdash_{\mathsf{R}_1} S_1 \vdash_{\mathsf{R}_2} \cdots \vdash_{\mathsf{Success}} \mathrm{SAT},$$

wobei $\vdash_{R_i}$, mit $i \in \{1, 2, \ldots\}$ und $R_i \in \{\textsf{Pivot}_1, \textsf{Pivot}_2, \textsf{Update}\}$ nur die, die jeweilige Ableitung $\vdash_{\mathsf{R_i}}$ vermittelnde, Regeln sind, die insonderheit die Invariante intakt lassen, die Konfiguration SAT abgeleitet wird, dann hat das Problem tatsächlich eine Lösung und wann immer aus S, durch eine Kette

$$S \vdash_{\mathsf{R}_1} S_1 \vdash_{\mathsf{R}_2} \cdots \vdash_{\mathsf{Failure}} \mathrm{UNSAT}$$

die Konfiguration UNSAT abgeleitet wird, dann hat das Problem tatsächlich keine Lösung.

Die *Vollständigkeit* des Simplexkalküls – wonach wann immer es tatsächlich eine Lösung für das durch ein System Linearer Ungleichungen beschriebene Erfüllungsproblem gibt, der Simplixalgorithmus eine Lösung findet und wann immer es tatsächlich keine Lösung gibt, der Algorithmus konstatiert, dass es keine gibt, und nicht etwa (weil stets immer wieder nur die Regeln $\textsf{Pivot}_1$, $\textsf{Pivot}_2$ und Update angewendet werden, nie jedoch eine der Regeln Success oder Failure angewandt wird)

niemals „terminiert“ – können wir hier aus Platzgründen nicht zeigen. Dass die Vollständigkeit des Simplexkalküls dann und nur – was wiederum impliziert, dass die Reihenfolge in der die Regeln Pivot_1, Pivot_2 und Update angewendet werden, nicht in jedem Fall beliebig sein kann – unter der Voraussetzung einer geschickten Auswahl- oder, Pivotisierungs-Strategie der Variablen, den jeweils „pivotalen“ Schlüssel-Elementen, des Basistauschs, gegeben ist, ist Gegenstand praktisch jedes Lehrbuchs zur Linearen Optimierung, etwa „Linear Programming and Network Flows“, [BJS04], von Bazaara, Jarvis und Sherali. Es sei des Weiteren auch auf die im Reluplexaufsatz, [KBD+17], diesbezüglich referenzierten Quellen verwiesen.

7.6 Die Funktionsweise des Reluplexalgorithmus

Mit dem im letzten Abschnitt beschriebenen Simplexverfahren verfügen wir nun über ein korrektes und, wie wir, auch ohne dies gezeigt zu haben, annehmen wollen, vollständiges, Kalkül, welches die Wahrheit des Existenzabschluss eines atomaren Proto-Satzes oder einer Menge von, durch Konjunktionen verknüpften, atomaren Proto-Sätzen ableitet. Damit ist das Simplexverfahren bereits ein Verifizierer von Eigenschaften Neuronaler Netze, wie wir uns anhand folgender Verifikationsstrategie klar machen wollen. Wir nehmen an, dass die mit Bezug auf das Netz N, zu verifizierende Eigenschaft P durch einen atomaren Proto-Satz oder eine Konjunktion von atomaren Proto-Sätzen beschrieben ist. Diese Vereinfachung in Bezug auf die Eigenschaft P ist nicht unzulässig, da wir, für den Fall, dass P, was in der Sprache *Marabou*s ja durchaus erlaubt wäre, Disjunktionen enthält, die zu P äquivalente Disjunktive Normalform $P^{\mathrm{DNF}} = (P_1 \vee \cdots \vee P_n)$ bilden können – in der jedes Disjunkt P_i, wobei $i \in \{1, \ldots, n\}$, entweder einen atomaren Proto-Satz oder eine Konjunktion von atomaren Proto-Sätzen darstellt – und das gleich zu beschreibende Prozedere für jedes der P_i anwenden können, wobei N die Eigenschaft P genau dann hat, wenn N mindestens eine der durch P_i beschriebenen Eigenschaften hat. Nun formen wir aus den entsprechenden Daten des Netzes N, das heißt, den Gewichten, Bias und dergleichen, die Netzwerkformel 21, F_N, und reformulieren diese sofort in die zu ihr äquivalente Disjunktive Normalform, das heißt konkret, zur Formel 22, $F_N^{\mathrm{DNF}} := (F_1 \vee \cdots \vee F_{2^n})$, deren 2^n – es sei an die entsprechende Aussage des Satzes 2 erinnert – Teilformeln unter anderem die Semantik der n *ReLU*-Knoten des Netzes beinhaltet und welche jeweils nichts als Konjunktionen atomarer Proto-Sätze sind. Folglich ist auch jede der Konjunktionen $(P \wedge F_i)$, wobei $i \in \{1, \ldots, 2^n\}$, eine Konjunktion atomarer Protosätze oder, dieselbe Tatsache anders ausgedrückt, wobei die Konjunkte dieser Konjunktion als Elemente einer Menge betrachtet werden, ein System Linearer Ungleichungen. Für jedes dieser System findet der Simplexalgorithmus, so hatten wir gesehen, jeweils genau dann eine Lösung, wobei er schließlich SAT, andernfalls UNSAT, ableitet, wenn der Existenzabschluss eines Proto-Satzes $(P \wedge F_i)$ wahr ist. Findet der Simplexalgorithmus nun für mindestens eines dieser Systeme eine Lösung, so ist der durch den Existenzabschluss über $(F_N^{\mathrm{DNF}} \wedge P)$ und damit über der zu ihr äquivalenten Formel $(F_N \wedge P)$ entstehende Satz wahr, was

dasselbe bedeutet wie, dass das Netz N die Eigenschaft P hat, sonst falsch und N hat nicht die Eigenschaft P.

Diese Art des Verifizierens bringt jedoch den entscheidenden Nachteil mit sich, dass, bezogen auf die Anzahl n der *ReLU*-Knoten eines Netzes N, der Simplexalgorithmus zu jedem Verifikationsproblem, ob ein Netz N die Eigenschaft P habe, 2^n verschiedene Systeme Linearer Ungleichungen lösen muss. Die Erfinder des *Reluplex*-Algorithmus haben dieses Problem erkannt und den im letzten Abschnitt vorgestellten Simplexalgorithmus um eine *ReLU*-eigene Komponente ergänzt, der wir uns jetzt zuwenden wollen. Anstatt von der Netzwerkformel 21 auszugehen, welche die problematischen – da sie das Logische Oder in die Formel injizieren – *ReLU*-Kodierungen in Form der Formeln 14 enthält, wird der Ausgang bei der unproblematischen – da diese eine Konjunktion atomarer Protoformeln ist – Formel 11 genommen, womit zunächst noch nicht die Semantik der *ReLU*-Knoten des Netzes beschrieben ist. Man beachte aber – zu welchem Behuf sich die nochmalige Betrachtung der Formel 10 als hilfreich erweisen könnte – dass bereits durch Formel 11 *alle* Variablen des Netzes eingeführt sind, insbesondere die Variablen, $v_{\ell,i}^{\mathrm{Pre}}$ und $v_{\ell,i}^{\mathrm{Post}}$, die auch in der Beschreibung der *ReLU*-Semantik (*via* Formel 14) Verwendung finden.

Um der Semantik der *ReLU*-Knoten Rechnung zu tragen, wird eigens zu diesem Zweck eine neue Datenstruktur – die *ReLU*-Relation R – eingeführt. Wir erinnern uns, im Rückblick auf die Abbildung 2, dass jeder *ReLU*-Knoten eines Neuronalen Netzes stets durch zwei Knotenvariablen bestimmt ist. Konkret besitzt der i-te *ReLU*-Knoten der Schicht $\ell > 0$ die beiden Knotenvariablen $v_{\ell,i}^{\mathrm{Pre}}$, der für den Wert steht, den der Knoten vor der Anwendung der *ReLU*-Aktivierungsfunktion hat, und $v_{\ell,i}^{\mathrm{Post}}$, der den Wert nach der Anwendung der Aktivierungsfunktion beschreibt. Die *ReLU*-Relation R zu einem gegebenen Netz N ist nun einfach die Menge aller *ReLU*-Knoten des Netzes in Form einer Menge geordneter Paare der entsprechenden *ReLU*-Komponenten:

$$R := \{(v_{\ell,i}^{\mathrm{Pre}}, v_{\ell,i}^{\mathrm{Post}}) \mid \text{der } i\text{-te Knoten in Schicht } \ell > 0 \text{ ist ein } \textit{ReLU}\text{-Knoten in N}\}.$$

Mithilfe dieser Relation wird überwacht, dass eine Variablenbelegung α die in Formel 1, in konzisester Weise, dargestellte Semantik der *ReLU*-Funktion nicht verletzt – konkret also, dass

$$\alpha(v_{\ell,i}^{\mathrm{Post}}) = \mathrm{ReLU}(\ \alpha(v_{\ell,i}^{\mathrm{Pre}})\ ) = \max\{0, \alpha(v_{\ell,i}^{\mathrm{Pre}})\} \tag{66}$$

gilt.

Die Grundidee der *Reluplex*-Algorithmus besteht nun darin, den Simplexalgorithmus des letzten Abschnitts auf die Menge Linearer Beschränkungen, aus denen die Konjunkte der Formel 11 bestehen und die in ihrer Gesamtheit somit ein System Linearer Ungleichungen darstellen, anzuwenden. Dabei werden dieselben Datenstrukturen, allerdings ergänzt um die *ReLU*-Relation R, wie bisher verwendet: die Basis- und Nichtbasisvariablentupel B und N, das Tableaus T, die aktualisierbare Funktion α sowie die Ober- und Untergrenzen u und l. Die *Reluplex*-Konfigurationen S sind somit Tupel der Art $S = (B, N, T, l, u, \alpha, R)$. Damit ist wiederum das System der Gleichungen 44 definiert und die Invariante – Jeder der durch

die Gleichungen 44 jeweils ausgedrückten Sätze ist unter der Belegung, welche in α spezifiziert ist, wahr. – gilt, nach der Initialisierung aller Variablenwertzuweisungen auf 0, auch für den *Reluplex*-Algorithmus. Neu ist hingegen, dass der Algorithmus nur dann eine – so gesehen: „*ReLU*"- – Lösung findet, wenn nicht nur, wie bisher, keine Variable ihre Ober- oder Untergrenze verletzt, sondern darüber hinausgehend auch die, durch Formel 66 ausgedrückte, *ReLU*-Semantik für jedes Paar der *ReLU*-Relation R erfüllt ist. Da der bisher verwendete Simplexalgorithmus dies mit den ihm zur Verfügung stehenden Mitteln allein – konkret den Regeln Pivot_1, Pivot_2 und Update – nicht gewährleisten kann, sind für den *Reluplex*-Algorithmus entsprechende Regeln zu ergänzen, beziehungsweise, da diese für den Erfolgsfall die Einhaltung der *ReLU*-Semantik nicht in ihrem Bedingungsteil inkorporiert, die Success-Regel wie folgt zur Regel ReluSuccess zu adaptieren.

$$\mathsf{ReluSuccess}\quad \frac{\begin{array}{c} S{=}(B,N,T,l,u,\alpha,R),\quad \forall v\colon (v\in(B\cup N))\Rightarrow (l(v)\leq\alpha(v)\leq u(v)),\\ \forall v_{\ell,i}^{\mathrm{Pre}}\forall v_{\ell,i}^{\mathrm{Post}}\colon ((v_{\ell,i}^{\mathrm{Pre}},v_{\ell,i}^{\mathrm{Post}})\in R)\Rightarrow(\alpha(v_{\ell,i}^{\mathrm{Post}})=\max\{0,\alpha(v_{\ell,i}^{\mathrm{Pre}})\}) \end{array}}{\mathrm{SAT}} \tag{67}$$

Konkret sind als neue Regeln, neben den Regeln Update, Pivot_1 und Pivot_2, die im *Reluplex*-Algorithmus unverändert bleiben, spezifische eigens auf die *ReLU*-Knoten zugeschnittene Versionen dieser Regeln zu formulieren. Überlegen wir, was diese leisten müssen, um der *ReLU*-Semantik aus Formel 66 genüge zu tun. Eine `update`-Operation – welche, wie bisher, nur mit Bezug auf Nichtbasisvariablen durchgeführt werden darf – der Wertzuweisung einer Pre-Knotenvariablen $v_{\ell,i}^{\mathrm{Pre}}$, also der ersten Komponente eines Elementes $(v_{\ell,i}^{\mathrm{Pre}}, v_{\ell,i}^{\mathrm{Post}})$ der *ReLU*-Relation R, dient nun dazu eine Verletzung der *ReLU*-Semantik, das heißt den Fall, dass, $\max\{0,\alpha(v_{\ell,i}^{\mathrm{Pre}})\}\neq\alpha(v_{\ell,i}^{\mathrm{Post}})$, zu korrigieren. Falls dabei der aktuelle Zuweisungswert der Post-Knotenvariablen größer oder gleich 0 ist – es soll schließlich, bei unveränderter Zuweisung der Post-Knotenvariable, der Pre-Knotenwert korrigiert werden –, muss die ins Auge zu fassende Wertveränderung δ von $v_{\ell,i}^{\mathrm{Pre}}$ zunächst den alten Zuweisungswert nivellieren, „löschen", $-\alpha(v_{\ell,i}^{\mathrm{Pre}})$, um dann den Wert für $v_{\ell,i}^{\mathrm{Pre}}$ auf den selben, nach Voraussetzung nichtnegativen, Wert zu setzen, wie $v_{\ell,i}^{\mathrm{Post}}$, das heißt, $+\alpha(v_{\ell,i}^{\mathrm{Post}})$ und somit $\delta=\alpha(v_{\ell,i}^{\mathrm{Post}})-\alpha(v_{\ell,i}^{\mathrm{Pre}})$. Damit haben wir folgende $\mathsf{Update}_{\mathsf{pre}}$-Regel.

$$\mathsf{Update}_{\mathsf{pre}}\quad \frac{\begin{array}{c} S{=}(B,N,T,l,u,\alpha,R),\quad (v_{\ell,k}^{\mathrm{Pre}},v_{\ell,k}^{\mathrm{Post}})\in R,\quad N_j\in N,\quad v_{\ell,k}^{\mathrm{Pre}}{=}N_j,\\ v_{\ell,k}^{\mathrm{Post}}\geq 0,\quad \max\{0,\alpha(v_{\ell,k}^{\mathrm{Pre}})\}\neq\alpha(v_{\ell,k}^{\mathrm{Post}}) \end{array}}{\mathtt{update}(\alpha,\mathtt{N_j},\alpha(\mathtt{v}_{\ell,\mathtt{k}}^{\mathrm{Post}})-\alpha(\mathtt{v}_{\ell,\mathtt{k}}^{\mathrm{Pre}}))=\alpha',\quad S'=(B,N,T,l,u,\alpha',R)} \tag{68}$$

Vor dem Hintergrund ähnlicher Erwägungen kann auch der Wert einer Post-Knotenvariablen, $v_{\ell,i}^{\mathrm{Post}}$, der die *ReLU*-Semantik des *ReLU*-Paares $(v_{\ell,i}^{\mathrm{Pre}}, v_{\ell,i}^{\mathrm{Post}})\in R$ verletzt, $\alpha(v_{\ell,i}^{\mathrm{Post}})\neq\max\{0,\alpha(v_{\ell,i}^{\mathrm{Pre}})\}$, korrigiert werden, indem sein alter Wert gelöscht wird, $-\alpha(v_{\ell,i}^{\mathrm{Post}})$, und der Wert, welcher ihm, unabhängig vom Wert $\alpha(v_{\ell,i}^{\mathrm{Pre}})$, gemäß *ReLU*-Semantik zu kommen müsste, setzt, $+\max\{0,\alpha(v_{\ell,i}^{\mathrm{Pre}})\}$, und somit $\delta=\max\{0,\alpha(v_{\ell,i}^{\mathrm{Pre}})\}-\alpha(v_{\ell,i}^{\mathrm{Post}})$ wählt. Die Regel $\mathsf{Update}_{\mathsf{post}}$ fasst diese Erkenntnisse

zusammen.

$$\textsf{Update}_{\textsf{post}}\ \frac{\begin{array}{c} S{=}(B,N,T,l,u,\alpha,R),\quad (v^{\mathrm{Pre}}_{\ell,k},v^{\mathrm{Post}}_{\ell,k})\in R,\quad N_j\in N,\quad v^{\mathrm{Post}}_{\ell,k}{=}N_j,\\ \alpha(v^{\mathrm{Post}}_{\ell,k})\neq \max\{0,\alpha(v^{\mathrm{Pre}}_{\ell,k})\}\end{array}}{\texttt{update}(\alpha,\mathtt{N_j},\max\{0,\alpha(\mathtt{v^{Pre}_{\ell,k}})\}-\alpha(\mathtt{v^{Post}_{\ell,k}}))=\alpha',\ S'=(B,N,T,l,u,\alpha',R)} \tag{69}$$

Es sei an dieser Stelle nachdrücklich darauf hingewiesen, dass die Anwendung der Regeln $\textsf{Update}_{\textsf{pre}}$ und $\textsf{Update}_{\textsf{post}}$ die Wahrheit der Invariante erhält, da insbesondere die Applikation der Funktion `update`, wie im letzten Abschnitt dargelegt, die Wahrheit der Invariante erhält. Wir hatten gesagt, dass ein „Update" nur dann erlaubt ist, wenn die entsprechende Variable, hier konkret eine der beiden *ReLU*-Variablen $v^{\mathrm{Pre}}_{\ell,k}$ oder $v^{\mathrm{Post}}_{\ell,k}$, eine Nichtbasisvariable ist. Es sind jedoch Fälle denkbar, in denen beide *ReLU*-Variablen Basisvariablen sind und eine verletzte *ReLU*-Semantik somit nicht durch eine der beiden Korrekturregeln $\textsf{Update}_{\textsf{pre}}$ oder $\textsf{Update}_{\textsf{post}}$ korrigiert werden könnte. Es bedarf folglich einer Regel, welche in diesen Fällen erlaubt, eine der Variablen des *ReLU*-Paares aus der Basis, gegen eine beliebige Nichtbasisvariable N_j mit von 0 verschiedenem Koeffizienten, herauszutauschen. Eine solche Regel ist PivotForRelu.

$$\textsf{PivotForRelu}\ \frac{\begin{array}{c} S{=}(B,N,T,l,u,\alpha,R),\quad (v^{\mathrm{Pre}}_{\ell,k},v^{\mathrm{Post}}_{\ell,k})\in R,\quad B_i\in B,\quad v^{\mathrm{Pre}}_{\ell,k}{=}B_i\vee v^{\mathrm{Post}}_{\ell,k}{=}B_i,\\ N_j\in N,\quad T_{i,j}{\neq}0\end{array}}{\texttt{pivot}(\mathtt{T},\mathtt{i},\mathtt{j})=(T',B',N'),\quad S'=(B',N',T',l,u,\alpha,R)} \tag{70}$$

Auch diese Regel ist bezüglich der Wahrheit der Invariante unkritisch, da jede beliebige `pivot`-Operation die Wahrheit der Invariante intakt lässt.

Damit sind die wesentlichen Regeln, welcher es für den *Reluplex*-Algorithmus bedarf, zusammengetragen: Mit einer der, von uns nicht diskutierten, Pivotisierungsstrategien, welche die Vollständigkeit des Simplexalgorithmus sichern, versucht der *Reluplex*-Algorithmus, wie bisher, die Regeln $\textsf{Pivot}_1$, $\textsf{Pivot}_2$ und Update und die neuen Regeln $\textsf{Update}_{\textsf{pre}}$ und $\textsf{Update}_{\textsf{post}}$, wobei im Bedarfsfall die Anwendung von PivotForRelu nötig sein kann, so lange anzuwenden, bis eine Konfiguration S, herbeigeführt wurde, die dem Bedingungsteil der Regel ReluSuccess entspricht und die Konfiguration SAT abgeleitet wird. Es gibt hierbei zwei offene Probleme. Wie wird die Unlösbarkeit des Problems festgestellt, welcher die entsprechende Konfiguration UNSAT entspricht? Ist das so beschriebene Verfahren vollständig, oder ist es möglich, dass die Anwendung der Ableitungsregeln zu einem Zirkel immer gleicher Konfigurationen oder einer unendlich langen Ableitungskette immer neuer Konfigurationen führen kann?

Die zweite Frage zuerst beantwortend, ist dies tatsächlich denkbar und *Reluplex* ist so nicht vollständig. Um diese zirkulären oder anderweitig unendlichen Ableitungsketten zu sprengen, gibt es im *Reluplex*-Algorithmus einen „*Countdown*" für jedes *ReLU*-Paar $(v^{\mathrm{Pre}}_{\ell,i}, v^{\mathrm{Post}}_{\ell,i})$. Wird mit Bezug auf dieses Paar eine der Regeln PivotForRelu, $\textsf{Update}_{\textsf{pre}}$ oder $\textsf{Update}_{\textsf{post}}$ – welche, wie gesehen, die Unternehmung beschreiben, die für diesen Knoten verletzte *ReLU*-Semantik wieder herzustellen –

angewendet, so wird der Zähler, sofern noch größer als Null, um eine Einheit verringert. Anschaulich gesprochen: Die Anzahl der Versuche, eine „kaputte“ *ReLU*, zu reparieren, ist endlich. Ist der Zähler schließlich bei 0 angelangt, so wird der entsprechende *ReLU*-Knoten „aufgespalten“, englisch *split*. Dies wird durch die Anwendung der Regel ReluSplit realisiert.

$$\textsf{ReluSplit}\ \frac{S = (B, N, T, l, u, \alpha, R), \quad (v^{\text{Pre}}_{\ell,k}, v^{\text{Post}}_{\ell,k}) \in R, \quad l(v^{\text{Pre}}_{\ell,k}) < 0, \quad u(v^{\text{Pre}}_{\ell,k}) > 0}{\begin{array}{c|c} u'(v^{\text{Pre}}_{\ell,k}) = 0, & l''(v^{\text{Pre}}_{\ell,k}) = 0, \\ u' = ((u \setminus \{u(v^{\text{Pre}}_{\ell,k})\}) \cup \{u'(v^{\text{Pre}}_{\ell,k})\}), & l'' = ((l \setminus \{l(v^{\text{Pre}}_{\ell,k})\}) \cup \{l''(v^{\text{Pre}}_{\ell,k})\}), \\ S' = (B, N, T, l, u', \alpha, R) & S'' = (B, N, T, l'', u, \alpha, R) \end{array}} \tag{71}$$

Die Spaltung, welche aus der Anwendung der Regel ReluSplit resultiert, ist im Konsequenzteil der Regel visuell anhand des vertikalen Striches, der den Ableitungsstrang $S \vdash_{\textsf{ReluSplit}} S'$ vom alternativen Ableitungsstrang $S \vdash_{\textsf{ReluSplit}} S''$ trennt, deutlich erkennbar. Tatsächlich spaltet die Regel ReluSplit den Lösungsprozess in künftig zwei zu untersuchende Fälle, genauer: Konfigurationen. Wir haben es somit fürderhin mit einem *Ableitungsbaum* zu tun, bei welchem jeder Ableitungsteilbaum ein Ableitungsstrang oder wiederum ein Ableitungsbaum ist. In Konfiguration S' wird unterstellt, dass die Obergrenze von $v^{\text{Pre}}_{\ell,k}$ ab sofort den Wert 0 habe oder, anders ausgedrückt, dass der Wert des *ReLU*-Knotens vor der Aktivierungsfunktion nie größer als 0 werden wird, was wiederum bedeutet, dass der Wert des *ReLU*-Knotens nach der Aktivierungsfunktion konstant der Wert 0 sein wird: dieser *ReLU*-Knoten ist stets inaktiv und entspricht, wie in Formel 17 veranschaulicht, drei einfachen Linearen Ungleichungen, mit denen insbesondere die Simplexkomponente von *Reluplex* keine Schwierigkeiten mehr haben wird. In der einen zweiten Teilbaum initiierenden Ableitung S'' wird der andere Fall angenommen. Demnach ist also die Untergrenze von $v^{\text{Pre}}_{\ell,k}$ künftig auf den Wert 0 festgelegt, wonach der Wert des *ReLU*-Knotens vor der Aktivierungsfunktion nie kleiner als 0 werden wird, was wiederum impliziert, dass der Wert des *ReLU*-Knotens nach der Aktivierungsfunktion stets, sofern die *ReLU*-Semantik nicht verletzt wird, denselben Wert wie vor der Anwendung der Aktivierungsfunktion haben wird, dass also dieser *ReLU*-Knoten invariant aktiv ist, was, wie in Formel 18, als Konjunktion von einfachen Linearen Ungleichungen ausgedrückt werden kann, mit welchen *Reluplex*, als erweiterter Simplex-Algorithmus, zurechtkommt.

Wir erkennen hieraus, weshalb die Vollständigkeit des *Reluplex*-Algorithmus stets gegeben ist. Nehmen wir an, der *Reluplex*-Algorithmus wäre unvollständig und es gäbe unendlich lange Ableitungsketten oder im Hinblick auf die Ableitungsbäume: unendlich lange Ableitungspfade. Ausgehend von der Startkonfiguration muss nach einer endlichen Anzahl von Ableitungensschritten eine der Regeln PivotForRelu, $\textsf{Update}_{\textsf{pre}}$ oder $\textsf{Update}_{\textsf{post}}$ verwendet werden, da der Simplexalgorithmus vollständig ist und insbesondere nicht in einem Ableitungszyklus der immer selben Konfigurationen „steckenbleibt“. Der Zähler eines *ReLU*-Knotens ist nun

um eine Einheit verringert und wir sind im Wesentlichen in der Ausgangssituation, welche impliziert, dass wiederum ein Zähler dekrementiert wird und so weiter. Ab irgendeinem Punkt dieser Ableitungskette muss aber der Zähler eines *ReLU*-Knotens, der durch das Knotenpaar $(v_{\ell,i}^{\mathrm{Pre}}, v_{\ell,i}^{\mathrm{Post}}) \in R$ beschrieben ist, abgelaufen sein und eine Spaltung mittels ReluSplit wird eingeleitet. Die beiden aus der Spaltung resultierenden Konfigurationen zeichnen sich nun jeweils dadurch aus, dass in ihnen, der die Aufsplittung verursachende *ReLU*-Knoten in beiden Fällen, wie besprochen, „linearisiert" wurde und, für seinen Teil, nun der Lösung durch den Simplexalgorithmus zugänglich ist. Folglich haben wir zwei Teilprobleme, mit jeweils einem potentiell nicht-linearen *ReLU*-Knoten weniger. Da ein Neuronales Netz nur über endlich viele *ReLU*-Knoten verfügt, ist die Tiefe des binären Ableitungsbaums endlich. Damit ist das Ausgangsproblem in den „Blättern" des Ableitungsbaumes – also den Konfigurationen des Konfigurationsstrangs des Baumes nach der allerletzten durch ReluSplit evozierten Spaltung – vollständig linearisiert und kann allein durch die Verwendung der Regeln Pivot_1, Pivot_2 und Update, also der „Simplexkomponente" von *Reluplex*, dahin überführt werden, dass entweder der Bedingungsteil der Regel ReluSuccess oder der Bedingungsteil der Regel Failure erfüllt ist. Da somit gezeigt ist, dass der Ableitungsbaum, den der *Reluplex*-Algorithmus erzeugt, für jeden Pfad des Baumes nach endlich vielen Ableitungsschritten in einem Blatt endet, deren letzte Konfiguration die Konfiguration SAT oder UNSAT ist, sehen wir uns, mit Hinblick auf unsere anderslautende Eingangsannahme, gezwungen, zu konstatieren, dass der *Reluplex*-Algorithmus vollständig ist.

Man beachte, dass der eben skizzierte „schlimmste Fall" tatsächlich eintreten kann, wonach dann, für ein Netz mit n *ReLU*-Knoten, ein Ableitungsbaum vorliegt, in denen in 2^n Blättern, da alle *ReLU*s linearisiert wurden, jeweils ein System Linearer Ungleichungen vorliegt, welches allein mit den Mitteln des Simplexalgorithmus gelöst wird. Theoretisch liegt damit der *Reluplex*-Algorithmus in der selben, exponentiellen, Komplexitätsklasse, wie die früher skizzierte Verifikationsstrategie mittels des Simplexverfahrens, wonach die Gesamtheit der Lösungen aller 2^n Disjunkte (jeweils zuzüglich der konjunktiven Verknüpfung mit der Eigenschaftsformel P) der Disjunktiven Normalform der Netzwerkformel berechnet werden muss. Praktisch jedoch führt der *Reluplex*-Algorithmus zu kleineren Ableitungsbäumen als den im schlimmsten Fall, und von der Simplexverifikationsstrategie immer, erzeugten Bäumen, was mit einer, je nach Größe des im Zuge des *Reluplex*-Algorithmus jeweils tatsächlich erzeugten Baumes, teils um Größenordnungen, schnelleren Berechnung des Verifikationsergebnisses einhergeht.

Ein durch den *Reluplex*-Algorithmus berechneter Ableitungsbaum, in dem die Ableitungskette jedes Pfades – von der Startkonfiguration bis zur letzten Konfiguration einer Ableitungskette des Blattes – in der Konfiguration UNSAT endet, zeigt, dass das ursprüngliche Problem – konkret also das System Linearer Ungleichungen, welche, bezogen auf das Neuronale Netz N, die Semantik der Pre-Knoten, welche durch die Formel 11 als Konjunktion von Proto-Sätzen beschrieben ist, zuzüglich der aus der Konjunktion Linearer Ungleichungen bestehenden Eigenschaftsbe-

schreibung P einerseits und der in Formel 66 beschriebenen *ReLU*-Beschränkungen andererseits – keine Lösung hat. Der durch den Existenzabschluss der Formel $F_N \wedge P$ gewonnene Satz ist in diesem Fall falsch, was dasselbe bedeutet, wie die Aussage, dass das Neuronale Netz N nicht die Eigenschaft P besitzt. Endet andererseits mindestens ein Pfad des Ableitungsbaums in der Konfiguration SAT, so hat das Problem, etwa die – in der die SAT-Konfiguration unmittelbar ableitenden Konfiguration $S = (B, N, T, l, u, \alpha, R)$ enthaltene – Lösung α, welche zugleich beweist, dass das Netz N die Eigenschaft P hat.

Damit ist zum Abschluss dieses Kapitels gezeigt, dass der *Reluplex*-Algorithmus ein Algorithmus zur Verifikation von Eigenschaften Neuronaler Netze und *Marabou* – welches den *Reluplex*-Algorithmus, nebst weiterer oben angedeuteter aber hier nicht zur weiteren Vertiefung anstehender, Heuristiken, implementiert – ein Programm zur Verifikation von Eigenschaften Neuronaler Netze ist.

8 Probleme der Fließkommaarithmetik

> Es folgte der Bau des Gerätes Z 3 [...]. Die Daten sind folgende: [...] Rechenwerk: rein dual, gleitendes Komma, Parallelrechenwerke für Exponenten und Mantisse, Wortlänge: 22 Bits (Vorzeichen, sieben Exponentenstellen, 14 Mantissenstellen), eingebaute Operationen: $+$, $-$, $\times$, $:$, $\sqrt{\ }$, Multiplikation mit 2, $1/2$, 10, 0.1, -1 [...].
>
> Konrad Zuse,
> *Entwicklungslinien einer Rechengeräte-Entwicklung von der Mechanik zur Elektronik* [Zus60]

> We next discuss [...] techniques that significantly boost the performance of Reluplex: use of [...] floating point arithmetic.
>
> Guy Katz, Clark Barrett, David Dill, Kyle Julian und Mykel Kochenderfer,
> *Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks* [KBD+17]

> [P]remature optimization is the root of all evil.
>
> Donald Ervin Knuth,
> *Structured Programming with go to Statements* [Knu74]

Bis zum jetzigen Zeitpunkt haben wir unterstellt, dass, da die Gewichte und Bias Neuronaler Netze durch Reelle Zahlen beschrieben sind, die Funktionen – im Wesentlichen Additionen und Multiplikationen – aus denen ein Neuronales *ReLU*-Netz konstituiert ist, Funktionen über Reelle Zahlen sind. Auch hatten wir bisher angenommen – und nicht zuletzt den Korrektheitsnachweis des Reluplexalgorithmus des letzten Kapitels darauf gestützt –, dass die Sprache *Marabou*s die Sprache der Linearen Arithmetik *Reeller Zahlen* sei. Beide Annahmen sind falsch. Sowohl Neuronale Netze als auch das Verifikationsprogramm für Eigenschaften Neuronaler Netze, *Marabou*, verwenden *Fließkommazahlen*, auch *Gleitkommazahlen*, englisch *Floating Point Numbers*, genannt, um die Reellen Zahlen zu approximieren. Den Problemen, die sich aus dem Unterschied zwischen der Arithmetik der Reellen Zahlen und der Fließkommaarithmetik ergeben und ihren möglichen Einfluss auf das Ergebnis der Verifikation von Eigenschaften Neuronaler Netze, ist dieses Kapitel gewidmet. Die folgenden Darstellungen der Fließkommazahlen und der Fließkommaarithmetik beziehen sich stets auf den *IEEE754-2019*-Standard, [IEE19] und werden daher im Weiteren nicht einzeln referenziert.

8.1 Fließkommazahlenrepräsentation

Wir beginnen unsere Untersuchung mit den Ganzen Zahlen, da diese als eine besondere Form der Festkommazahlen vorgestellt werden können: Per Konvention

folgen dem Komma – tatsächlich geben wir in dieser Arbeit dem englischen Punkt den Vorzug gegenüber dem Komma, was mit Darstellungsvorteilen, etwa bei der Repräsentation von reellwertigen Tupeln, einhergeht – stets nur Nullen. In der, neben der gebräuchlicheren Zweierkomplement- oder Einerkomplementdarstellung der Ganzen Zahlen, ist in der sogenannten „Darstellung nach Betrag und Vorzeichen“ das höchstwertige Bit – in unseren Darstellungen ist dies stets das Bit ganz links – das sogenannte *Vorzeichenbit*, englisch *Sign Bit*, kurz S, welches, durch den Wert 1 dieses Bits angibt, dass die dargestellte Zahl negativ oder, falls dieses Bit den Wert 0 hat, dass die dargestellte Zahl positiv ist. Die Gesamtheit aller anderen Bits wird *Mantisse* genannt und kurz durch M gekennzeichnet. Abweichend vom Einerkomplementformat, wonach das Einerkomplement einer durch n Bits dargestellten Binärzahl z durch $\sim z$, bei der das Zeichen „$\sim$“, als Kurzform von $(2^n-1)\oplus z$, mit „$\oplus$“ als der bitweisen *xor*-Funktion, für die bitweise Negation steht, beschrieben wird, wählt die „Darstellung nach Betrag und Vorzeichen“ von z das „Einerkomplement“ als $(z\oplus 2^{n-1})$ – diese Operation invertiert, „toggelt“, nur das Vorzeichenbit, lässt alle anderen Bits jedoch unverändert –, was der Repräsentation der Zahl z als Komposition von Vorzeichenwert und Absolutwert, der Mantisse, entspricht. Der Wert des niedrigstwertigen Bits – in unserern Darstellung stets das Bit ganz rechts – steht an der (von rechts nach links betrachtet) nullten Position dieser Binärziffern und wird mit 2^0 gewichtet. Der Wert des nächsthöherwertigen Bits an Position 1 wird mit 2^1 gewichtet und ganz allgemein geht, wenn das Vorzeichenbit an Position $n-1$ steht, der Wert des Bits an Position i, wobei $i \in \{0,\ldots,n-2\}$, mit dem Gewicht 2^i in die Bewertung ein. Der Wert der Zahl ist schließlich die Summe der gewichteten Bitwerte der Bits auf den Positionen 0 bis $n-2$ multipliziert mit -1^s, wobei s der Wert des Vorzeichenbits auf Bitposition $n-1$ ist. Die Position des Kommas befindet sich in unserer Darstellung implizit rechts vom niedrigstwertigen Bit – hinter der 0-ten Position; wir wollen ganz Allgemein anstelle von „hinter“ der i-ten Position auch von „an“ der Position i sprechen, wenngleich damit stets „zwischen Position i und $i-1$“ gemeint ist. In Abbildung 20 ist die Betrag-Vorzeichen-Darstellung für eine 8-Bit-Zahl veranschaulicht.

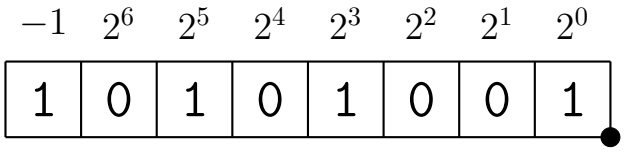


Abbildung 20: Darstellung einer 8-Bit-Zahl gemäß der im Text besprochenen Adaption der Betrag-Vorzeichendarstellung. Die Abgebildete Binärziffernfolge steht für die Zahl $-1^1\cdot(2^5+2^3+2^0)$ also der Zahl -41.00, wobei das Komma, dargestellt als fett gedruckter schwarzer Punkt, stets fix hinter der mit 2^0 gewichteten Stelle, der Position 0, verbleibt.

Eine naheliegende Erweiterung dieser Vorzeichendarstellung stellt die Verschiebung der Kommastelle – des Punkts – von der nullten Stelle an eine beliebige, aber

zunächst feste, Stelle $i \in \{0, \ldots, n-1\}$ der n-Bit langen Vorzeichendarstellung – man beachte, dass für $i = n-1$ die Vorzeichendarstellung nur aus „Nachkommastellen“ besteht. Durch diese *Festkommadarstellung* können nun auch Gebrochene Zahlen zwischen benachbarten Ganzzahlen dargestellt werden. Die Wahl der Position des Kommas stellt einen *Tradeoff* – einer Abwägung der jeweiligen Vor- und Nachteile – dar, zwischen maximaler absoluter Größe der darstellbaren Zahlen und der Anzahl der darstellbaren nichtganzzahligen Gebrochenen Zahlen zwischen den Ganzzahlen. Abbildung 21 stellt die Situation für zwei verschiedene Fixpunkte des Kommas dar.

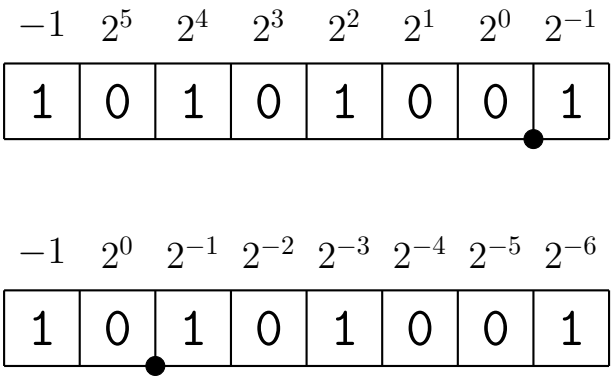


Abbildung 21: Festkommarepräsentation. Oben: Durch die Wahl einer niedrigwertigen Kommaposition können vergleichsweise große und kleine Ganzzahlen, hier: bis $\pm(2^6 - (1/2) = \pm 63.5)$, dargestellt werden, zulasten der Auflösung zwischen den Ganzzahlen; konkret beträgt die Schrittweite hier $2^{-1} = (1/2)$. Die dargestellte Zahl ist $-1 \cdot (16 + 4 + (1/2)) = -20.5$. Unten: Durch die Wahl einer höherwertigen Kommaposition können nur moderat große und kleine Ganzzahlen, hier: bis $\pm(2^1 - (1/64)) = \pm 1.984375$, dargestellt werden, was jedoch mit einer feineren Auflösung zwischen den Ganzzahlen einhergeht; konkret beträgt die Schrittweite hier $2^{-6} = (1/64)$. Die dargestellte Zahl ist $-1 \cdot ((1/2) + (1/8) + (1/64)) = -0.640625$.

So, wie man im Dezimalsystem die Kommastelle um ξ Positionen verschiebt, indem man den Wert mit 10^ξ multipliziert – der Wert von $123.4 \cdot 10^0$ bleibt 123.4 hingegen führt $123.4 \cdot 10^{+1} = 1234$ zu einer Verschiebung der Kommaposition um eine Stelle nach rechts und $123.4 \cdot 10^{-1} = 12.34$ zu einer Verschiebung um eine Position nach links –, so verschiebt man die Kommaposition für allgemeine Basen b durch Multiplikation mit b^ξ und im Speziellen innerhalb des Binärsystems durch Multiplikation mit 2^ξ. Man beachte, dass dies auch für unsere Beispiele gilt: In Abbildung 20 ist die dargestellte Zahl $-41 \cdot 2^0$ und in Abbildung 21 oben, wo die Kommastelle um eine Position nach links verschoben wurde, lautet sie $-41 \cdot 2^{-1} = -(41/2) = -21.5$, in Abbildung 21 unten schließlich, in welcher die Kommastelle um sechs Positionen nach links verschoben wurde, ist die Zahl $-41 \cdot 2^{-6} = -(41/64) = -0.640625$

dargestellt. Wir müssen demnach die Kommastelle gar nicht wirklich verschieben, solange wir uns merken, welchen Wert der *Exponent* ξ hat und die M-Bits – in diesem Fall interpretiert als Ganze Zahl, mit Festkomma an der nullten Position – mit 2^{ξ} multiplizieren.

Diese Überlegungen führen schließlich auf die Idee eines „fließenden Kommas“. Die sich daraus ergebenden Fließkommazahlen kombinieren die Vorteile einer an niedriger Bitposition gewählten mit einer an hoher Bitposition fixierten Festkommastelle: Bei Bedarf können einerseits Zahlen nahe Null mit hoher Präzision dargestellt werden und andererseits sehr große und sehr kleine Zahlen repräsentiert werden. Der dafür zu zahlende „Preis“ besteht in gravierenden Präzisionseinbußen der Resultate arithmetischer Operationen – wie Addition, Subtraktion, Multiplikation, Division –, in denen Zahlen aus beiden Bereichen, also einerseits Zahlen nahe der 0 und andererseits sehr große Absolutwerte von Zahlen, gleichzeitig als Argumente firmieren. Anstatt sich nun, um auf das fließende Komma zurückzukommen, den Wert des Exponenten zu merken – und ihn damit „erstarren“ zu lassen –, geht der Wert des Exponenten in die Binärdarstellung für gebrochene Zahlen, neben dem Vorzeichenbit und den Mantissenbits, mit ein und kann dadurch, ebenso dynamisch wie diese, verschiedene Werte annehmen, wodurch sich der Exponent – und damit die Binärzahlenkommastelle – „verflüssigt“. Die für die Darstellung des Exponenten reservierten Bits werden mit E bezeichnet. Diese Form der Kodierung für Fließkommazahlen ist in Abbildung 22 dargestellt.

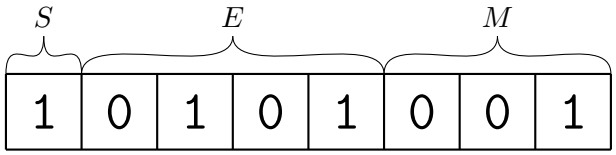


Abbildung 22: Kodierung einer 8-Bit großen Fließkommazahl. Das höchstwertige Bit ist das Vorzeichenbit S. Es folgen die Bits E, welche den Wert des Exponenten beschreiben. Die sich anschließenden letzten Bits sind die Mantissenbits M.

Wenn wir die Exponentenbits E als die Darstellung einer Ganzen Zahl interpretieren, fällt auf, dass wir das Komma eigentlich nur in eine Richtung – nach rechts – verschieben können, da die E-Bits kein eigenes Vorzeichenbit besitzen. Dies ist unvorteilhaft, da damit zwar sehr große aber eben keine gebrochenen Zahlen dargestellt werden können. Eine entsprechende Korrektur der Interpretation der Exponentenbits werden wir im Folgenden vorstellen. Desweiteren hat es sich als nützlich herausgestellt, dem „größten“ durch E darstellbaren Wert – konkret: der Wert in dem alle Bits in E auf 1 gesetzt sind – eine Signalisierungsfunktion zu geben: Sind alle E-Bits auf 1 gesetzt, so ist die dargestellte Zahl keine Zahl im klassischen Sinne. Ganz ähnlich, signalisiert die Binärdarstellung in der alle E-Bits den Wert 0 haben, dass hier keine „normale“ Zahl vorliegt: Entweder haben wir es in diesem Fall mit der Kodierung der positiven oder negativen Null zu tun oder mit den sogenannten

Subnormalen Zahlen, englisch *Subnormal Numbers*, zu tun. Auf beides werden wir später zu sprechen kommen. Für den Moment genügt uns die Feststellung, dass die Anzahl von n_E Bits von E, wobei wir unterstellen, dass $n_E > 1$, zunächst einmal dazu hinreicht, 2^{n_E} Zahlen, abzüglich der Zahlen in der alle Bits auf 1 oder alle Bits auf 0 gesetzt sind, also insgesamt $(2^{n_E} - 2)$ „normale“ Exponentenwerte – konkret die Natürlichen Zahlen $1, \ldots, (2^{n_E} - 2)$ –, darzustellen. Damit der Exponent nun auch negative Werte umfassen kann – das Komma nach links schieben kann –, wird der Wert der nichtnegativen Ganzzahl E durch die Subtraktion eines Biaswertes b dergestalt verschoben, „verzerrt“, dass das Komma in etwa so weit – genauer: um eine Stelle weiter – nach rechts wie nach links geschoben werden kann. Der Wert des Bias ergibt sich demnach als $2^{n_E-1} - 1$, wie folgende Abbildung 23 veranschaulicht.

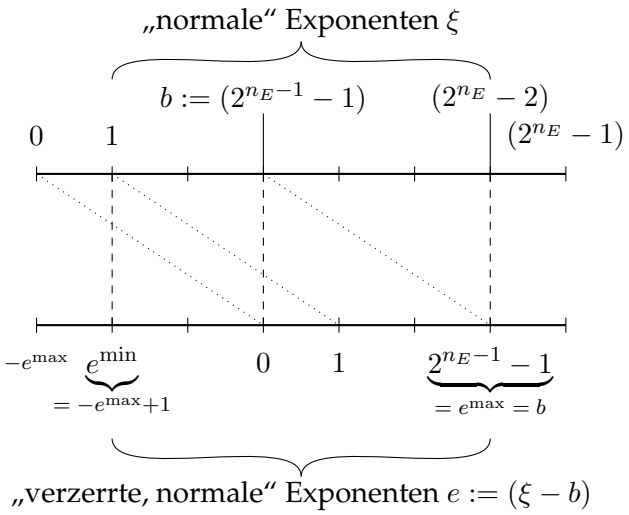


Abbildung 23: Die in den insgesamt 2^{n_E} verschiedenen E-Bits darstellbaren Exponenten ξ werden durch den Wert b in ein Intervall verschoben, in welchem der größte darstellbare normale Exponent $\xi^{\max}$ nur um eine Einheit größer ist als der Absolutbetrag des kleinsten darstellbaren Exponenten $\xi^{\min}$. Die Abbildung zeigt den Fall für ein $n_E = 3$ Bit breites E.

Indem man die n_M-vielen Mantissenbits M nun nicht als eine vorzeichenlose Ganzzahl z_M, bei der das Komma als an der nullten Position von M gedacht wird, auffasst, sondern als reine Nachkommastellen – das heißt hier konkret als die Zahl $z_M \cdot 2^{-n_M}$, also die Zahl, die entsteht, wenn man das Komma an die n_M-te Position von M geschoben denkt –, kann man ein zusätzliches Bit an Information gewinnen, indem man, per Konvention, für die normalen Zahlen den Wert der – wohlgemerkt nicht in der Binärdarstellung erscheinenden – Vorkommastelle implizit als 1 und die Mantisse somit als die Zahl $1.M$, für $M = 001$ also etwa als binär 1.001 (dezimal 1.125) annimmt.

Da das E in Abbildung 22 weder 0000 noch 1111 lautet, handelt es sich bei der dort abgebildeten Zahl `1 0101 001` um eine normale Zahl, die wir nun dekodieren können. Der Exponent wird über $n_E = 4$ Bits repräsentiert und wir bestimmen den Bias entsprechend als $b = (2^{n_E-1} - 1) = 7$ und den durch $E =$ `0101` binär dargestellten Exponenten als dezimal $\xi = 5$, womit sich der verzerrte Exponent ergibt als $(-b + \xi) = e = -2$. Die Mantisse $M =$ `001` präpendieren, „bevorhängen“, wir, wie eben gesehen, mit $1.$, erhalten somit 1.001 und fügen das, durch das auf 1 gesetzte Vorzeichenbit, gefordete unäre Minus davor: -1.001. Nun verschieben wir das Komma um zwei Einheiten nach links, das heißt, wir multiplizieren $-1.001 \cdot 2^e$ und erhalten schließlich den Wert der in Abbildung 22 kodierten Fließkommazahl `1 0101 001` als die dekodierte Binärzahl -0.01001 respektive als die Dezimalzahl $-1 \cdot (2^{-2} + 2^{-5}) = -0.28124$.

Wir haben nun die beiden Sonderfälle zu untersuchen, in denen die Exponentenbits E alle auf 0 beziehungsweise alle auf 1 gesetzt sind. Den ersten Fall unterscheiden wir weiters danach, ob auch alle Mantissenbits M auf 0 gesetzt sind. Ist dies tatsächlich der Fall und auch das Vorzeichenbit ist auf 0 gesetzt – die Ziffernfolge $S \circ E \circ M =$ `0` $\cdots$ `0`$\cdots$`0` $\cdots$ `0`$\cdots$`0` besteht nur aus Nullen – so wird die Zahl $+0$, die *positive Null* kodiert; ist jedoch das Vorzeichenbit als einziges Bit der Binärziffernfolge auf 1 gesetzt, so wird die *negative Null*, -0, dargestellt. In allen anderen Fällen, in denen alle Exponentenbits E alle auf 0 gesetzt sind, werden die sogenannten *Subnormalen Zahlen* kodiert. Mit ihnen hat es folgende Bewandtnis. Wie bei den normalen Zahlen werden die Mantissenbits M als binäre Nachkommastellen interpretiert. Anders als bei den normalen Zahlen, bei denen implizit eine 1 vor dem Komma angenommen wurde, wird bei den Subnormalen Zahlen nun eine 0 vor dem Komma angenommen und der Wert der Mantisse lautet entsprechend $0.M$. Der Wert des verzerrten Exponenten e wird nicht, wie man nach dem für die normalen Zahlen Gesagten annehmen sollte, als $0 - b$, sondern stets als $0 - b\mathbf{+1}$ – das ist, da $b = e^{\max}$, wegen $1 - e^{\max} = e^{\min}$, genau $e^{\min}$. Die Kodierung in Abbildung 24 stellt eine Subnormale Zahl dar.

Der noch zur Untersuchung ausstehende Sonderfall betrifft die Möglichkeit, dass alle Exponentenbits auf 1 gesetzt sind. Wieder unterscheiden wir, in Abhängigkeit davon ob alle Mantissenbits auf 0 gesetzt sind oder mindestens eines dieser Bits auf 1 gesetzt ist, zwei Fälle. Im ersten Fall – alle E-Bits sind auf 1 und alle M-Bits auf 0 gesetzt – hängt der Wert der Kodierung $S \circ E \circ M$ vom Wert des Vorzeichenbits ab. Ist das Vorzeichenbit auf 0 gesetzt, so wird $+\infty$, die *positive Unendlichkeit*, kodiert, ist es hingegen auf 1 gesetzt, so $-\infty$, die *negative Unendlichkeit*. In allen anderen Fällen ist der Sachverhalt, dass alle Exponentenbits auf 1 gesetzt sind ein Signal, englisch *Flag*, dafür, dass diese Kodierung keine Zahl – englisch *Not a Number*, kurz *NaN* – ist. Dieser spezielle *NaN*-Wert hat seine Berechtigung immer dann, wenn sich das Ergebnis einer Fließkommaoperation nicht sinnvoll interpretieren lässt, wie dies beispielsweise bei der Division ∞/∞ der Fall ist. Der *IEEE*-Standard, [IEE19], unterscheidet innerhalb der vorgestellten *NaN*-Kodierungen weitere Abstufungen von Kodierungen des *NaN*-Wertes, die uns hier nicht interessieren. Damit haben wir,

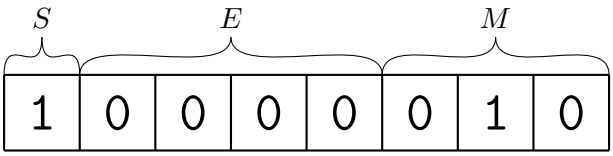


Abbildung 24: Kodierung einer 8-Bit großen subnormalen Fließkommazahl. Die in ihrer Gesamtheit auf 0 gesetzten Exponentenbits E lauten 0000 und, da nicht alle Bits in M auf 0 gesetzt sind, steht die Bitziffernfolge somit für eine Subnormale Zahl. Die sich anschließenden Mantissenbits $M = 001$ stehen für den binären Wert 0.010. Das auf 1 gesetzte höchstwertige Bit, das Vorzeichenbit S, impliziert die Negativität der kodierten Subnormalen Zahl. Wegen $b = 2^{n_E-1} - 1 = 2^3 - 1 = 7$, und da $b = e^{\max}$ ist, ergibt sich $e = e^{\min} = 1 - e^{\max} = -6$. Damit dekodiert sich die abgebildete Ziffernfolge $1\ \ 0000\,010$ zur Binärzahl $-0.010 \cdot 2^{-6} = -0.000000010$ respektive, als Dezimalzahl, $-0.25 \cdot 2^{-6} = -1 \cdot (1/4) \cdot (1/64) = -0.00390625$.

bevor wir uns im Folgenden mit Fließkommaoperationen, der Fließkommazahlenarithmetik oder auch nur Fließkommaarithmetik, auseinandersetzen werden, die Form der Darstellung von Fließkommazahlen, für unsere Zwecke, umfassend dargelegt.

Die Größe der Felder – wobei die Größe des Feldes für das Vorzeichenbit S genau ein Bit umfassen sollte, falls, wie in unsererm Fall, negative Fließkommazahlen dargestellt werden können sollen – für die Exponentenbits E und die Mantissenbits M kann im Prinzip beliebig gewählt werden. Um sowohl Normale als auch Subnormale Zahlen umfassen zu können, sollte E jedoch mindestens zwei Bits breit sein und M natürlich, um überhaupt Information tragen zu können, mindestens ein Bit breit. Die kleinste diesen Bedingungen genügende Fließkommazahlenrepräsentation ist demnach die Darstellung der 4-Bit kleinen Fließkommazahlen, die deswegen zuweilen auch, englisch, *Minifloats* genannt werden. Ihre Werte sind vollumfänglich in Abbildung 25 dargestellt.

Bevor wir uns den mit der Fließkommaarithmetik verbunden Funktionen zuwenden wollen, sei auf folgende begriffliche Konvention hingewiesen. Fließkommazahlen, mit fester Größe für die S-, E- und M-Bitfelder, sind genau die Zahlen, welche sich, nach der oben gegebenen Dekodierungsvorschrift, *exakt* – das ist, ohne dass für die Repräsentation dieses Wertes eine Rundung nötig würde – in diesem Format darstellen lassen. Demnach ist die Zahl -2.0 eine *Minifloat*-Fließkommazahl, nicht jedoch die Zahlen 4 oder $(1/3)$, welche keine Fließkommazahlen im Sinne der *Minifloats* sind. Man beachte, dass einige dieser zuletztgenannten Zahlen, etwa die Zahl 4 durch die Wahl größerer Felder für E oder M zu Fließkommazahlen werden können, andere, wie etwa die Zahl $(1/3)$, hingegen nicht.

S	E	M	Dekodierung	dezimal	Kommentar
0	00	0	0.0	0.0	positive Null
0	00	1	$0.1 \cdot 2^0$	0.5	subnormal
0	01	0	$1.0 \cdot 2^0$	1.0	normal
0	01	1	$1.1 \cdot 2^0$	1.5	normal
0	10	0	$1.0 \cdot 2^1$	2.0	normal
0	10	1	$1.1 \cdot 2^1$	3.0	normal
0	11	0	∞	∞	pos. Unendlichkeit
0	11	1	NaN	NaN	keine Zahl
1	00	0	-0.0	-0.0	negative Null
1	00	1	$-0.1 \cdot 2^0$	-0.5	subnormal
1	01	0	$-1.0 \cdot 2^0$	-1.0	normal
1	01	1	$-1.1 \cdot 2^0$	-1.5	normal
1	10	0	$-1.0 \cdot 2^1$	-2.0	normal
1	10	1	$-1.1 \cdot 2^1$	-3.0	normal
1	11	0	$-\infty$	$-\infty$	neg. Unendlichkeit
1	11	1	NaN	NaN	keine Zahl

Abbildung 25: Dargestellt sind alle möglichen Kodierungen für die *Minifloats* genannten 4-Bit-Fließkommazahlen, welche neben dem Vorzeichenbit S aus zwei Exponentenbits E und einem Mantissenbit M zusammengesetzt sind.

8.2 Fließkommaarithmetik versus Arithmetik der Reellen Zahlen

Am Eingang dieses Kapitels hatten wir konstatiert, dass Fließkommazahlen, als eine Teilmenge der Reellen Zahlen, diese approximieren sollen. Darüber hinausgehend sollen auch die Fließkommaoperationen die gängigen der auf den Reelen Zahlen definierten Funktionen emulieren. Wir haben es dabei, ganz allgemein, mit einer *Algebra* – womit die Gesamtheit der Gesetzmäßigkeiten und Eigenschaften der Funktionen über der Menge dieser Zahlen gemeint ist – der Fließkommazahlen zu tun, welche der Algebra der Reellen Zahlen ähnlich, mit dieser aber nicht identisch, ist. Da wir uns im Speziellen auf die Grundrechenarten Addition, Subtraktion, Multiplikation und Division beschränken werden, sprechen wir stattdessen jedoch von der *Arithmetik* der Fließkommazahlen im Unterschied zur Arithmetik der Reellen Zahlen.

Wenn die Fließkommaoperationen nun die entsprechenden Funktionen über den Reellen Zahlen nachahmen sollen, so ergibt sich nachgerade unmittelbar folgende Schwierigkeit. Das Ergebnis einer solchen Operation über zwei Fließkommazahlen muss nicht notwendigerweise wieder eine Fließkommazahl sein. Formal ausgedrückt sind die Fließkommaoperationen damit nicht *abgeschlossen*. Als Beispiel ist uns im letzten Abschnitt bereits die Zahl $(1/3)$ begegnet, die das Ergebnis der

Division der *Minifloat*-Zahl 1 durch die *Minifloat*-Zahl 3 bezeichnen müsste, aber nicht kann, da $(1/3)$ keine Fließkommazahl ist. Als Ausweg aus dieser aporetischen – „ausweglosen", vom griechischen ἀπορέω [aporéō] wörtlich: „ohne Weg sein"– Situation sieht der Fließkommastandard, [IEE19], verschiedene *Rundungs*-Modi vor. Wir betrachten hier nur den in Fließkommaprogrammen – insbesondere in *Marabou* und allen von uns betrachteten implementierten Neuronalen Netzen – voreingestellten Modus *round-to-nearest*, welcher vorschreibt, dass das exakte Ergebnis einer Fließkommaoperation auf den nächsten, am Naheliegendsten, exakt darstellbaren Fließkommazahlenwert der gegebenen Präzision gerundet wird. Im Falle der Äquidistanz – die zu rundende Zahl liegt genau in der Mitte zwischen zwei exakt repräsentierbaren Fließkommazahlen – ist in der Regel der *ties-to-even*-Modus voreingestellt, der diejenige der beiden in Frage kommenden Fließkommazahlen auswählt, die, ausgehend vom exakten Ergebnis, in Richtung der Geraden Zahl liegt. So ist das Ergebnis der *Minifloat*-Multiplikationsoperation $((1/2) \cdot -(1/2)) = -(1/4)$ gleichweit von den exakt darstellbaren *Minifloats* -0 – es ist hier die negative von der positiven Null zu unterscheiden – und $-(1/2)$ entfernt. Da das Ergebnis $-(1/4)$ zwischen der geraden Zahl -0 und der ungeraden Zahl -1 liegt, wird das Ergebnis dieser Operation, in Richtung der -0, auf den Wert -0 gerundet. Tatsächlich sind in der Programmiersprache *Python*, in welcher die Implementierungen der Software zur vorliegenden Arbeit realisiert sind, die Modi *round-to-nearest* und *ties-to-even* unabänderlich fix voreingestellt. Die Rundung, darauf sei an dieser Stelle nachdrücklich hingewiesesn, erfolgt dabei jedesmal *unmittelbar nach* Ausführung der jeweiligen Operation.

Die Fließkommaarithmetik hat, im Unterschied zur Arithmetik der Reellen Zahlen, desweiteren zu spezifizieren, welche Ergebnisse die jeweiligen Funktionen liefern, wenn mindestens eines der Argumente ein Fließkommawert ist, der für die positive respektive negative Unendlichkeit $\pm\infty$ oder einen der Sonderwerte *NaN* steht. Auch muss die Spezifikation umfassen, wie diese besonderen, in der Menge der Reellen Zahlen nicht existenten, Werte durch die jeweiligen Funktionen als Ergebnis entstehen können. Die diesbezüglichen Details, welche sich samt und sonders im *IEEE*-Standard, [IEE19], dargelegt finden, sind für die vorliegende Arbeit von untergeordneter Bedeutung, sodass wir hier nur den folgenden Fall hervorheben wollen, der zugleich die gravierenden Unterschiede zwischen der Fließkommaarithmetik und der Arithmetik der Reellen Zahlen aufzeigen möge. Wie erwähnt, erfolgt die Rundung des Ergebnisses jeweils stets direkt nach der Ausführung der entsprechenden Operation. Der aus *Minifloats* komponierte Term

$$\frac{1}{(\frac{1}{2} \cdot -\frac{1}{2})} \cdot -\frac{1}{2}$$

wertet demnach zunächst, wie oben gesehen, $((1/2) \cdot -(1/2))$ zu -0 aus. Im Unterschied zur Arithmetik der Reellenzahlen, stellt die anschließende Ausführung der Divisionsoperation, bei der die – negative – Null im Nenner steht und somit für die Reellen Zahlen nicht definiert ist, innerhalb der Fließkommaarithmetik kei-

ne Schwierigkeit dar: Haben Zähler und Nenner, wie im vorliegenden Fall, unterschiedliche Vorzeichen, so ist das Ergebnis die negative (andernfalls die positive) Unendlichkeit, $-\infty$. Die abschließende Multiplikation von $-\infty$ mit dem *Minifloat* $-(1/2)$ liefert wiederum die – diesmal jedoch, aufgrund der Negativität des Faktors $-(1/2)$: positive – Unendlichkeit ∞, sodass der Term $(\,(1/((1/2)\cdot-(1/2)))\cdot-(1/2)\,)$ in der *Minifloat*-Fließkommaarithmetik schließlich zu ∞ auswertet und nicht, wie in der Arithmetik der Reellen Zahlen, zu 2 – und dies, wohlgemerkt, obwohl die 2 eine *Minifloat*-Fließkommazahl ist.

Wir wollen die Probleme, welche daraus resultieren können, dass man die Fließkommazahlen für Reelle Zahlen und die Fließkommaarithmetik für die Arithmetik der Reellen Zahlen nimmt, anhand eines weiteren Beispiels untersuchen. Funktionen, bei denen kleine Änderungen der Eingabe zu großen Änderungen der Ausgabe führen, stellen – in den Fällen, in denen die Eingabe keine Fließkommazahl ist – aufgrund der Notwendigkeit der Rundung, auf die nächste repräsentierbare Fließkommazahl, ein in der Numerik bekanntes Problem für die Fließkommaarithmetik dar. Die *Konditionszahl einer Funktion* gibt – ohne dass wir hier den Raum greifen wollen, um auf die diesbezüglichen Details einzugehen – dementsprechend an, wie sehr sich der Wert der Funktion bei einer kleinen Änderung der Werte der Eingabe der Funktion verändern kann. Je höher der Wert dieser Zahl, desto größer die Änderung. Denselben Sachverhalt aus einem anderen Blickwinkel betrachtend, bedeutet eine große Konditionszahl, dass „Ausgaben" einer Funktion nahe beieinander liegen können, obwohl sie sehr weit auseinander liegende Eingabewerte haben. Da Matrizen $\boldsymbol{M}$, mit einem „Eingabevektor" $\boldsymbol{x}$ multipliziert, einen Vektor $\boldsymbol{y} = \boldsymbol{M} \cdot \boldsymbol{x}$ als „Ausgabevektor" berechnen, kann eine Matrix als eine – lineare – Funktion, die von $\boldsymbol{x}$ auf $\boldsymbol{y}$ abbildet, angesehen werden, der als solcher eine Konditionszahl zukommt.

Eine in dieser Hinsicht besondere Familie von Matrizen stellen die Hilbertmatrizen dar, welche nach dem Mathematiker David Hilbert, welcher, nebst anderen, mit Emmy Noether, Wilhelm Ackermann, Paul Bernays, Gottlob Frege das akademisch Dach teilte, benannt ist, dessen 1928 gestelltes Entscheidungsproblem, durch die gleichwertige (negative) Lösung von Alan Turing und Alonzo Church, wesentlichen Beitrag an der Entstehung der Informatik als Einzelwissenschaft, der Theoretischen Informatik im Speziellen, gehabt haben dürfte. Bei diesen Hilbertmatrizen, kurz $\boldsymbol{H}$, handelt es sich um quadratische $(n \times n)$-Matrizen, die für jede Dimension $n > 0$ – und die entsprechend als $\boldsymbol{H_n}$ bezeichnet werden – wie folgt definiert sind.

Definition 28. *Sei $\boldsymbol{H_n}$ eine $(n \times n)$-Matrix, in deren i-ter Zeile und j-ter Spalte der Wert $h_{ij} := 1/(i + j - 1)$ steht und somit die Gestalt*

$$\boldsymbol{H_n} = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{pmatrix} \tag{72}$$

hat. Eine solche Matrix heißt Hilbertmatrix *der Dimension n.*

Hilbertmatrizen weisen die Besonderheit auf – was hier nur erwähnt, aber nicht gezeigt, werden soll –, dass die Konditionszahl mit steigender Dimensionszahl n immer weiter wächst. Wir erwarten demnach, dass Abbildungen von Fließkommazahleingabevektoren $\boldsymbol{x}$ zu immer größeren Verzerrungen der Ausgabe $\boldsymbol{y}$ führen sollten, je größer die Dimension n wird – oder, dual dazu, dass mit steigendem n, der die nahezu gleichen Werte des Ausgabevektors $\boldsymbol{y_1} \approx \boldsymbol{y_2}$ durch immer weiter von einander abweichenden Werten der jeweiligen Komponenten der Eingabevektoren $\boldsymbol{x_1}$ und $\boldsymbol{x_2}$ induziert sein kann. Zu den weiteren Eigenschaften jeder Hilbertmatrix $\boldsymbol{H_n}$ gehört, dass sie regulär ist. Sie besitzt, was dieselbe Aussage ist, folglich eine Inverse $\boldsymbol{H_n}^{-1}$ für die gilt, dass $\boldsymbol{H_n}\boldsymbol{H_n}^{-1} = \boldsymbol{I_n}$, wobei $\boldsymbol{I_n}$ die $(n \times n)$-Identitätsmatrix ist. Dieser Umstand entspricht, wobei wir die Matrix $\boldsymbol{H_n}$ in ihrer äquivalenten Gestalt als Lineare Abbildung betrachten, der Tatsache, dass die durch $\boldsymbol{H_n}$ beschriebene Funktion bijektiv – das heißt surjektiv und, insbesondere, injektiv – ist. Es kann, aufgrund der Injektivität dieser Funktion, demnach nicht der Fall sein, dass derselbe Ergebnisvektor $\boldsymbol{y}$ durch zwei verschiedene Eingabevektoren $\boldsymbol{x_1}$ und $\boldsymbol{x_2}$ erzeugt wird; das heißt, es gilt $\neg\exists\boldsymbol{x_1}\exists\boldsymbol{x_2}\exists\boldsymbol{y}\colon (\boldsymbol{H_n}\boldsymbol{x_1} = \boldsymbol{y}) \land (\boldsymbol{H_n}\boldsymbol{x_2} = \boldsymbol{y}) \land \neg(\boldsymbol{x_1} = \boldsymbol{x_2})$.

Betrachten wir vor diesem Hintergrund die Matrix

$$\boldsymbol{H_2} = \begin{pmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{pmatrix}.$$

Wir wollen für einen *Minifloat*-Eingabevektor $\boldsymbol{x_1} = (1.0, 1.0)^\top$ die Ausgabe $\boldsymbol{y} = \boldsymbol{H_2}\boldsymbol{x_1}$ berechnen:

$$\boldsymbol{y} = \boldsymbol{H_2}\boldsymbol{x_1} = \begin{pmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{pmatrix} \cdot \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix} = \begin{pmatrix} (1 \cdot 1.0) + (\frac{1}{2} \cdot 1.0) \\ (\frac{1}{2} \cdot 1.0) + (\frac{1}{3} \cdot 1.0) \end{pmatrix} \approx \begin{pmatrix} 1.5 \\ 1.0 \end{pmatrix}$$

Das Ergebnis von $\boldsymbol{y} = \boldsymbol{H_2}\boldsymbol{x_1}$ muss gerundet werden, wofür an der entsprechenden Stelle das Approximationszeichen „$\approx$“ steht. Für die zweite Vektorkomponente $((1/2) \cdot 1.0) + ((1/3) \cdot 1.0)$ ist nämlich der Wert des zweiten Summanden, $((1/3) \cdot 1.0)$, keine Fließkommazahl und wird auf die naheste repräsentierbare *Minifloatzahl* 0.5 gerundet bevor diese mit dem ersten Summanden, welcher zur Fließkommazahl 0.5 auswertet, addiert wird, was schließlich den *Minifloat* 1.0 zum Ergebnis hat. Dies, wohlgemerkt selbst dann, wenn, anders als hier angenommen, die vorher auf *Minifloat*-Fließkommazahlen gerundete Hilbertmatrix

$$\boldsymbol{H_2}^{\mathtt{MF}} := \begin{pmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

verwendet würde.

Probieren wir anhand von $\boldsymbol{H_2}$ eine alternative *Minifloat*-Eingabe, den Vektor $\boldsymbol{x_2} = (+0.0, 3.0)^\top$, wobei $+0.0$ die positive Null bezeichnet:

$$\boldsymbol{y} = \boldsymbol{H_2 x_2} = \begin{pmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{pmatrix} \cdot \begin{pmatrix} +0.0 \\ 3.0 \end{pmatrix} = \begin{pmatrix} (1 \cdot (+0.0)) + (\frac{1}{2} \cdot 3.0) \\ (\frac{1}{2} \cdot (+0.0)) + (\frac{1}{3} \cdot 3.0) \end{pmatrix} = \begin{pmatrix} 1.5 \\ 1.0 \end{pmatrix}$$

Dieses Ergebnis ist exakt, da alle Operationen, insbesondere auch die Multiplikation $(\frac{1}{3} \cdot 3.0)$, welche den Wert der Fließkommazahl 1.0 annimmt, zu *Minifloats* auswerten.

Als Moral aus diesem Beispiel wollen wir zwei Dinge festhalten. Erstens, die Verwendung der Fließkommaarithmetik anstelle der Arithmetik der Reellen Zahlen kann dazu führen, dass die Eigenschaft der Regularität einer Matrix verlustig geht, da obiges Beispiel impliziert, dass für $\boldsymbol{x_1} = (1.0, 1.0)^\top$ und $\boldsymbol{x_2} = (+0.0, 3.0)^\top$ gilt, dass $(\boldsymbol{H_n x_1} = \boldsymbol{y}) \wedge (\boldsymbol{H_n x_2} = \boldsymbol{y}) \wedge \neg(\boldsymbol{x_1} = \boldsymbol{x_2})$, im Widerspruch zur Annahme der Injektivität, und damit Bijektivität, der durch H_2 dargestellten linearen Abbildung. Als zweite Beobachtung halten wir fest, dass, was wir oben vermutet hatten, tatsächlich zutrifft: Nahe beieinander liegende Ausgaben – in der Tat sogar die selben Ausgaben $y = (1.5, 1.0)^\top$ – können durch weit auseinander liegene Eingaben zustande kommen. Die Eingabe $\boldsymbol{x_2} = (+0.0, 3.0)^\top$ ist von der Eingabe $\boldsymbol{x_1} = (1.0, 1.0)^\top$ in dem Sinne maximal entfernt, dass keine als *Minifloat* darstellbare positive Zahl in Richtung der 0 weiter von der 1.0 entfernt sein könnte als $+0.0$ oder (ohne bereits selbst $+\infty$ zu sein) in Richtung $+\infty$ weiter von der 1.0 entfernt sein könnte, als die *Minifloat*-Zahl 3.0. Für die vorher auf Fließkommazahlen gerundete Hilbertmatrix $\boldsymbol{H_2^{\mathtt{MF}}}$ wäre statt $\boldsymbol{x_2} := (+0.0, 3.0)^\top$, der weniger, aber immer noch von $(1.0, 1.0)^\top$ deutlich, abweichende Vektor $\boldsymbol{x_2}' := (0.5, 2.0)^\top$, da $(1/2) \cdot 0.5$, an der entsprechenden Stelle der Matrixmultiplikation, in der Arithmetik der *Minifloats* auf den Wert $((1/4) \approx 0)$, in Richtung der nächsten geraden Ganzen Zahl, rundet, zu wählen.

8.3 Mögliche Konsequenzen für die Korrektheit der Verifikation

Bevor wir dieses Kapitel schließen, wollen wir investigieren, ob die bisher aufgezeigten Probleme, welche durch die Approximation der Arithmetik der Reellen Zahlen durch die Fließkommaarithmetik zuweilen entstehen, ein tatsächliches Problem für *Marabou*, als Verifizierer – welcher, wie wir erwähnt hatten, die Reellen Zahlen durch Fließkommazahlen zu ersetzen gedenkt – darstellen könnten, oder ob wir uns in akademischen Scheindebatten verfangen haben. Um die Argumentation zu erleichtern, unterstellen wir im Folgenden, kontrafaktisch, dass *Marabou* auf der Basis von *Minifloats* operiert. Auch wenn wir auf dieser Grundlage keinen strikten Nachweis erbringen können, dass die in *Marabou* bei der Berechnung des Verifikationsergebnisses durch den *Reluplex*-Algorithmus tatsächlich verwendeten doppelt genauen 64-Bit-Fließkommazahlen keinen gleichwertigen Ersatz für die Reelen Zahlen darstellen können – der Versuch der Falsifikation *Marabou*s soll schließlich auf empirischem Wege, zu welchem Behuf das finale Kapitel 9 vorgesehen ist, und

nicht auf analytische Weise unternommen werden –, so bekommen wir dennoch einen Hinweis auf eine potentielle Fehleranfälligkeit des Verifikationsprozesses, die von den Unterschieden zwischen der Fließkommaarithmetik und der Arithmetik der Reellen Zahlen herrührt.

Die im Zuge des Verifikationsprozesses verwendeten Funktionen, von welchen im *Reluplex*-Algorithmus unentwegt Gebrauch gemacht wird, sind die in Kapitel 7.5 vorgestellten Operationen `pivot` und `update`. Abweichungen der Fließkommaarithmetik von der Arithmetik der Reellen Zahlen mit Bezug auf diese beiden Funktionen haben potentielle Auswirkungen auf die Korrektheit des *Reluplex*-Algorithmus und sind somit von größter Bedeutung. Uns den Zweck der `pivot`-Funktion erneut vergegenwärtigend, erinnern wir, dass dieser darin besteht, einen Basiswechsel durchzuführen. Nehmen wir nun zum Beispiel an, dass *Marabou* zu einem gegebenen System Linearer Ungleichungen bereits das in den Gleichungen 44 allgemein dargestellte System von Linearen Gleichungen angelegt hat, welche in ihrer Gesamtheit die Invariante einhalten, wonach die im Tableau gespeicherten Koeffizienten und die Belegung α für die Basisvariablen in B und die Nichtbasisvariablen in N so beschaffen sind, dass jeder der durch die Gleichungen 44 ausgedrückten Sätze wahr ist. Die i-te dieser Gleichungen könnte nun, mit den wie ebenda angegeben Tableauwerten, vor der Ausführung der `pivot`-Operation, bei der die Basisvariable B_i gegen die Nichtbasisvariable N_2 getauscht werden soll, beispielsweise so aussehen

$$B_i = \overbrace{T_{i,1}}^{:=1} N_1 + \overbrace{T_{i,2}}^{:=3} N_2 \tag{73}$$

beziehungsweise, äquivalent dazu,

$$0 = 1 \cdot N_1 + 3 \cdot N_2 - 1 \cdot B_i.$$

Für die Pivotisierung ist eine Division durch -3 – die sich wohlgemerkt auf die, gegebenenfalls prospektiven, Koeffizienten des Tableaus bezieht – für diese Gleichung erforderlich, was für unsere *Minifloats* die jeweils entsprechend angegebene Rundung erforderlich macht

$$0 = \overbrace{(1/-3)}^{\approx -(1/2)} \cdot N_1 + \overbrace{(3/-3)}^{=-1} \cdot N_2 - \overbrace{(1/-3)}^{\approx -(1/2)} \cdot B_i, \tag{74}$$

bevor mit der Gleichung

$$N_2 = -\frac{1}{2} \cdot N_1 - \frac{1}{2} \cdot B_i \tag{75}$$

der Basiswechsel für Gleichung i abgeschlossen ist. Da die anschließend zu berechnenden Auswirkungen des Basiswechsels auf alle anderen Gleichung keinen Einfluss auf die Gleichung 75 haben, verzichten wir auf ihre Darstellung. Auch von den obligatorischen Umbenennungen $B_i' := N_2$, $N_2' := B_i$ und so weiter sehen wir ab und – ein Basiswechsel, so hatten wir in Kapitel 7.5 bezüglich der Reellen Zahlen gezeigt, ist jederzeit, ohne die, die Wahrheit der Gleichungen 44 garantierende,

Invariante zu verletzen, möglich – tauschen die Variable N_2 zurück gegen B_i, was offenbar eine Division mit $-(1/2)$, das ist, eine Multiplikation mit -2, nötig macht, wodurch wir

$$-2 \cdot N_2 = 1 \cdot N_1 + 1 \cdot B_i$$

und damit

$$B_i = \overbrace{T_{i,1}}^{:=1} N_1 + \overbrace{T_{i,2}}^{:=2} N_2 \tag{76}$$

erhalten. Supponieren wir, dass die Ersetzung der Reellen Zahlen und ihrer Arithmetik durch die Fließkommazahlen und deren Arithmetik keine Auswirkungen auf die Gültigkeit der `pivot`-Funktion bezüglich der Nichtverletzung der Invariante hat. Wir können demnach annehmen, dass, gegeben die initiale Gültigkeit der Invariante, der durch Gleichung 73 ausgedrückte Satz wahr ist und, aufgrund der eben aufgestellten Supposition, nach welcher die `pivot`-Operation auch für die Fließkommaarithmetik die Invariante intakt lässt, dass die Proposition der Gleichung 76 ebenfalls wahr ist. Demnach hätten wir, dass

$$\overbrace{N_1 + 3N_2}^{\text{Gl. 73}} = \overbrace{N_1 + 2N_2}^{\text{Gl. 76}},$$

was offenbar genau dann der Fall ist, wenn

$$N_2 = 0, \tag{77}$$

welches wiederum, ohne Not, eine Belegung von $\alpha(N_2) = 0$ erzwingt. Für alle anderen Belegungen von $\alpha(N_2)$ – man beachte, das wir das Beispiel ohne die Angabe einer Belegung α eingeführt hatten – wird die Invariante verletzt. Wir können also festhalten, dass die Verwendung der Fließkommaarithmetik anstelle der Arithmetik der Reellen Zahlen im *Reluplex*-Algorithmus, durch die Applikation der `pivot`-Operation, für die Korrektheit des Verifikationsverfahrens eine potentielle Gefahr darstellt.

Eine ähnliche Gefährdung evoziert die Verwendung der, in Kapitel 7.5 eingeführten, `update`-Funktion, was wir wie folgt zu untermauern versuchen wollen. Die `update`-Funktion übernimmt, wie erinnerlich, die Aufgabe der Aktualisierung der Zuweisungsfunktion α auf eine Weise, welche die Invariante, wonach alle Gleichungen 44 für die jeweils aktuelle Belegung α wahre Aussagen darstellen, intakt lässt. Eine solche Aktualisierung darf demgemäß jederzeit auf einer beliebigen Nichtbasisvariablen erfolgen – was indes eine „ausgleichende“ Korrektur der Zuweisung bei den Basisvariablen erforderlich macht. Nehmen wir wiederum eine i-te Gleichung aus der Menge der Gleichungen 44, dieses Mal jedoch explizit mit einer, durch `update` zu ändernden, gegebenen Belegung α, wie folgt, an.

$$\overbrace{(\alpha(B_i))}^{=3.0} = \overbrace{T_{i,1}}^{=1.0} \cdot \overbrace{(\alpha(N_1))}^{=1.0} + \overbrace{T_{i,2}}^{=1.0} \cdot \overbrace{(\alpha(N_2))}^{=1.0} + \overbrace{T_{i,3}}^{=1.0} \cdot \overbrace{(\alpha(N_3))}^{=1.0} \tag{78}$$

Wir beobachten, dass die Gleichung 78 die Invariante einhält. Es soll die Zuweisung für $\alpha(N_2)$ um $\delta = 0.5$ Einheiten erhöht werden, was die folgende Situation nach der Applikation der entsprechenden `update`-Operation impliziert,

$$\overbrace{\underbrace{(\alpha'(B_i))}_{=(\underbrace{\alpha(B_i)}_{3.0}+\underbrace{\delta}_{0.5})\approx 3.0}}^{=3.0} = \overbrace{T_{i,1}}^{=1.0} \cdot \overbrace{(\alpha'(N_1))}^{=1.0} + \overbrace{T_{i,2}}^{=1.0} \cdot \overbrace{\underbrace{(\alpha'(N_2))}_{=1.0+\delta}}^{=1.5} + \overbrace{T_{i,3}}^{=1.0} \cdot \overbrace{(\alpha'(N_3))}^{=1.0}, \tag{79}$$

die nun – aufgrund der für *Minifloats* bestehenden Notwendigkeit, das Ergebnis von $(3.0 + 0.5)$ auf die nächste exakt darstellbare Fließkommazahl, 3.0, zu runden – dazu führt, dass die Gleichung 79 den falschen Satz $3 = (1 \cdot 1) + (1 \cdot 1.5) + (1 \cdot 1)$ behauptet, was eine offensichtliche Verletzung der Invariante darstellt, und dadurch die Korrektheit des *Reluplex*-Algorithmus für Fließkommazahlen und deren Arithmetik in Frage stellt.

Die beiden zuletzt diskutierten Beispiele haben gezeigt, dass fundierte Gründe zum Zweifel an der Korrektheit des *Reluplex*-Algorithmus bestehen, sofern dieser auf Basis der Fließkommazahlenarithmetik anstelle der Arithmetik der Reellen Zahlen realisiert wird. Im nächsten Kapitel werden wir versuchen, zu zeigen, dass *Marabou*, welches den *Reluplex*-Algorithmus auf den Fundamenten der Fließkommaarithmetik implementiert hat, zuweilen tatsächlich falsche Verifikationsergebnisse berechnet, was seinen Status als *Verifizierer* konterkariert.

9 Falsifikation Marabous

> τὸ μὲν γὰρ λέγειν τὸ ὂν μὴ εἶναι ἢ τὸ μὴ ὂν εἶναι ψεῦδος, τὸ δὲ τὸ ὂν εἶναι καὶ τὸ μὴ ὂν μὴ εἶναι ἀληθές
> [tò mèn gàr légein tò òn mē eīnai ē tò mē òn eīnai pseūdos, tò dè tò òn eīnai kaì tò mē òn mē eīnai alēthés]
> [Falsch ist, wenn man sagt, das Seiende sei nicht oder das Nichtseiende sei; wahr ist, wenn man sagt, das Seiende sei und das Nichtseiende sei nicht.]
>
> Aristoteles,
> *Metaphysik, [1011, b25]. Übersetzung von Eugen Rolfes* [Ari20]

> Every genuine test of a theory is an attempt to falsify it, or refute it.
>
> Karl Raimund Popper,
> *Conjectures and Refutations: The Growth of Scientific Knowledge* [Pop65]

> I went down, down, down / And the flames went higher
>
> Johnny Cash,
> *Ring Of Fire*

Die bisherigen Kapitel waren, zugespitzt, nichts als ein Präludium zu diesem Kapitel, welches den Versuch der Falsifikation *Marabou*s, als einem Verifizierer von Eigenschaften Neuronaler Netze, unternimmt. Im letzten Kapitel 8 hatten wir elaboriert, dass fließkommazahlenbasierte Systeme bei dem Versuch, die Arithmetik der Reellen Zahlen durch die Fließkommaartithmetik zu emulieren, zuweilen an ihre Grenzen geraten. Besondere Probleme bereiten dabei demnach insbesondere Berechnungen numerisch schwieriger Funktionen. Zu diesen gehören, wovon wir im letzen Kapitel eine Probe gegeben hatten, auch die durch die Hilbertmatrizen dargestellten Funktionen. In diesem Kapitel werden wir eine Methode entwickeln, Hilbertmatrizen so aufzubereiten und zusätzlich eine „numerisch schwierige“ Eigenschaft zu konzipieren, dass wir *Marabou* damit konfrontieren und die daraus resultierenden Ergebnisse quantitativ, vor allem aber: qualitativ, aufbereiten können. Anschließend werden wir untersuchen, ob sich die dabei gefundenen Ergebnisse „in die Praxis“ und auf „echte“ Netze – im Speziellen, auf bilderkennende Klassifizierer – übertragen lassen. Zunächst gilt es jedoch, eine nicht unwesentliche begriffliche Frage zu klären.

9.1 Der Begriff der Falsifikation

Bevor wir, dem Titel der vorliegenden Arbeit entsprechend, in diesem Kapitel den Versuch unternehmen wollen, *Marabou* – als Verifizierer Neuronaler Netze, genauer: als Verifizierer von Eigenschaften derselben – zu falsifizieren, müssen wir klären,

was es mit dieser merkwürdigen Begrifflichkeit auf sich hat. Im Besonderen ist der, im Anschluss zu untersuchende, folgende Einwand schwerwiegend: Ist denn nicht der Versuch der Falsifikation von *Marabou*, eines Programms, also letztlich eines Gegenstandes, *qua* Kategorienfehler – Gegenstände fallen nicht in die Kategorie der Entitäten (im Unterscheid etwa zu den wahrheitswertfähigen Sätzen), deren Wahrheit, im Zuge einer Verifikation, oder Falschheit, durch eine Falsifikation, festgestellt werden könnte, einfach weil Gegenstände so wenig wahr oder falsch sein können, wie Sätze warm oder faltig sind –, schlechterdings unmöglich?

Am Ende der folgenden Reprise, Wiederaufnahme, der in Kapitel 6.3, im Zuge der Spezifikation der zu untersuchenden Eigenschaften, begonnen und in Kapitel 7 in Einzelheiten skizzierten Darlegung der prinzipiellen Funktionsweise *Marabou*s, werden wir eine Erwiderung auf den Einwand geben können. Erinnern wir uns zunächst, ganz allgemein, des Zwecks, den *Marabou* bei der Verifikation von Eigenschaften Neuronaler Netze, zu erfüllen hat: Ziel des Verifikationsprozesses ist es – wahrheitsgemäß –, zu entscheiden, ob der Satz, der durch Formel 36 ausgedrückt wird, wahr ist. Ist er wahr, dann – denn dies ist die Behauptung des Satzes, und ein Satz ist wahr dann und nur dann, wenn der behauptete Sachverhalt (in der Welt) besteht – hat das Netz N die Eigenschaft P und *Marabou*, *qua* Verifizierer, wird dies feststellen und ausgeben. Es hat damit *verifiziert*, dass das Netz N die Eigenschaft P hat. *Marabou* wird in diesem Fall sogar einen Beweis der Wahrheit des Satzes $(\exists v_1 \ldots \exists v_\ell \colon F_N \wedge P)$, das heißt, von Formel 36, liefern. Dafür wird *Marabou* jeder Knotenvariable $v_1, \ldots, v_\ell$ einen Gegenstand $d_1, \ldots, d_\ell$ des Diskursbereichs $\mathbb{D}$ zugeordnen: $[v_1 \leftarrow d_1, \ldots, v_\ell \leftarrow d_\ell]$, eine, wir erinnern uns, Substitutionsvorschrift. Ist der Satz $(F_N \wedge P)[v_1 \leftarrow d_1, \ldots, v_\ell \leftarrow d_\ell]$, in dem jede Variable v_i durch den Gegenstand d_i ersetzt wurde, nun tatsächlich, wie von *Marabou* insinuiert, wahr, so zeigt dies ja gerade, dass es wirklich jeweils ein $v_1, \ldots, v_\ell$ dergestalt *gibt*, dass $(F_N \wedge P)$ wahr ist. Ist der Satz $(\exists v_1 \ldots \exists v_\ell \colon F_N \wedge P)$ hingegen falsch, und hat N somit nicht die Eigenschaft P, wird *Marabou* dies ebenfalls feststellen und zu verstehen geben. Es hat dann den Satz $(\exists v_1 \ldots \exists v_\ell \colon F_N \wedge P)$ *falsifiziert*, kann aber schlechterdings nicht auf die obige Art beweisen, dass der Satz falsch ist, denn dazu müsste es zu *jeder möglichen* Zuordnung von Gegenständen $d_1, \ldots, d_\ell$ des Diskursbereichs an jede der Variablen $v_1 \ldots v_\ell$ zeigen, dass der Satz $(F_N \wedge P)[v_1 \leftarrow d_1, \ldots, v_\ell \leftarrow d_\ell]$ falsch ist, da nur so gezeigt ist, dass es kein $v_1, \ldots, v_\ell$ so gibt, dass $(F_N \wedge P)$ wahr ist – eine auf diese Weise nicht lösbare Aufgabe, vor dem Hintergrund, dass es überabzählbar unendlich viele Gegenstände des Diskursbereichs von *Marabou* gibt. *Marabou* wird also nach der Falsifikation eines Satzes die Falschheit zwar konstatieren aber keinen Beweis seiner Falschheit angeben.

Dass *Marabou* einen Kategorienfehler der Art wie in obigem Einwand angeführt nicht begeht, ist klar: *Marabou*s Angelegenheit sind schließlich Sätze, nämlich, für ein gegebenes Netz N und eine spezifizierte Eigenschaft P, Sätze der Art, wie sie in Formel 36, $(\exists v_1 \ldots \exists v_\ell \colon F_N \wedge P)$, ihren Ausdruck finden. Die Falschheit eines solchen Satzes zu finden, das heißt, seine Falsifikation, oder seine Wahrheit zu zeigen, sprich: seine Verifikation, ist ja gerade der Daseinszweck *Marabou*s.

Bemerkenswerterweise allerdings, erzeugt *Marabou*, implizit, am Ende des Verifikationsprozesses seinerseits ebenfalls Aussagen der Art: „Es ist der Fall, dass $(\exists v_1 \ldots \exists v_\ell \colon F_N \wedge P)$.“, falls *Marabou* durch die Berechnung von SAT als Verifikationsergebnis zu verstehen gibt, verifiziert zu haben, dass N die Eigenschaft P hat, oder „Es ist nicht der Fall, dass $(\exists v_1 \ldots \exists v_\ell \colon F_N \wedge P)$.“, falls *Marabou* glaubt, und durch die Berechnung von UNSAT ausdrückt, dass N nicht die Eigenschaft P hat. Wenn wir im Folgenden von anthropomorphen Sprechweisen der Art „*Marabou* meint, glaubt, behauptet, sagt, gibt zu verstehen, dass ....“, Gebrauch machen, so stets in diesem eben dargelegten, direkten, Sinne oder, zuweilen, in dem stärkeren, indirekten, Sinne, wonach auch die Logischen Implikationen des „Behaupteten“ mitbehauptet werden. Diese, direkten oder indirekten, Aussagen können nun ihrerseits wahr oder falsch sein. Folglich ist es prinzipiell möglich, *Marabou* selbst zu verifizieren oder falsifizieren; und zwar mit der von Aristoteles, im Eingangszitat zu diesem Kapitel, vorgegebenen „Methode“:

Satz 8. *Zu sagen, dass das Seiende (die Tatsache, dass das Neuronale Netz N die Eigenschaft P hat) nicht sei – das heißt,* Marabou *errechnet als Ergebnis der Verifikation für N und P den Wert* UNSAT*, gleichbedeutend mit der Behauptung, dass $(\exists v_1 \ldots \exists v_\ell \colon F_N \wedge P)$ falsch ist, kurz:* Marabou *behauptet „N hat nicht P.“ – oder, dass das Nichtseiende (die Tatsache, dass N die Eigenschaft P nicht hat) sei –* Marabou *berechnet* SAT*: „N hat P.“ –, ist falsch.*

Beweis. Dieser Satz des Aristoteles ist analytisch. (Ein Satz ist analytisch, oder analytisch wahr, wenn die Wahrheit sich aus dem Begriff – beispielsweise: Ein Dreieck hat drei Ecken. – ergibt. Da der Wahrheitsbegriff des Aristoteles mit dem, von uns in der vorliegenden Arbeit geteilten, Wahrheitsbegriff von Tarski – wonach die Behauptung dass P wahr ist genau dann, wenn P der Fall ist – zusammenfällt, ist der Beweis damit vollständig. Für alternative Wahrheitsbegriffe – mit Bezug auf welche auch der Begriff des „Beweises“ selbst neu zu bestimmen wäre – ist dieser Beweis nicht notwendigerweise zwingend.) □

Zu zeigen, dass einer dieser beiden Fälle vorliegt, entspricht so der Falsifikation *Marabou*s.

Eine alternative Riposte, Erwiderung, auf obigen Einwand leitet sich direkt aus dem Konzept der Programmverifikation ab. Demnach ist ein zu prüfendes Programm genau dadurch „verifiziert“ – diese Redefinition des Begriffs „Verifikation“ erlaubt allerdings den Anschlusseinwand des Vorbeigehens dieser Antwort an der eigentlichen Frage, englisch *question begging*: der Kategorienfehler, nach welchem Entität e nicht in die Kategorie k fällt, wird dadurch „behoben“, dass man e *definiert*, als die Entität, die auch unter k fällt –, dass das Programm nachweislich den in der Spezifikation angegebenen Funktionsumfang, „den Zweck“, des Programms erfüllt. Für *Marabou* hatten wir diesen eingangs identifiziert als „Ziel des Verifikationsprozesses ist es – wahrheitsgemäß –, zu entscheiden, ob der Satz, der durch Formel 36 ausgedrückt wird, wahr ist.“ Der Nachweis einer entsprechenden „Entscheidung“ *Marabou*s, die, entgegen dieser „Spezifikation“, nicht „wahrheitsgemäß“ ist, ent-

spricht demnach der Falsifikation von *Marabou*, als Verifikationsprogramm Neuronaler Netze.

Dem Zitat des Aristoteles ist hinzuzufügen:

Satz 9. *Notwendig falsch ist auch, wenn Q („das Seiende") der Fall ist, zu sagen, dass P_1 und, dass ... und, dass P_n – aber Q und P_1 bis P_n implizieren logisch in ihrer Gesamtheit, als Konjunktion, einen Widerspruch. Hierbei stehen Q sowie P_1 bis P_n, mit $n > 0$, für Aussagevariablen.*

Beweis. Dies ist sofort einsichtig, wenn man sich klar macht, dass nur Widersprüchliches einen Widerspruch impliziert. Widersprüchliches ist aber stets logisch äquivalent zur Kontradiktion $\bot$, dem immer falschen Satz. □

Für das Projekt der Falsifikation *Marabou*s bedeutet dies, dass auch der Nachweis eines derartigen Widerspruchs – der aus einer Menge von Aussagen $P_1, \ldots, P_n$ *Marabou*s, in obigem Sinne, und gegebenenfalls Vorannahmen Q, welche anderweitig als wahr bewiesen worden sind, folgt – als Falsifikation *Marabou*s gilt.

9.2 Falsifikation mittels Hilbertmatrizen

Bereits in Kapitel 8 hatten wir in Definition 28 die n-dimensionalen Hilbertmatrizen $\boldsymbol{H_n}$ als kompakte Notation einer Linearen Abbildung vorgestellt, für die, aufgrund ihrer hohen – und mit steigendem n stets größer werdenden – Konditionszahl, die Approximation der Reellen Zahlen durch Fließkommazahlen problematisch ist. Aufgrund kleinster Rundungsfehler bei der Eingabe $\boldsymbol{x}$, können große Fehler bei der Ausgabe $\boldsymbol{y} := \boldsymbol{H_n x}$ entstehen beziehungsweise – was in diesem Abschnitt von größerem Interesse ist – können zwei fast identische Ausgabevektoren $\boldsymbol{H_n x_1} := \boldsymbol{y_1} \approx \boldsymbol{y_2} := \boldsymbol{H_n x_2}$ generiert werden, für die die Eingaben $\boldsymbol{x_1}$ respektive $\boldsymbol{x_2}$ jedoch, im Hinblick auf ihre jeweiligen Vektorkomponenten, sehr weit auseinander liegen. Es ist eine untersuchenswerte Frage, inwieweit *Marabou*, als Fließkommaarithmetik basierter Verifizierer, den mit den Abbildungen durch die Hilbertmatrix verbundenen Schwierigkeiten zurechtkommt. Leider allerdings ist *Marabou* ein Verifizierer für Neuronale Netze und nicht für Matrizen.

Wenn wir die Struktur der, mit den jeweiligen Gewichten aus $\boldsymbol{w} = (w_1, \ldots, w_n)^\top$ multiplizierten, in ein Neuron v eines Neuronalen Netzes einlaufenden Kanten der Neuronen der Vorgängerschicht $\boldsymbol{x} = (x_1, \ldots, x_n)^\top$ von v, wie in Abbildung 3 aus Kapitel 2, betrachten, und anhand dieser Abbildung erinnern, dass aus diesen Gewichten der Wert für v^{pre}, dem Wert des Neurons vor der Aktivierung, berechnet wird als $\sum_{i=1}^{n} w_i x_i + b$, für einen Biaswert b, so fällt, falls dieser Wert b als $b = 0$ angenommen wird, eventuell eine gewisse Ähnlichkeit der Berechnung dieses Wertes und der Berechnung einer Vektorkomponente einer Linearen Abbildung auf. Es ist nämlich das Produkt einer $(1 \times n)$-Matrix $\boldsymbol{W}$ – oder einem Zeilenvektor einer Matrix, dessen Zeilenvektor $\boldsymbol{W}$ ist – mit demselben Vektor $\boldsymbol{x}$ ganz ähnlich, konkret als $\sum_{i=1}^{n} W_i x_i$, bestimmt. Nun muss v nicht der einzige Knoten in seiner Schicht ℓ sein. Nehmen wir an, es gäbe $n - 1$ weitere Neuronen, ebenfalls mit jeweiligem Biaswert

$b = 0$, in dieser Schicht, und Schicht ℓ hätte exakt n Knoten – genausoviele, wie die Vorgängerschicht $\ell - 1$. Mit der in Kapitel 3 in Abbildung 9 getroffenen Benennungsvereinbarung für die Gewichte eines Neuronalen Netzes und der, im selben Kapitel eingeführten, kompakten Notation, wonach im Speziellen

$$\mathcal{W}_{\ell,[1,n],[1,m]} = \begin{pmatrix} w_{\ell,1,1} & \dots & w_{\ell,1,m} \\ \vdots & \ddots & \vdots \\ w_{\ell,n,1} & \dots & w_{\ell,n,m} \end{pmatrix},$$

ist $\boldsymbol{\mathcal{W}_{\ell,[1,n],[1,m]} v_\ell}$, mit $\boldsymbol{v_\ell} := (v_{\ell,1}, \dots, v_{\ell,n})^\top$, da wir ja angenommen hatten, dass alle Biaswerte $\mathcal{W}_{\ell,[1,n],-1} = (b_{\ell,1}, \dots, b_{\ell,n})^\top$ den Wert 0 haben, ein Ausdruck für den Wert der gesamten Schicht ℓ – betrachtet als der Vektor v_ℓ, dessen Komponenten die Werte der einzelnen Knoten der Schicht ℓ, wohlgemerkt vor der Aktivierung, darstellen. Indem wir nun die Schicht $\ell - 1$ als die Eingabeschicht setzen, und somit $\ell := 1$ wählen und keine weitere Schicht in diesem Netz vorsehen, so haben wir – bis auf die offene Frage, welche Aktivierungsfunktionen dieses Netz hat – ein Neuronales Netz geschaffen, welches, wie die Hilbertmatrix $\boldsymbol{H_n}$, die Abbildung eines n-dimensionalen Eingabevektors, hier $\boldsymbol{v_0} := (v_{0,1}, \dots, v_{0,n})^\top$ auf den n-dimensionalen Ausgabevektor $\boldsymbol{v_1} := (v_{1,1}, \dots, v_{1,n})^\top$ über die durch, wegen $m = n$, die $(n \times n)$-Gewichtsmatrix $\mathcal{W}_{\ell,[1,n],[1,n]}$ beschriebene Abbildung, vermittelt.

Um die Implementierung der Funktion der Hilbertmatrix als Neuronales Netz zu komplettieren, kommt die Verwendung der *ReLU*-Aktivierungsfunktion nicht in Frage, da diese in einer nicht zulässigen Weise alle negativwertigen Vektorkomponenten $y_i \in (y_1, \dots, y_n)^\top =: \boldsymbol{y} = \boldsymbol{H_n x}$ auf Null setzen würde. Eine Aktivierungsfunktion, welche, *comme il faut*, alle Präaktivierungswerte $v_{\ell,i}^{\text{pre}}$ des i-ten Neurons der ℓ-ten Schicht unverändert lässt, ist die Identitätsfunktion, mit $I(z) = z$, für alle $z \in \mathbb{R}$ – womit insbesondere $v_{\ell,i}^{\text{pre}} = I(v_{\ell,i}^{\text{pre}}) := v_{\ell,i}^{\text{post}}$ –, welche wir folgerichtig in unserer Netzimplementierung der Hilbertmatrix als Aktivierungsfunktionen wählen. Es bleibt, nach dem oben Gesagten, nun nur noch, die Gewichtsmatrix entsprechend zu definieren als

$$\mathcal{W}_{\ell,[1,n],[1,n]} = \begin{pmatrix} w_{1,1,1} & \dots & w_{1,1,n} \\ \vdots & \ddots & \vdots \\ w_{1,n,1} & \dots & w_{1,n,n} \end{pmatrix} := \begin{pmatrix} \frac{1}{1} & \dots & \frac{1}{n} \\ \vdots & \ddots & \vdots \\ \frac{1}{n} & \dots & \frac{1}{2n-1} \end{pmatrix} = \boldsymbol{H_n}$$

und wir haben die durch eine Hilbertmatrix H_n beschriebene Lineare Abbildung in exakt dieselbe Abbildung als Neuronales Netz transformiert. Wir haben – vor dem Hintergrund der Diskussion in Kapitel 6.2, ob Neuronale Netze Programme oder Funktionen sind – damit, nebenbei, ein weiteres starkes Indiz, dass Neuronale Netze tatsächlich Funktionen sind und, wie dies auch Matrizen nicht sind, keine Programme. Diese Netze nennen wir N_{H_n}, wobei die Funktionalität von N_{H_n}, für jedes $n > 0$, identisch ist mit der Hilbertmatrix $\boldsymbol{H_n}$ der entsprechenden Dimension. Da wir im Folgenden stets tatsächliche Implementierungen dieser Netze in Fließkommazahlen meinen, bezeichnet „N_{H_n}“ künftig stets diese und nicht ihre reellwertigen

Gegenstücke. Abbildung 26 stellt als Beispiel eines solchen Hilbertmatrixnetzwerks das Netz N_{H_3} dar.

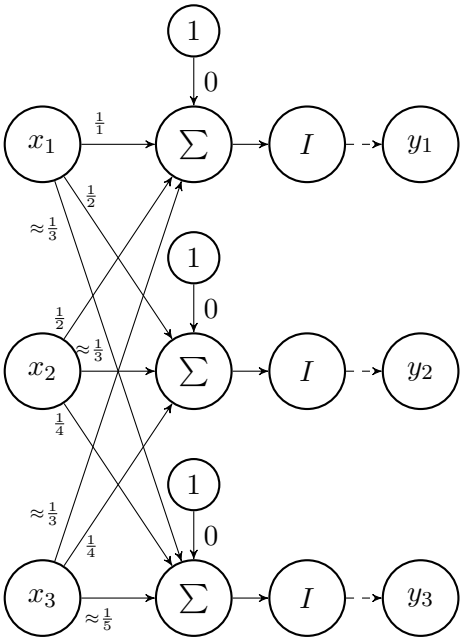


Abbildung 26: Ein aus der Hilbertmatrix $\boldsymbol{H_n}$, für $n = 3$, abgeleitetes Neuronales Netz N_{H_3}, bei dem die Kantengewichte entsprechend gewählt wurden. Dass einige Kantengewichte, da sie keine Fließkommazahlen darstellen, auf die nächste exakt darstellbare Fließkommazahl gerundet werden müssen, ist durch das „≈"-Symbol an entsprechender Stelle angedeutet.

Um *Marabou*, wie im letzten Abschnitt dargelegt, in Widersprüche zu verwickeln, die konkret die Hilbertmatrix involvieren, gehen wir wie folgt vor. Zunächst schaffen wir Fakten – in der Terminologie von Satz 9: Wir etablieren die Wahrheit eines Satzes Q. Da, wie wir in Kapitel 8 untersucht hatten, die Fließkommaarithmetik von der Arithmetik der Reellen Zahlen verschieden und insbesondere nichttrivial ist und die, in der konkreten Implementierung, auf Fließkommazahlen gerundete Gewichtsmatrix des Hilbertmatrixnetzes N_{H_n} eine Fließkommaarithmetik vermittelte Abbildung von Fließkommazahlenvektoren auf Fließkommazahlenvektoren beschreibt, ist die Vorhersage, wie ein konkretes Netz N_{H_n} einen spezifischen fließkommazahlenwertigen Eingabevektor $\boldsymbol{x}$ auf einen fließkommazahlenwertigen Ausgabevektor $\boldsymbol{y}$ abbildet, ebenfalls nichttrivial. Aber, dem englischen Sprichwort folgend, nach welchem „*The proof of the pudding is in the eating.*", der Beweis der Korrektheit einer Abbildung eines Eingabevektors $\boldsymbol{x}$ – wobei wir hier, wie bereits in Kapitel 8.2, konkret die Eingabe stets als $\boldsymbol{x} := \vec{\mathbf{1}}$, den n-dimensionalen Spaltenvektor, der nur aus Einsen komponiert ist, annehmen –, auf einen Ausgabevektor $\hat{\boldsymbol{y}}$ *ist*

dieser Vektor. Die Idee besteht darin, die fließkommazahlenbasierte Implementierung von N_{H_n} mit dem Fließkommavektor $\vec{\mathbf{1}}$ als Argument aufzurufen: Dass

$$N_{H_n}(\vec{\mathbf{1}}) =: \hat{\boldsymbol{y}}_{\vec{\mathbf{1}}} \tag{80}$$

ist, *qua* definierter Identität, eine Tatsache, genauer: sind n verschiedene – und zwar für jedes $n > 1$ eine eigene – Tatsachen. Selbst wenn wir nicht genau sagen können, was die Fließkommazahlenwerte der Elemente dieses Vektors im Einzelnen sind, so können wir eine obere Grenze für jedes Vektorelement $\hat{y}_{\vec{\mathbf{1}}i}$, mit $i \in \{1, \ldots, n\}$, von $\hat{\boldsymbol{y}}_{\vec{\mathbf{1}}} = (\hat{y}_{\vec{\mathbf{1}}1}, \ldots, \hat{y}_{\vec{\mathbf{1}}n})^\top$ wie folgt vornehmen.

$$\hat{y}_{\vec{\mathbf{1}}i} = \overbrace{\underbrace{(1/i)}_{\approx z_{i,1} \leq 1} \cdot 1}^{=z_{i,1}} + \ldots + \overbrace{\underbrace{(1/(i+n))}_{\approx z_{i,n} \leq 1} \cdot 1}^{=z_{i,n}} \leq n \tag{81}$$

Um die i-te Vektorkomponente $\hat{y}_{\vec{\mathbf{1}}i}$ von $\hat{\boldsymbol{y}}_{\vec{\mathbf{1}}}$ zu berechnen, muss, gemäß der Semantik der Matrixmultiplikation, der i-te Zeilenvektor der Gewichtsmatrix, das ist, von $\boldsymbol{H_n}$, als $h_i := ((1/i), \ldots, (1/(i+n)))$, mit dem n-dimensionalen Einsenvektor $\vec{\mathbf{1}}$, ein Spaltenvektor, multipliziert werden: $h_i \cdot \vec{\mathbf{1}}$, was in Formel 81 explizit dargestellt ist. Dabei sind die entsprechenen Gewichtsvektoreinträge, wo nicht als Fließkommazahlen der entsprechenden Präzision repräsentierbar, in der Implementierung auf die nächste darstellbare Fließkommazahl $z_{i,j}$, mit $j \in \{1, \ldots, n\}$, gerundet worden. Dabei kann diese Rundung den Wert 1 nicht überschritten haben, da diese Zahl, wie jeder Eintrag der Hilbertmatrix, vor der Rundung kleiner als oder gleich 1 gewesen ist und damit die 1, welche wohlgemerkt eine Fließkommazahl ist, oder eine kleinere Fließkommazahl die nächste darstellbare Fließkommazahl gewesen sein muss, auf die entsprechend gerundet wird. Die anschließende Multiplikation von $z_{i,j}$ mit 1 verändert diesen Wert offenkundig nicht mehr, weil das exakte Ergebnis dieser Multiplikation, welches wiederum $z_{i,j}$ lautet, wie eben bereits dargelegt, eine Fließkommazahl ist, die keiner Rundung bedarf. Da die Summe in Formel 81 nun genau n Summanden umfasst, kann kein Vektorelement $\hat{y}_{\vec{\mathbf{1}}i}$ von $\hat{\boldsymbol{y}}_{\vec{\mathbf{1}}}$ den Wert von n überschreiten, was den durch Formel 81 ausgedrückten Satz zu einer weiteren Tatsache macht.

Für das Projekt der Falsifikation *Marabou*s mittels Hilbertmatrizen, fragen wir – die Idee der Falsifikationsstrategie zunächst in Umgangssprache darlegend – *Marabou* danach, wie, durch welchen Eingabevektor $\hat{\boldsymbol{x}}_{\boldsymbol{M}} := (\hat{x}_{M_1}, \ldots, \hat{x}_{M_n})^\top$, wobei das „$M$“ uns daran erinnern soll, dass die Antwort, der konkrete Wert dieser Eingabe, von *Marabou* gegeben worden sein wird, der Wert $\hat{\boldsymbol{y}}_{\vec{\mathbf{1}}}$ zustande gekommen sein kann. Da wir ja, aufgrund der in Formel 80 beschriebenen Tatsache, wissen, dass *eine* korrekte Antwort der Einsenvektor $\vec{\mathbf{1}}$ ist, geben wir *Marabou*, gewissermaßen als Hilfestellung, einen Suchraum der Größenordnung r, für diese Elemente so vor, dass dieser Suchraum, in dessen Zentrum stets die uns als eine mögliche Lösung bekannte 1 steht, durch den Satz

$$((1 - 10^r) \leq \hat{x}_{M_i}) \ \wedge \ (\hat{x}_{M_i} \leq (1 + 10^r)), \quad \text{für alle } i \in \{1, \ldots, n\} \tag{82}$$

beschrieben ist – eine Tatsache *per* Definition der Größe des Suchraums, welche unmittelbar impliziert, dass

$$\big(((1-10^r) \leq \hat{x}_{M_i}) \wedge (\hat{x}_{M_i} \leq (1+10^r))\big) \;\Rightarrow\; \big(((1-10^{r+1}) \leq \hat{x}_{M_i}) \wedge (\hat{x}_{M_i} \leq (1+10^{r+1}))\big), \quad \text{für alle } i \in \{1,\ldots,n\}, \tag{83}$$

was, induktiv, die Tatsache ausdrückt, dass Werte die im Radius eines Suchraums liegen, sich auch im Radius eines um eine Größenordnung größeren Suchraumes befinden müssen. Man beachte, dass r die logarithmische Beschreibung dieses Suchraums ist oder, was das gleiche ist, dass eine Inkrementierung von r den Suchraum exponentiell, „um eine Größenordnung“, vergrößert. Formal übergeben wir Marabou für verschiedene n ein Hilbertmatrixnetz N_{H_n} für die Verifikation. Als begleitende, implizit existenzquantifizierte, Eigenschaft P ist das eben umgangssprachlich Formulierte wie folgt zu formalisieren.

$$\begin{aligned} P = {} & ((1-10^r) \leq x_0) \;\wedge\; (x_0 \leq (1+10^r)) \quad \wedge \\ & \quad \vdots \\ & ((1-10^r) \leq x_{n-1}) \;\wedge\; (x_{n-1} \leq (1+10^r)) \quad \wedge \\ & (\hat{y}_{\vec{\mathbf{1}}1} \leq y_0) \;\wedge\; (y_0 \leq \hat{y}_{\vec{\mathbf{1}}1}) \quad \wedge \\ & \quad \vdots \\ & (\hat{y}_{\vec{\mathbf{1}}n} \leq y_{n-1}) \;\wedge\; (y_{n-1} \leq \hat{y}_{\vec{\mathbf{1}}1}) \end{aligned} \tag{84}$$

Man beachte, dass, für gegebene r und n, die Werte $(1-10^r)$, $(1+10^r)$ sowie $\hat{y}_{\vec{\mathbf{1}}i}$, für alle $i \in \{1,\ldots,n\}$, Konstanten sind, deren tatsächliche Werte, als konkrete Zahlen, bei der Formulierung dieser Eigenschaft in der Implementierung dieses Falsifikationsversuchs entsprechend einzusetzen sind. Es sei desweiteren betont, dass die Formulierung der Formeln $(\hat{y}_{\vec{\mathbf{1}}i} \leq y_{i-1}) \;\wedge\; (y_{i-1} \leq \hat{y}_{\vec{\mathbf{1}}i})$, in Ermangelung der Identitätsrelation in dieser Sprache, der *QFLRA*-Ausdruck für $(\hat{y}_{\vec{\mathbf{1}}i} = y_{i-1})$, wobei $i \in \{1,\ldots,n\}$, ist sowie, dass *Marabou* die Indizierung der Ein- und Ausgabevariablen bei 0 und nicht bei 1 beginnt.

Wir sind nun in der vorteilhaften Lage, dass wir, aufgrund der durch Formel 80 beschriebenen Tatsache, *wissen*, dass das Hilbertmatrixnetz N_{H_n} die Eigenschaft P hat, denn der Einsenvektor $\vec{\mathbf{1}}$ ist ja immer ein Beweis für die Wahrheit der entsprechenden, durch Formel 36 ausgedrückten, Existenzbehauptung. Die Frage ist nun, ob auch *Marabou*, durch die Berechnung eines SAT-Ergebnisses bezüglich der gestellten Verifikationsanfrage, eine Lösung findet. Wie in der im Eingang des letzten Abschnitts gegebenen Reprise hervorgehoben, berechnet *Marabou*, in diesem Falle – in dem das SAT-Ergebnis gleichbedeutend damit ist, dass *Marabou* behauptet, dass das Netz N_{H_n} die durch P beschriebene Eigenschaft besitzt – zusätzlich eine Belegung aller Variablen, woraus sich unmittelbar der Vektor $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$ extrahieren lässt.

Aus unseren in Kapitel 8.2 angestellten Vorüberlegungen, sollten wir uns nicht überrascht zeigen, falls *Marabou* einen entsprechenden Eingabevektor $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$ „findet“, der, vektorkomponentenweise, sehr weit von den 1-Werten des Einsenvektors $\vec{\mathbf{1}}$ entfernt liegt. Um diese Entfernung quantifizieren zu können, wählen wir das „Fehler“-Maß

$$\sum_{i\in\{1,\ldots,n\}} \big((1/n)(|1-\hat{x}_{M_i}|)\big)/10^r. \tag{85}$$

Dabei handelt es sich bei dem Ausdruck $(|1-\hat{x}_{M_i}|)$ um den absoluten Abstand, den ein einzelnes Element des von *Marabou* berechneten Vektors $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$ vom entsprechenden Element des Einsenvektors an derselben Stelle, also stets der Zahl 1, hat. Diese Einzelabstände werden in ihrer Gesamtheit aufsummiert und anschließend durch die Anzahl n, die ja auch die Dimension der Hilbertmatrix H_n beschreibt, dieser Elemente geteilt, was somit die *durchschnittliche* Abweichung der Elemente der uns als richtig bekannten Lösung, dem Einsenvektor, beschreibt. Der kleinste Wert dieses Maßes wird bei 0 offenkundig dadurch angenommen, dass $\hat{x}_{M_i} = 1$, für alle $i \in \{1,\ldots,n\}$ ist, was lediglich eine etwas sperrigere Bezeichung für $\vec{\mathbf{1}}$ darstellt. Für die Normierung wird das Maß insgesamt nun noch so skaliert, dass der maximale Fehler auf 1 abgebildet wird. Dieser größte Fehler ergibt sich in Abhängigkeit vom Radius 10^r des Suchraums, wobei die 1 das Zentrum dieses Radius ist, und wird erreicht, genau dann, wenn alle Werte $\hat{x}_{M_i}$ maximal weit, also $|1-(1-10^r)| = |1-(1+10^r)| = 10^r$, von der 1 entfernt liegen. Durch diesen Wert, 10^r, endlich wird abschließend geteilt, und ermöglicht so einen Vergleich der Werte des Maßes für verschiedene durch r bestimmte Radii: Je stärker die Abweichungen, desto näher, als relative, durch Werte des Intervalls $[0,1]$, „prozentual“, ausgedrückte, Angabe der insgesamt möglichen Distanz, liegen die Werte – im Durchschnitt – an den durch den jeweiligen Radius beschriebenen Grenzen, je kleiner die Abweichungen, desto, relativ, näher liegen sie am Zentrum, der ursprünglichen 1.

Den Versuch der Falsifikation Marabous beginnen wir mit einem einfachen Testlauf. Wir übergeben *Marabou* ein Hilbertmatrixnetz, konkret das Netz N_{H_7} sowie die entsprechende, im Protosatz 84 ausgedrückte, Eigenschaft P. Als Suchradius wählen wir den Radius 10, was $r := 1$ entspricht. *Marabou* errechnet für diese Anfrage, das Verifikationsergebnis SAT – „N hat P.“ – und gibt uns, als Beweis dafür, eine Belegung aller Variablen, aus der wir $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$ – die Eingabe, von der *Marabou* meint, dass $N_{H_7}(\hat{\boldsymbol{x}}_{\boldsymbol{M}}) = \hat{\boldsymbol{y}}_{\vec{\mathbf{1}}} = NH_7(\vec{\mathbf{1}})$ gilt – extrahieren können als:

$$\hat{\boldsymbol{x}}_{\boldsymbol{M}} = \begin{pmatrix} \hat{x}_{M_0} \\ \hat{x}_{M_1} \\ \hat{x}_{M_2} \\ \hat{x}_{M_3} \\ \hat{x}_{M_4} \\ \hat{x}_{M_5} \\ \hat{x}_{M_6} \end{pmatrix} = \begin{pmatrix} 0.9966329966301042 \\ 1.1385281385998889 \\ -0.3636363640243048 \\ 6.38720538782069 \\ -8.999999999843743 \\ 9.727272726243081 \\ -1.888888888308861 \end{pmatrix} \tag{86}$$

Wir erkennen, dass, wie erwartet, die Ränder des Suchraums, -9 und $+11$, etwa

durch $\hat{x}_{M_4}$ und $\hat{x}_{M_5}$ durchaus angestrebt werden, wobei aber auch eine Tendenz, in der Nähe des Zentrums, beim Wert 1, zu bleiben, etwa für $\hat{x}_{M_0}$ oder $\hat{x}_{M_1}$, erkennbar ist. Als durchschnittlichen normalisierten Fehler, im Sinne des Fehlermaßes 85, ergibt sich für diesen Vektor $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$ ein Wert von circa 0.4072. Diese bereits starken Abweichungen falsifizieren *Marabou*, *wenn* die Fließkommaversion der Hilbertmatrix regulär ist und die durch sie beschriebene Funktion, als injektive Funktion, somit nur den Einsenvektor $\vec{\mathbf{1}}$ als Eingabe für das Ergebnis $\hat{\boldsymbol{y}}_{\vec{\mathbf{1}}}$ erlaubt. Dass die Gründe zur Annahme der Wahrheit des Antezedenz – des durch „wenn“ eingeleiteten Nebensatzes – schwach sind, davon haben wir uns im Beispiel in Kapitel 8.2 überzeugen können, welches exemplifizierte, dass die Eigenschaft der Regularität der Hilbertmatrix in Fließkommadarstellung und unter Verwendung der Fließkommaarithmetik durchaus verloren gehen kann. Für das soeben von *Marabou* berechnete $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$ könnte tatsächlich gelten, dass es als Eingabe für N_{H_n} denselben Wert liefert wie die Eingabe $\vec{\mathbf{1}}$, kurz, dass $N_{H_7}(\hat{\boldsymbol{x}}_{\boldsymbol{M}}) = \hat{\boldsymbol{y}}_{\vec{\mathbf{1}}} = N_{H_7}(\vec{\mathbf{1}})$. Inwiefern dies tatsächlich der Fall ist, werden wir später untersuchen.

Zunächst wollen wir *Marabou* für weitere Werte von n und r – analog zum eben betrachteten Beispiel für $n = 7$ und $r = 1$ – potentielle Eingaben berechnen lassen, die vermeintlich den Wert $\hat{\boldsymbol{y}}_{\vec{\mathbf{1}}}$, für die jeweilige Dimension n, bei Eingabe in N_{H_n} annehmen. Wir werden die Ergebnisse in Form einer „Wärmekarte“, englisch *Heatmap*, aufbereiten. Wie bei Wärmebildkameras stehen dunkle, im Extremfall: schwarze (r, n)-Koordinaten, je ein Tupel aus dem logarithmischen Wert der Größe des Radius des Suchraums und der Dimension der repräsentierten Hilbertmartrix, für „kalte“ Werte, rote Koordinaten für „wärmere“ Werte, gelbe für „heiße“ und weiße schließlich für „ganz heiße“ Werte. Da wir an der Falsifikation *Marabou*s interessiert sind, stehen uns dunkle, „kalte“ Koordinaten für (r, n)-Kombinationen, für von *Marabou* berechnete Ergebnisse, in denen der Fehler klein ist. Größere Fehler sind „wärmer“. „Heiße“ Fehler, etwa die dunkel- oder hellgelben und insbesondere die weißen (p, n)-Koordinaten, sind so gravierend, dass sie die Falsifikation *Marabou*s implizieren. Betrachten wir nach diesen heranführenden Bemerkungen, in Abbildung 27, nun, unter Verwendung des eben vorgestellten Fehlermaßes, wie die von *Marabou*, für verschiedene Werte von n und r, berechneten Ergebnisse aussehen.

Zu den in der *Heatmap* in Abbildung 27 aufbereiteten Verifikationsergebnissen lassen sich folgende Punkte konstatieren. Unsere Vermutung, wonach mit steigendem n die normalisierten mittleren Fehlerwerte rasch ansteigen findet ihre Bestätigung etwa in der $(r = 1)$-Spalte, ausgehend von der oben beispielhaft vorgestellten $(r = 1, n = 7)$-Kombination, der Abbildung: Für $n = 12$ wird zum erstem Mal ein Fehlerwert größer als 0.6 gemessen, die 0.8 wird bei $n = 29$ überschritten, ein Wert von 0.9 das erste Mal bei $n = 61$ berührt und so weiter. Für das Projekt der Falsifikation *Marabou*s, sind aber weniger diese „warmen“ Stellen der Abbildung – die Hilbertmatrix könnte für Fließkommazahlen ja schließlich auch nichtregulär sein, und dieser Fehlerwert würde wenig aussagen über die Qualitäten von *Marabou* als Verifizierer – bemerkenswert, sondern die „heißen“ Koordinaten.

Für jede (r, n)-Koordinate, die durch ein weißes Pixel beschrieben ist, ist *Marabou*s Verifikationsergebnis UNSAT, wodurch *Marabou* zu verstehen gibt, *behauptet*, dass der Vektor $\hat{\boldsymbol{y}}_{\vec{\mathbf{1}}}$ durch keine Eingabe in N_{H_n} erzeugt werden kann, kurz: „N_{H_n} hat nicht P." Nun *wissen* wir – in dem Sinne von „Wissen" der die Wahrheit des Gewussten impliziert, im Unterschied zur Verwendung etwa im Begriff „Wissensbasis", englisch *Knowledge Base*, in welcher auch „falsches Wissen" enthalten sein kann, wodurch es zu kuriosen Konzepten wie Inkonsistenzen aus der Vereinigung zweier „Wissens"-Basen kommen kann – jedoch, dass N_{H_n} die Eigenschaft P hat und können die Wahrheit des Gewussten durch den Einsenvektor $\vec{\mathbf{1}}$ auch beweisen. Gemäß Satz 8 ist damit die Falschheit des von *Marabou* Gesagten gezeigt und *Marabou* ist somit falsifiziert. In Abbildung 27 erkennen wir, insbesondere in der $(r = 3)$- und $(r = 4)$-Spalte, eine Reihe derartig weißfarbener Koordinaten. Man beachte auch das auffallende Kuriosum, dass für die kleinen, und damit vergleichsweise numerisch unproblematischen, $(n = 2)$-, $(n = 5)$- und $(n = 6)$-Zeilen *Marabou* systematisch UNSAT behauptet, welches wir hier nur notieren, dessen Genese an dieser Stelle aber nicht weiter nachgespürt werden soll.

Doch auch die anderen „heißen" Stellen der *Heatmap* verdienen der Beachtung. Durch die dunkelgelb dargestellten Koordinaten, wobei eine solche sich zum ersten Mal in der $(r = 1)$-Spalte für $n = 62$ zeigt, erkennt *Marabou* tatsachenkonform zwar, durch das Verifikationsergebnis SAT, dass N_{H_n} die Eigenschaft P hat, liefert als Beweis dieser Wahrheit jedoch eine Belegung aller Variablen, die einen Vektor $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$ implizieren, in welchem einige der Vektorkomponenten $\hat{x}_{M_i}$ den Suchraum $(1 - 10^r) \leq \hat{x}_{M_i} \leq (1 + 10^r)$ verlassen haben, was einer Falschaussage entspricht, da eine derartige Belegung im Widerspruch zu der in Formel 82 ausgedrückten Tatsache steht, wodurch *Marabou* wiederum falsifiziert wurde. Einschlägig in diesem Kontext sind auch weiße (r, n)-Koordinaten rechts neben $(r-1, n)$-Koordinaten, die selbst nicht weiß (und nicht hellgelb) sind, etwa die Koordinate $(2, 7)$, deren Nachbarkoordinate, dem Hilbertmatrixnetz N_{H_7} mit Suchradius 10, wir oben beispielhaft ausführlicher untersucht und in Formel 86 die von *Marabou* gefundende Lösung dargestellt hatten. Ein solches weißes Pixel rechts neben einem nicht-weißen und nicht-hellgelben Pixel der *Heatmap* steht, im konkreten Beispielfall, für die Behauptung *Marabou*s, dass die für $(r, n) = (1, 7)$ gefundene und in Formel 86 exemplifizierte Lösung innerhalb des Suchradius 10, keine Lösung für die selbe Matrix im um eine Größenordnung größeren Suchradius 100 sein kann, da, gemäß *Marabou*, für diesen Suchraum *expressis verbis* gar keine Lösung existiert. Diese im Widerspruch zu Tatsache 83 stehende Behauptung falsifiziert *Marabou* ein weiteres Mal. Damit sind fast alle weißen Koordinaten der Spalten $(r = 3)$ und $(r = 4)$ gewissermaßen „doppelt" suspekt: Nicht nur, dass sie weiß sind, sondern auch, dass (sofern) sie irgendwo rechts neben einer rötlich-orangenen Koordinate stehen.

Schließlich steht jede der hellgelben Koordinaten, beispielsweise die Koordinaten $(r, n) = (3, 53)$ oder $(2, 65)$, für eine falsche Aussage *Marabou*s. Zwar behauptet *Marabou*, durch die Berechnung eines SAT-Wertes als Verifikationsergebnis, dass N_{H_n} die Eigenschaft P habe, aber der, aus der Belegung der Variablen *Marabou*s

zu diesem Ergebnis hervorgehende, Vektor $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$ enthält Elemente, die entweder eine unendliche Größe, ausgedrückt durch die Werte $+\infty$ oder $-\infty$, oder Werte die für NaN, und somit für gar keine Zahl stehen. Da ein so beschaffener Vektor als Eingabe in die Netzfunktion N_{H_n} wiederum nur Vektoren erzeugen kann, die mindestens ein Element mit $+\infty$, $-\infty$ oder NaN enthalten, kann dieser Vektor – anders als von *Marabou* durch die Anführung von $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$ als Beweis dafür, dass N_{H_n} die Eigenschaft P habe, behauptet – nicht der Vektor $\hat{\boldsymbol{y}}_{\vec{\mathbf{1}}}$ sein. Dieser Nachweis von derartigen Falschaussagen entspricht somit einer nochmaligen Falsifikation *Marabou*s.

Dass auch in den meisten anderen der, in Abbildung 27 im Farbspektrum von dunkel bis orange dargestellten, Koordinaten ein Problem für *Marabou* verborgen ist, liegt darin begründet, dass in einer zweiwertigen Logik „fast wahr“, gleichbedeutend mit „falsch“ ist. Die von *Marabou*, im Zuge der Berechnung eines SAT-Verifikationsergebnisses, durch die Belegung aller Variablen, erzeugten Vektoren $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$ liefern als Eingabe in das zur Dimension n passende Hilbertmatrixnetz oft „fast“, bis auf kleinere Abweichungen, meist nur um 10^{-7}, das tatsächliche Ergebnis $\hat{\boldsymbol{y}}_{\vec{\mathbf{1}}}$ und $N_{H_n}(\vec{\mathbf{1}}) = \hat{\boldsymbol{y}}_{\vec{\mathbf{1}}} = N_{H_n}(\hat{\boldsymbol{x}}_{\boldsymbol{M}}) =: \hat{\boldsymbol{y}}_{\boldsymbol{M}} = (\hat{y}_{M_1}, \ldots, \hat{y}_{M_n})^\top$ – wobei das M in $\hat{\boldsymbol{y}}_{\boldsymbol{M}}$, als Wert des Hilbertmatrixnetzes bei Eingabe des von *Marabou* berechneten Eingabevektors $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$, uns daran erinnern möge, dass diese Ausgabe des Netzes durch eine von *Marabou* berechnete Eingabe zustande kommt – ist somit „fast wahr“, wie die Abbildung 28 zeigt.

Selbst wenn, wie die Abbildung 28 zeigt, die Fehler *Marabou*s, die als

$$\sum_{i \in \{1,\ldots,n\}} \left((1/n)(|\hat{y}_{\vec{\mathbf{1}}i} - \hat{y}_{M_i}|) \right) \tag{87}$$

berechnet werden, klein sind, so sind sie dennoch – mit Ausnahme der Werte der gänzlich schwarzen (r, n)-Tupel $(0, 1)$ bis $(4, 1)$ sowie $(0, 3)$ bis $(4, 3)$, nicht jedoch der Tupel für $n = 4$, da beispielsweise der Wert des durch die Koordinate $(1, 4)$ beschriebenen Tupels einen Wert von rund $8.33 - 10^{-17}$ annimmt, der in der angegebenen Farbskala nicht von 0 unterscheidbar ist – nicht, wie in der Eigenschaftsspezifikation gefordert, Null und damit als Beweis für die Frage, ob N_{H_n} die Eigenschaft P hat, fehlerhaft, und die Behauptung, dass es sich bei ihnen jeweils um einen entsprechenden Beweis handelt, falsch. Denn wir haben in der Spezifikation der Eigenschaft P nicht formuliert, dass

$$((\hat{y}_{\vec{\mathbf{1}}i} - 10^{-7}) \leq y_{i-1}) \ \wedge \ (y_{i-1} \leq (\hat{y}_{\vec{\mathbf{1}}i} + 10^{-7})). \tag{88}$$

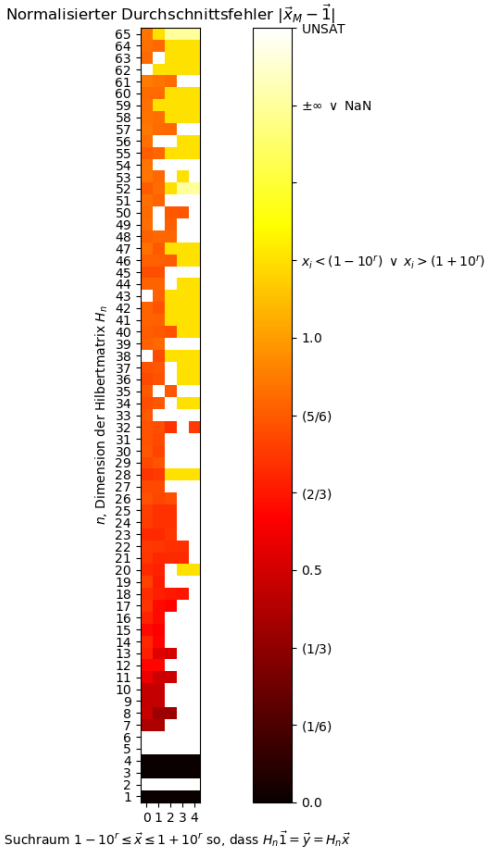


Abbildung 27: Heatmap zur Veranschaulichung des normalisierten Durchschnittsfehlers bezüglich des ursprünglichen Einsenvektors $\vec{\mathbf{1}}$. Zunehmende Fehlerwerte sind durch das Farbspektrum schwarz, Fehler nahe 0.0, über rot, Fehler nahe 0.5, bis orange, Fehler nahe 1.0, dargestellt. Die diskreten Farben, Dunkelgelb, Hellgelb und Weiß, stehen für Inkonsistenzen von Seiten *Marabou*s. Dunkelgelb: Eine der Vektorkomponenten, x_i, des von *Marabou* berechneten Vektors $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$, hier als: $\vec{x}_M$, liegt nicht innerhalb der Grenzen des zulässigen Suchraums. Hellgelb: Eine der Komponenten von $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$ ist keine Zahl, sondern einer der Werte $+\infty$, $-\infty$ oder NaN. Weiß: *Marabou* berechnet UNSAT als Verifikationsergebnis.

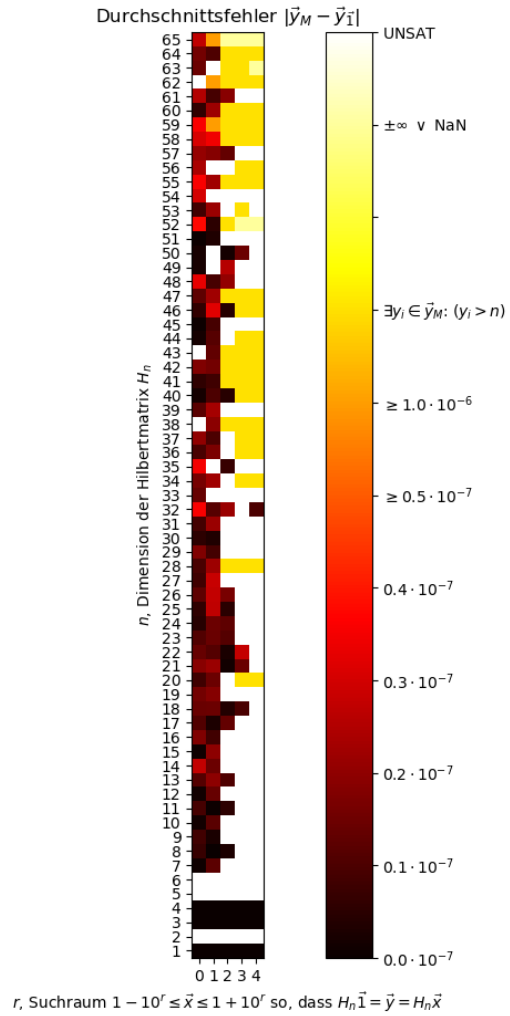


Abbildung 28: Heatmap zur Veranschaulichung des absoluten Durchschnittsfehlers, der für die jeweiligen Werte von r und n, aus der durchschnittlichen Abweichung der jeweiligen Komponononenten von $\hat{\boldsymbol{y}}_{\vec{1}}$ und $\hat{\boldsymbol{y}}_{\boldsymbol{M}}$ entsteht. Die Abbildung zeigt die auf Fehlerwerte zwischen 0 und $0.5 \cdot 10^{-7}$ skalierten Fehlerwerte. Fehlerwerte zwischen $0.5 \cdot 10^{-7}$ und $1.0 \cdot 10^{-6}$ werden auf $0.5 \cdot 10^{-7}$ projiziert. Endlichgroße Fehlerwerte größer oder gleich $1.0 \cdot 10^{-6}$ werden auf den diskreten Wert $1.0 \cdot 10^{-6}$ projiziert. Die diskreten Werte welche durch die Farben Dunkelgelb, Hellgelb und Weiß repräsentiert sind, stehen für Inkonsistenzen von *Marabou*s. Dunkelgelb: Einer der Vektorkomponenten von $\hat{\boldsymbol{y}}_{\boldsymbol{M}}$ überschreitet den Wert n. Hellgelb: Eine der Komponenten von $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$ ist keine Zahl, sondern einer der Werte $+\infty$, $-\infty$ oder NaN. Weiß: *Marabou* berechnet UNSAT als Verifikationsergebnis.

Man beachte, dass selbst für eine auf diese Weise angepasste Eigenschaftsformulierung *Marabou* falsche Ergebnisse berechnet hätte: Die (r, n)-Koordinaten $(1, 59)$, $(1, 62)$ und $(1, 65)$ belegen, durch ihre den Fehler 10^{-6} mindestens erreichenden Werte, dass eine Verletzung selbst dieser „numerischen Sicherheitsgrenzen“ um, wohlgemerkt, mehr als eine binäre Größenordnung, als Verifikationsergebnis *Marabou*s vorkommen kann – nicht zu vergessen: *neben* den Ergebnissen wie $\pm\infty$ oder NaN. In der Heatmap der Abbildung 28 sind diese Fehler *Marabou*s durch orangen gefärbte Pixel gekennzeichnet. Mit Hinblick auf die als dunkelgelb dargestellten Koordinaten aus Abbildung 28 sei angemerkt, dass es Hilbertmatrixnetze N_{H_n}, insbesondere für $r > 1$, so gibt, dass der von *Marabou* berechnete Vektor $\hat{\boldsymbol{x}}_{\boldsymbol{M}}$ einen Vektor $\hat{\boldsymbol{y}}_{\boldsymbol{M}}$ induziert, welcher Vektorelemente enthält, die einen Wert haben, der größer ist als n – im Widerspruch zur Tatsache, die in Formel 81 ausgedrückt ist.

Das in diesem Abschnitt entwickelte, auf Hilbertmatrixnetzen basierende, Verfahren zur Falsifikation eines fließkommabasierten Verifizierers Neuronaler Netze, ist – nach bestem Wissen und Gewissen des Autors der vorliegenden Arbeit – ohne Präzedenz. Seine Applikabilität bezüglich anderer fließkommabasierter Verifizierer als *Marabou* stellt ein Forschungsdesiderat dar.

9.3 Der Jia-Rinard-Exploit fließkommabasierter Verifizierer

Im letzten Abschnitt hatten wir anhand der Verifikation von Eigenschaften von Hilbertmatrixnetzen dargelegt, dass *Marabou*, für numerisch anspruchsvolle Probleme, implizit, entweder direkt falsche Behauptungen aufstellt – indem es behauptet, ein Netz habe eine Eigenschaft nicht, obwohl es sie tatsächlich, und nachweislich, hat – oder Behauptungen tätigt, welche zu Widersprüchen führen, etwa der Umstand, dass die Existenzbehauptung einer Lösung durch die Vergrößerung des Suchraums anschließend verneint wird. Es könnte nun aber der Einwand entstehen, dass die bisherigen Falsifikationen *Marabou*s aus gänzlich akademischen Beispielen – in der Tat bilden die Hilbertmatrizen ja einen, sozusagen künstlich herbeigeführten, numerischen Härtefall – herrühren, welche für „echte“, im Alltag anzutreffende Netze, keinerlei Relevanz besitzen. Wir wollen daher in diesem und im nächsten Abschnitt versuchen, zu zeigen, dass die Verwendung von *Marabou* als Verifizierer von Eigenschaften Neuronaler Netze in der Praxis im Gegenteil sogar ein Sicherheitsrisiko darstellen kann.

Stellen wir uns dazu das folgende Szenario vor, in dem die Neuronalen Netze, mit Bezug auf welche eine Eigenschaft verifiziert werden soll, die in Definition 1 eingeführten Klassifizierer sind; und zwar konkret, die in Kapitel 2.5 vorgestellten Klassifizierer zur Klassifikation von handgeschriebenen *MNIST*-Ziffernbilder. Eine, sagen wir, Institution möchte *Marabou* dazu verwenden, um für diesen Klassifizierer bei Bedarf „Robustheitszertifikate“ für beliebige Bilder zu generieren. Dabei handelt es sich um die in Kapitel 6.3 beschriebene Eigenschaft der Mindestrobustheit, ausgedrückt durch das Tschebyscheff-Distanzmaß aus Formel 37, welches in unserem Fall ausdrückt, wie stark sich der Wert jedes einzelnen Pixels des entsprechenden Bildes

maximal verändern darf – was wir auch die „Perturbation“ oder „Verrauschung“ des Bildes bezüglich dieses Wertes nennen –, sodass keine „unerwünschte“ Klassifikation eintritt. Wir hatten in Kapitel 6.3 zwei Arten dieser „Unerwünschtheiten“ unterschieden, im Hinblick auf welche, die Klassifikation folglich robust sein soll: Zum einen, dass das, durch die Perturbationen der Pixel entstehende, Bild durch den Klassifizierer *überhaupt anders* klassifiziert wird als das ursprüngliche Bild; oder, zum anderen, dass das perturbierte Bild, als einer *bestimmten*, hier und im Folgenden immer: von der Klassfikation des ursprünglichen Bildes verschiendenen, Klasse t angehörig klassifiziert wird. Im Folgenden geht es uns ausschließlich um die Robustheit gegenüber dieser zweiten Art unerwünschten Verhaltens des Klassifizierers, das heißt konkret, um die in Formel 42 spezifizierte Robustheitseigenschaft.

Wir wollen uns, für unser Szenario, vorstellen, dass eine konkrete Gefahr davon ausgeht, wenn ein Bild vom Klassifizierer als Klasse t eingestuft wird, alle anderen Klassifikationen – selbst wenn sie nicht die Klasse des ursprünglichen Bildes sind – jedoch harmlos sind. Da eine zu klein gewählte Mindestrobustheitseigenschaft wenig Aussagekraft besitzt – denn eine Mindestrobustheit von 0, und zwar für jedes Bild, besitzt, wie wir untersucht hatten, schließlich jeder Klassifizierer, der überhaupt mehr als eine Klasse unterscheiden kann –, soll *Marabou* die *größte* Mindestrobustheit, durch das in Kapitel 6.3 beschriebene Verfahren der Binärsuche, ermitteln. Die im Zuge der Binärsuche ermittelte Epsilonuntergrenze ϵ^l gilt als „sicher“, die mindestens $\epsilon_{\text{precision}}$ Tschebyscheffdistanz-Einheiten entfernt liegende Epsilonobergrenze ϵ^u, jedoch nachweislich – denn das Bild $\hat{\boldsymbol{x}}^{\mathbf{adv}}$ ist ein bekanntes Gegenbeispiel für die Mindestrobustheit eines Perturbationsradius von ϵ^u Einheiten – als „unsicher“. Für eine Auffrischung des Begriffs „Perturbationsradius“, sei auf die Abbildung 16 rückverwiesen, in welcher die Kanten des zweidimensionale Hyperwürfels den „Radius“ der, in den jeweiligen Dimensionen erlaubten, Perturbation darstellen. Im Falle der *MNIST*-Ziffernklassifizierer ist der Hyperwürfel $(28 \times 28) = 784$-dimensional und die Veranschaulichung des Radius entsprechend anzupassen. Die implizite Aussage von *Marabou*, dass für ein gegebenes Bild keine Perturbation, im Sinne des Tschebyscheffdistanzmaßes, von höchstens ϵ^l Einheiten pro Pixel, dazu führen kann, dass das Klassifizierer-Netz die gefährliche Klasse t für dieses Bild vorhersagt – denn *Marabou* hat für die entsprechende Eigenschaftsspezifikation, welche durch die Formel 42 ausdrückt, dass es ein Beispiel eines maximal ϵ^l-perturbierten Bildes gibt, das vom Klassifizierer als die Klasse t klassifiziert wird, die Konfiguration UNSAT berechnet –, stellt somit ein *Zertifikat* der Mindestrobustheit von ϵ^l für dieses Bild mit Bezug auf die Klasse t dar. Die Institution unseres Szenarios verlässt sich nun darauf, dass *Marabou*, als Verifizierer, die Zertifikate zurecht ausgestellt hat und innerhalb des als „sicher“ zertifizierten Perturbationsradius ϵ^l um das für den Klassifizierer entsprechend zertifizierte Bild kein Bild existieren kann, welches als t klassifiziert wird. Jede Person, so wollen wir für das Szenario annehmen, welche das Originalbild und das Robustheitszertifikat, welches jenem die Mindestrobustheit ϵ^l gegenüber der „gefährlichen“ Klasse t bescheinigt, vorzeigen kann, darf in unserer Institution dem entsprechenden Klassifizierer jedes

beliebige Bild, das um höchstens ϵ^l Einheiten pro Pixel vom Originalbild abweicht, zur Klassifikation übergeben.

Von einem ähnlichen Szenario gehen auch Kai Jia und Martin Rinard in ihrem Artikel *Exploiting Verified Neural Networks via Floating Point Numerical Error*, [JR20], aus, auf welchen wir uns in diesem Abschnitt mit dem Folgenden beziehen, von dem dort beschriebenen Vorgehen aber in einigen Punkten abweichen werden. Die Beobachtung ist demnach diese. Die Institution unseres Szenarios ist nun prinzipiell verwundbar, falls das von *Marabou* ausgestellt Zertifikat fehlerhaft ist. Wenn es, anders als von *Marabou* zertifiziert, eine Verrauschung des Bildes um höchstens ϵ^l Einheiten gibt, welche dennoch als t klassifiziert wird, kann ein Angreifer, der um die Schwächen *Marabou*s als Verifizierer weiß, versuchen, dieses Bild zu finden, um den Fehler *Marabou*s auszunutzen – im Englischen spricht man daher von einem *Exploit* – und der Institution aufgrund der unterstellten Gefährlichkeit der Klassifikation des Klassifizierers von t für dieses Bild potentiell großen Schaden zufügen. Das Gesagte gilt um so stärker, falls ein Angreifer, durch die geschickte Wahl bestimmter Bilder, das zu attackierende Zertifikat, etwa mit Bezug auf die kritische Zielklassifikation t, selbst bestimmen kann. In diesem Fall spricht man im Englischen vom *Certificate Spoofing*, wonach ein Angreifer durch das Zertifikat eine Sicherheit vortäuschen, „*spoofen*“, kann, während er bereits die Mittel kennt, sie zu unterlaufen. Wir wollen im Folgenden entbergen, dass sich *Marabou* für genau diese Attacke – diesen *Exploit* – anfällig zeigt.

Im, in diesem Kontext in der Literatur meist unreflektiert verwendeten, martialischen Vokabular ausgedrückt, folgt der „Angriff“ auf *Marabou* – was wir verstehen als: Die Falsifikation *Marabou*s wird „in Angriff“ genommen – dem in Abbildung 29 dargelegten Schema. Man beachte, dass sich die dort abgelisteten Bilder ab dem mit „Quasi-Safe“ beschriftetem Bild, pixelweise in der Größenordnung von höchstens 10^{-6} von einander unterscheiden. Die mit bloßem, guten, Auge kaum zu unterscheidenden 256 verschiedenen 8-Bit-Graustufen haben eine Schrittweite von $(1/255)$, was *circa* $4 \cdot 10^{-3}$ entspricht. Wie Jia und Rinard in ihrem, oben erwähnten, Aufsatz herausgefunden haben, zeigen sich fließkommabasierte Verifizierer bezüglich der Verifikation der Robustheitseigenschaft – hier, wie gesagt, stets im Sinne der Robustheit von Formel 42 – von Klassifizierern N anfällig insbesondere solchen Bildern, auf die sich die jeweilige Robustheit im jeweiligen Einzelfall bezieht, gegenüber, für die das Netz N von vornherein wenig robust ist, das heißt, anschaulich, bei welchen es keiner großen Änderung der Werte ihrer Pixel mehr Bedarf, damit das Netz für diese leicht veränderten Bilder seine Vorhersage in Richtung der unerwünschten Klassifikation t abändert. Jia und Rinard führen, am angegebenen Ort, dieses Verhalten des Verifizierers auf eine Abnahme der Präzision, der für dieses Zahlenspektrum exakt als Fließkommazahlen darstellbaren Zahlen, zurück. Für die – insbesondere die Antwort auf die Frage, weshalb der vorzustellende Angriff sehr erfolgreich mit Bezug auf von vornherein unrobust klassifizierte Bilder ist, mit Bezug aber auf als robust klassifizierte Bilder in der Regel scheitert, implizierenden – diesbezüglichen, spärlichen, Details, sei auf den Aufsatz, [JR20], verwiesen.

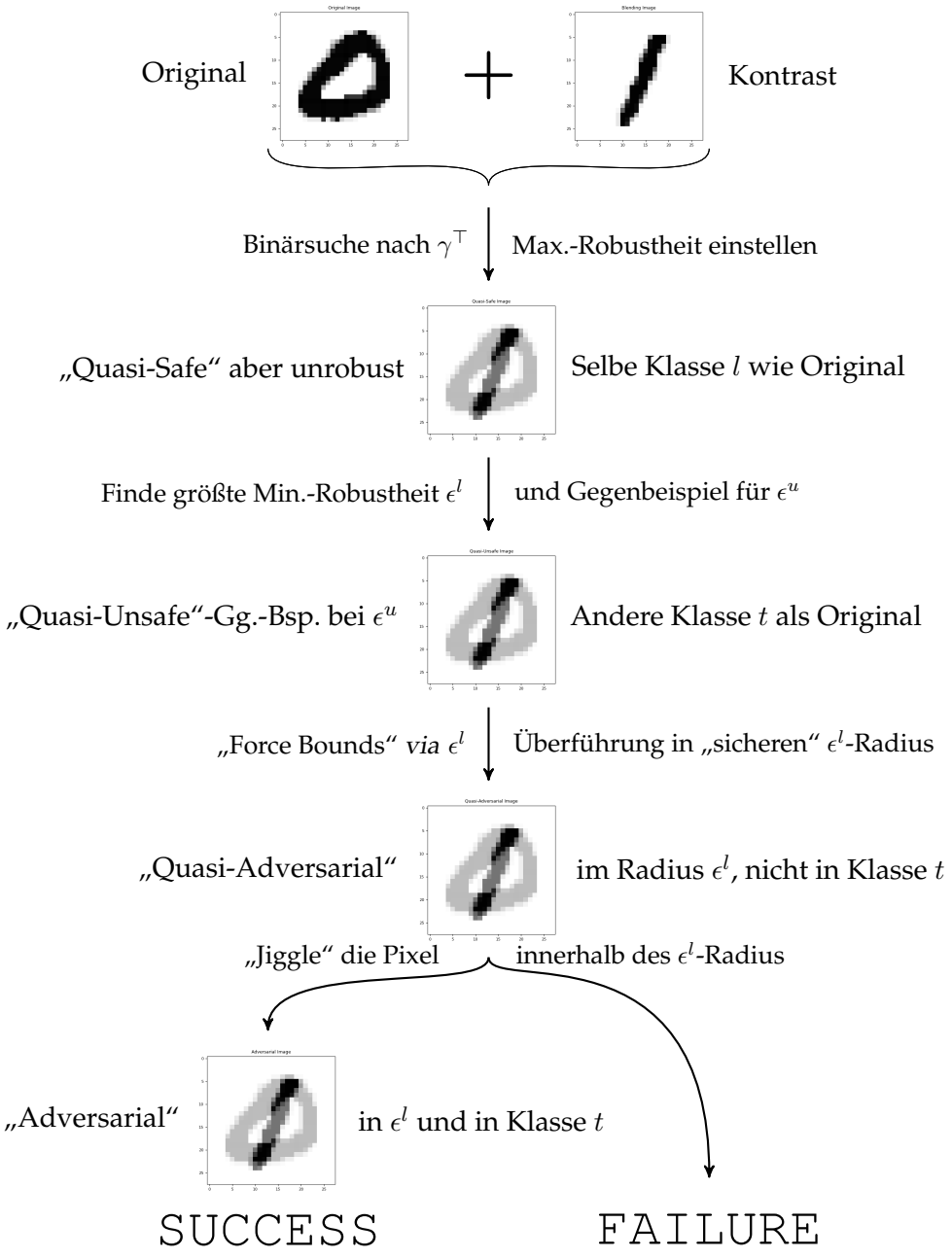


Abbildung 29: Angriffs-*Pipeline* zur im Text beschriebenen Attacke auf *Marabou* als fließkommazahlenbasierten Verifizierer. Adaptation des in Jia und Rinard, [JR20], beschriebenen Verfahrens.

Bevor demnach eine Attacke im Stile von Jia und Rinard gegen *Marabou* gefahren werden kann, bedarf es, im Ausgangspunkt, folglich eines unrobusten, sozusagen „quasi-sicheren“ – denn das Bild soll in dem Sinne sicher sein, dass es, gerade noch so, wie gewünscht als nicht-t, die „gefährliche“ Klasse, klassifiziert wird, aber es ist eben nur so, als, „*qua*“, ob, „*si*“, es sicher wäre: die Vorbereitungen des Angriffs auf dieses laufen ja bereits – Eingabebildes $\hat{\boldsymbol{x}}^{\mathbf{qs}}$. Um sich ein solches zu beschaffen, wurde von den Autoren des Aufsatzes die Methode des Aufhellens, Heller-Machens, eines zunächst robust-klassifizierten Bildes, eingeführt: Man erhöht die Helligkeit eines von einem Netz N als l, dem *Label*, klassifizierten, „originalen“, Bildes $\hat{\boldsymbol{x}}^{\mathbf{o}}$, prozentual um $0 \leq \gamma \leq 1$ Einheiten, solange bis eine winizige weitere Erhöhung der Helligkeit $\gamma^{\top}$ dazu führte, dass das aufgehellte ursprüngliche Bild $\gamma^{\perp} \cdot \hat{\boldsymbol{x}}^{\mathbf{o}}$, wobei $\gamma^{\perp} < \gamma^{\top}$, von N nicht mehr als l klassifiziert würde. Man beachte, dass eine *Verringerung* von γ das Bild heller macht, da 0 der Pixelwert für weiß, 1 aber für schwarz, ist.

Dieser Ansatz ist wie folgt verbesserungsfähig. Es besitzt nämlich auch das „ganz weiße Bild“ – welches entsteht, indem man für $\gamma^{\top} := 0$ das ursprüngliche Bild $\hat{\boldsymbol{x}}^{\mathbf{o}}$ vollständig zu $\gamma^{\top} \cdot \hat{\boldsymbol{x}}^{\mathbf{o}}$ „aufblendet“ – eine Klassifikation durch das infragestehende Netz N. Lautet diese Klassifikation nun, wie für das Ausgangsbild, ebenfalls l, so kann es passieren, dass *für alle* Werte von $0 \leq \gamma^{\top} \leq 1$ das aufgeblendete Bild $\gamma^{\top} \cdot \hat{\boldsymbol{x}}^{\mathbf{o}}$ als l klassifiziert wird, was bereits an dieser Stelle zum Scheitern des Angriffs führt, da in diesem Fall kein unrobustes Startbild erzeugt werden kann. Die gedankliche Weiterführung dieser Überlegung führt indes von der „Überblendung“ eines gänzlichen weißen Bildes über das Originalbild auf die Idee eines – wohlgemerkt von N nicht als l klassifizierten – Kontrastbildes $\hat{\boldsymbol{x}}^{\mathbf{c}}$, englisch *Blending Image*, wörtlich: „mischendes Bild“, und einer „Ineinanderblendung“, Mischung, beider Bilder durch einen γ-Faktor wie folgt.

$$\hat{\boldsymbol{x}}^{\mathbf{qs}} := \gamma \cdot \hat{\boldsymbol{x}}^{\mathbf{o}} + (1 - \gamma) \cdot \hat{\boldsymbol{x}}^{\mathbf{c}}, \quad 0 \leq \gamma \leq 1 \tag{89}$$

Diesen γ-Faktor können wir interpretieren als – wofür das „γ“, als Pendant für das lateinische „g“, steht, γενναῖος [gennaīos], „echt“, „wahr“, englisch *genuine* – den Grad der Originalität, Echtheit, des erzeugten Bildes: Ist der Wert γ gleich 1, so handelt es sich bei der „Mischung“ um das unveränderte Originalbild. Ist γ gleich 0.5, so ist das ermischte Bild eine Chimäre die aus exakt gleichen Anteilen des Originalbildes wie des Kontrastbildes, $\hat{\boldsymbol{x}}^{\mathbf{c}}$, besteht. Ist, schließlich, γ gleich 0, so ist das Ergebnis der „Mischung“ nur das Kontrastbild.

Damit haben wir aber die vorteilhafte Verbesserung erreicht, dass – da laut Voraussetzung, die Klassifikation l des Originalbilds von der Klassifikation, sagen wir, diese sei s, des Kontrastbildes verschieden ist – wir eine Invariante installieren können, wonach $\gamma^{\top}$, mittels Formel 89, immer ein Bild generiert, welches als l klassifiziert wird und $\gamma^{\perp}$ stets ein Bild erzeugt, welches *nicht* als l klassifiziert wird. Initial wählen wir dazu $\gamma^{\top} := 1$, was wie gesehen das Originalbild hervorbringt, welches als l klassifiziert wird, und $\gamma^{\perp} := 0$, was das nicht als l klassifizierte Kontrastbild

entstehen lässt. Den Abstand zwischen $\gamma^\top$ und $\gamma^\perp$ vermag man beschreiben durch

$$\delta^\gamma := \gamma^\top - \gamma^\perp. \tag{90}$$

Dieser Wert δ^γ kann offenbar durch das Verfahren der Binär-„Suche“ beliebig nahe an den Wert 0, ohne diesen jedoch zu erreichen, herangeführt werden – wohlgemerkt, bei gleichzeitiger Nichtverletzung der Invariante –, indem $\gamma^\top$ auf den mittleren Wert $\gamma^m := (1/2) \cdot (\gamma^\top - \gamma^\perp)$ gesetzt wird, wenn das aus diesem Wert ermischte Bild $\gamma^m \cdot \hat{\boldsymbol{x}}^{\mathbf{o}} + (1 - \gamma^m) \cdot \hat{\boldsymbol{x}}^{\mathbf{c}}$ vom untersuchten Netz N als l klassifiziert wird, andernfalls aber $\gamma^\perp$ auf diesen Wert γ^m gesetzt wird.

Am Ende dieser Binärsuche – welches durch das Erreichen oder Unterschreiten eines vorher konkret spezifizierten Wertes für δ^γ erreicht wird – ist der höchstens um δ^γ von $\gamma^\perp$ entfernt liegende Wert $\gamma^\top$ von nicht unerheblichem Interesse, da es der Faktor ist, der, aus dem Originalbild und dem Kontrastbild, das „quasi-sichere“ Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ generiert. Der Wert für δ^γ ist dabei bemerkenswerterweise eine obere Abschätzung für die Robustheit dieses Bildes, wie folgende Überlegung zeigen möge. Für den Extremfall, schließlich suchen wir eine „obere“ Abschätzung, nehmen wir an, dass das l-klassifizierte Originalbild das nur aus einswertigen Pixeln bestehende gänzlich schwarze und das nicht-l-klassifizierte Kontrastbild das gänzlich weiße, nur aus nullwertigen Pixeln komponierte, Bild sei. Da die oben formulierte Invariante zu jedem Zeitpunkt der Binärsuche eingehalten wird, ist durch den Abstand δ^γ zu jedem Zeitpunkt *eine* Möglichkeit beschrieben, vom als l klassifizierten quasisicheren Bild $\gamma^\top \cdot \hat{\boldsymbol{x}}^{\mathbf{o}} + (1 - \gamma^\top) \cdot \hat{\boldsymbol{x}}^{\mathbf{c}}$ zu einem als nicht-l klassifizierten Bild, und zwar konkret $\gamma^\perp \cdot \hat{\boldsymbol{x}}^{\mathbf{o}} + (1 - \gamma^\perp) \cdot \hat{\boldsymbol{x}}^{\mathbf{c}}$ zu gelangen: Man vermindere, wegen $\gamma^\perp = \gamma^\top - \delta^\gamma$, alle Pixelwerte des quasisicheren Bildes $\gamma^\top \cdot \hat{\boldsymbol{x}}^{\mathbf{o}} + (1 - \gamma^\top) \cdot \hat{\boldsymbol{x}}^{\mathbf{c}}$ gleichzeitig um δ^γ Einheiten. Folglich kann die Robustheit von $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ nicht größer sein als δ^γ. Damit ist dieser δ^γ-Wert, da man ihn ja für die Binärsuche prinzipiell beliebig zwischen, jeweils exklusive, 1 und 0 wählen kann, ein Parameter für die maximale Robustheit des Ausgangsbild des Angriffs. Je kleiner dieser Wert ist, desto numerisch schwieriger dürfte, nach dem oben Gesagten, die Verifikation werden.

Damit disponieren wir nun, mit dem quasisicheren Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$, über ein als l und damit insbesondere als nicht-t klassifiziertes Ausgangsbild, welches, wie eben dargelegt, über eine Robusthheit von höchstens δ^γ verfügen kann. Betrachten wir für das Netz N, mit Bezug auf welches wir den Angriff durchführen, den Klassifikationsvektor für dieses Bild:

$$N(\hat{\boldsymbol{x}}^{\mathbf{qs}}) = (\hat{y}_0, \ldots, \hat{y}_l, \ldots, \hat{y}_t, \ldots, \hat{y}_9)^\top =: \hat{\boldsymbol{y}}^{\mathbf{qs}}, \tag{91}$$

wobei für $l \neq t$ insbesondere $l, t \in \{0, \ldots, 9\}$ sind. Für diesen, genauer: für jeden Vektor $\boldsymbol{y} := (y_0, \ldots, y_{n-1})^\top$, Klassifikationsvektor, lässt sich, in Bezug auf den Index eines seiner Vektorkomponenten, l, das – wesbezüglich man [JR20] vergleiche – Carlini-Wagner Abstandsmaß L_{CW} wie folgt definieren.

$$L_{\mathrm{CW}}(\boldsymbol{y}, l) := y_l - \max_{i \in \{0,\ldots,n-1\} \setminus \{l\}} (y_i) \tag{92}$$

Ist, demnach, y_l selbst das größtwertige Element des Vektors $\boldsymbol{y}$ – was im Fall von Ausgabevektoren von Klassifizierern insbesondere bedeutet, dass die Eingabe als l klassifiziert wird –, dann ist der Wert des Maßes positiv und gibt den Abstand zu einem – es konnte derer mehrere gleichen Wertes geben – zweitgrößten Element des Vektors an. Ist y_l andererseits nicht der größte Wert im Vektor $\boldsymbol{y}$, so ist der entsprechende Wert für L_{CW} negativ und sein Absolutwert gibt den Abstand von y_l zu einem größten Element des Vektors an.

Bisher hatten wir unterstellt, dass t, der für die Instutution unserers Szenarios „gefährliche" Klassifikationswert, gezielt angegriffen werden soll: Die Institution gibt vor, welcher Index t ist und der Angreifer erschleicht für ein quasi-sicheres Bild von *Marabou* ein Zertifikat die Sicherheit, dass, innerhalb einer von *Marabou* mit dem Zertifikat spezifizierten ϵ^l-Verrauschung, aus diesem Bild keine t-Klassifikation entstehen kann. Diese Form des Angriffs nennen wir einen *gerichteten Angriff*, englisch *Targeted Attack*. Aber auch folgendes ist denkbar: Der Angreifer weiß nicht, welches das für die Institution kritische t ist und versucht es „auf gut Glück" mit irgendeiner Klasse s und hofft, dass $s = t$. Diese Art Angriff nennen wir ungerichtet, englisch *Untargeted Attack*. Für die Institution unseres Beispiels sind offenbar beide Angriffsszenarien eine Bedrohung.

Um nun den Angriff fortzusetzen, simulieren wir einen Angriff der zweiten, ungerichteten Art, wonach der Angriff auf ein von l verschiedenes Ziel t geht, in der Hoffnung, dieses t sei das der Institution gefährliche. Konkret wählen wir, ausgehend vom quasisicheren Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$, „auf gut Glück", ein, nach $\hat{y}_l$, zweitgrößtes Vektorelement von $\hat{\boldsymbol{y}}^{\mathbf{qs}}$ als Angriffsziel aus:

$$t := \underset{i \in \{0,\ldots,n-1\} \setminus \{l\}}{\arg\max} (\hat{y}_i) \tag{93}$$

Dass dieses Ziel t zugleich ein Element ist, dessen Abstand, gemessen im L_{CW}-Maß von Formel 92, zu $\hat{y}_l$ minimal ist, ist dabei genau ein Teil des Planes.

Es wurde oben dargelegt, dass $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ eine Robusthheit von höchstens δ^γ haben kann. Der eigentliche Angriff auf den Verifizierer, welcher in unserem Fall *Marabou* ist, startet über den, an diesen gerichten, Auftrag, mittels der am Ende von Kapitel 6.3 eingeführten Methode der Binärsuche, nach der *tatsächlich* größten Mindestrobustheit von N mit Bezug auf das Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ und das zu vermeidende Ziel t. Wie erinnerlich wird dabei von *Marabou*, ein Wert ϵ^l errechnet, der für einen „sicheren Radius" steht, für den *Marabou* „garantiert" – dies ist das Zertifikat –, dass innerhalb dieses Radius um das quasisichere Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ keine Bilder existieren, welche von N aals t klassifiziert würden. *Marabou* errechnet desweiteren einen Wert ϵ^u, der einen „unsicheren" Radius um $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ so darstellt, dass im Inneren des Radius als t-Klassifizierte Bilder liegen. Als „Beweis" für die letzte Behauptung hält *Marabou* eine Belegung aller Variablen bereit, welche ein Eingabebild impliziert, von welchem *Marabou* meint, es würde von N als das „unsichere" t klassifiziert. Dieses Bild wollen wir im Folgenden als *quasi-unsicheres Bild*, $\hat{\boldsymbol{x}}^{\mathbf{qu}}$, bezeichnen, da es zwar, durch seine Eigenschaft von N potentiell als t klassifiziert zu werden, unsicher ist, im Zuge des

Angriffs aber in ein, wohlgemerkt vermeintlich, sicheres Bild transformiert werden soll, worauf später zurückzukommen sein wird.

Dieses quasiunsichere Bild hat nun folgende beachtenswerte Eigenschaft. Es befindet sich in der Nähe des – sofern, wovon wir zunächst einmal ausgehen sollten, *Marabou* bis hierher keine allzu großen Fehler gemacht hat – tatsächlichen Robustheitsradius. Zwischen der Grenze der, nach *Marabou*, sicheren Untergrenze ϵ^l, dem kleineren Radius, und der, gemäß *Marabou*, unsicheren Obergrenze ϵ^u, dem größeren Radius, auf welchem, im, aus Sicht des Angreifers, schlechtesten Falle, das Bild $\hat{\boldsymbol{x}}^{\mathbf{qu}}$ maximal weit von der durch *Marabou* als sicher zertifizierten Grenze ϵ^l liegt, befinden sich höchstens $\epsilon^{\text{precision}}$ Einheiten – das ist der Wert, der im Zuge des Angriffs als Parameter gewählt werden kann und die Unterschreitung dessen das Abbruchkriterium der Binärsuche nach der größten Mindestrobustheit darstellt. Wir können für dieses quasiunsichere Bild also erwarten, dass kleinste Pixel-Perturbationen es mal diesseits, mal jenseits der zertifiziert sicheren Grenze von ϵ^l zurücklassen werden, wobei es vom Netz N mal als t, mal als nicht-t klassifiziert wird. Im Moment befindet $\hat{\boldsymbol{x}}^{\mathbf{qu}}$ sich auf der „unsicheren Seite“, was Abbildung 30 schematisch andeutet.

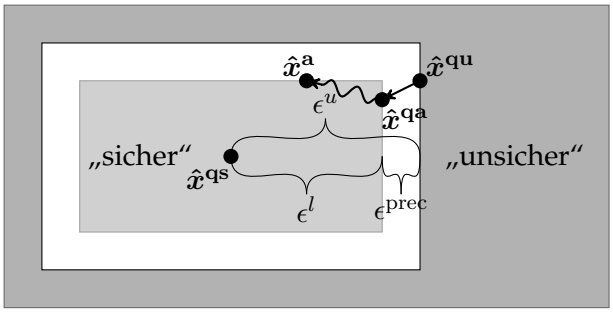


Abbildung 30: Schematisch abgebildet ist die jeweilige Lage der Bilder mit Bezug auf den als sicher geltenden Perturbationsradius ϵ^l respektive den unsicheren Perturbationsradius ϵ^u. Das Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ ist, nach *Marabou*, mindestens bis zu einem Radius von ϵ^l robust, $\hat{\boldsymbol{x}}^{\mathbf{qu}}$ liegt, gemäß *Marabou*, nachweislich nicht in einem sicheren Perturbationsradius. Durch Forcierung des ϵ^l-Radius für $\hat{\boldsymbol{x}}^{\mathbf{qu}}$, welches als t klassifiziert wird, entsteht das quasi-adversarielle Bild $\hat{\boldsymbol{x}}^{\mathbf{qa}}$, welches als nicht-$t$ klassifiziert angenommen wird. Durch pixelweise Manipulation unter Beachtung des ϵ^l-Radius, wird das adversarielle Bild $\hat{\boldsymbol{x}}^{\mathbf{a}}$ innerhalb des ϵ^l-Radius herbeigeführt, welches als t klassifiziert wird.

Der nächste Schritt des Angriffs besteht nun darin, das quasiunsichere Bild $\hat{\boldsymbol{x}}^{\mathbf{qu}}$ in den nicht weit entfernt liegenden „sicheren“ ϵ^l-Radius „zu verschieben“. Für jeden einzelnen Pixel in $\hat{\boldsymbol{x}}^{\mathbf{qu}}$ wird dafür überprüft, ob er größer ist, als der um ϵ^l vergrößerte Wert an der selben Pixelposition im quasisicheren Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$. Ist dies tatsächlich der Fall, wird der Wert von jenem auf den Wert von diesem heruntergesetzt. Analog

werden zu kleine Pixelwerte in $\hat{\boldsymbol{x}}^{\mathbf{qu}}$, die also kleiner sind als der um ϵ^l verringerte Wert an der selben Pixelposition in $\hat{\boldsymbol{x}}^{\mathbf{qs}}$, auf diesen angehoben. Diese „Forcierung" der Einhaltung des durch ϵ^l mit Bezug auf $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ vorgegebenen sicheren Radius führt zu einem neuen Bild, welches wir *quasiadversariales Bild*, kurz $\hat{\boldsymbol{x}}^{\mathbf{qa}}$, nennen wollen. Man vergleiche auch Abbildung 30. Für den, in der Praxis anzutreffenden, Fall, dass *Marabou* ungenau gerechnet hat, ist es bereits ein „adversariales" Bild, das ist, da (sofern) es vom zur Untersuchung stehenden Netz N als t klassifiziert wird, ein Gegenbeispiel für den von *Marabou* als sicher angegebenen Radius ϵ^l, das *Spoofing* des Zertifikats und die Falsifikation *Marabou*s. Wir gehen aber davon aus, dass *Marabou* so ungenau nicht gerechnet hat und, da es innerhalb des sicheren Radius liegt, dass das durch die Forcierung entstandene Bild $\hat{\boldsymbol{x}}^{\mathbf{qa}}$ von N *nicht* als das vermutete „gefährliche" t klassifiziert wird.

Oben hatten wir eine Eigenschaft von Bildern betont, die sich in der Nähe des, von $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ aus betrachtet, „unsicheren" Radius befinden, welche auch für Bilder, die in der Nähe des „sicheren" Radius liegen, gilt: Kleineste Pixel-Perturbationen können bewirken, dass das Bild über die durch den Radius definierte Grenze wandern, aber, und das ist hier das wichtige, auch, dass sich ihre Klassifikation ändert. Das Ziel des nächsten und letzten Schritts des Angriffes besteht nun darin, die Pixel des quasiadversarialen Bildes $\hat{\boldsymbol{x}}^{\mathbf{qa}}$ – unter penibler Beachtung der für diesen Pixel mit Bezug auf $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ als sicher geltenden und durch ϵ^l beschriebenen Perturbationsgrenzen – aus der Klassifikation nicht-t durch N in die Klassifikation t durch N zu überführen. Da, gemäß Annahme, $\hat{\boldsymbol{x}}^{\mathbf{qa}}$ nicht als t klassifiziert wird, misst das in Formel 92 angegebene Maß $L_{\mathrm{CW}}(N(\hat{\boldsymbol{x}}^{\mathbf{qa}}), t)$ einen negativen Wert. Das Ziel ist es, aus Sicht des Angreifers, den Wert dieses Maßes, durch Veränderung der Pixelwerte in $\hat{\boldsymbol{x}}^{\mathbf{qa}}$, in einen positiven Wert zu überführen. Konkret wird dabei wie folgt verfahren.

Wir wählen, als Parameter, den wir Inkrement-Präzision ι, englisch *increment precision*, nennen, die Größe mit der die einzelnen Pixel verändert werden sollen und starten das Verfahren beim ersten Pixel (einer Kopie) von Bild $\hat{\boldsymbol{x}}^{\mathbf{qa}}$ so: Wir addieren auf diesen Pixelwert den Inkrementpräzisionswert ι und überprüfen zunächst, ob etwas gegen diese Inkrementierung spricht. Dies ist insbesondere dann der Fall, wenn der durch die Inkrementierung entstandene Wert des Pixels größer als 1 wird, da Pixelwerte ja nur in den Grenzen zwischen 0 und 1 definiert sind, oder dieser Wert größer ist als der um ϵ^l vergrößerte Wert des Pixels an der selben Stelle im Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$. In jedem dieser Fälle wird die getätigte Veränderung des Pixelwertes zurückgenommen. Liegt hingegen keine derartige Transgression vor, so wird überprüft, ob sich dadurch der Abstand vergrößert hat, ob also der Wert für $L_{\mathrm{CW}}(N(\hat{\boldsymbol{x}}^{\mathbf{qa}}), t)$ gestiegen ist, ist dies nicht der Fall, wird die getätigte Veränderung ebenfalls zurückgenommen. Woraufhin nun noch – unter derselben Beachtung der sich aus der Verringerung um ϵ^l des entsprechenden Pixelwertes im Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ ergebenden Pixelwertgrenze, nur eben jetzt „nach unten", sowie der Verhinderung einer Unterschreitung des Wertes 0 – probiert, ob sich der Abstand durch die Subtraktion des Wertes ι vom ursprünglichen Pixelwert in $\hat{\boldsymbol{x}}^{\mathbf{qa}}$ der Abstand verringern lässt. Konnte – sei es durch Inkrementierung, sei es durch Dekrementierung des Pixelwertes – nun auf

diese Weise eine Abstandsverringerung erreicht werden, wird sie fix gemacht und der Wert für das jeweilige Pixel (der Kopie) von $\hat{\boldsymbol{x}}^{\mathbf{qa}}$ entsprechend aktualisiert. Nun wiederholen wir das Verfahren für den nächsten Pixel. Sind wir beim letzten Pixel, konkret dem 783sten Pixel angekommen, beginnen wir das Verfahren anschließend erneut beim ersten Pixel. Aufgrund der Art, wie eine Abstandsvergrößerung pro Pixel möglichst erreicht werden soll, nämlich das winzige Erhöhen, anschließend, gegebenenfalls, Herabsenken des Pixelwertes, kann damit leicht die Vorstellung des „An-den-Pixeln-Wackelns“, oder, englisch *jiggling*, assoziiert werden.

Ein mögliches Abbruchkriterium des Verfahrens ist der Umstand, dass in einem Durchlauf durch den ersten bis zum letzten Pixel von $\hat{\boldsymbol{x}}^{\mathbf{qa}}$ für keinen der Pixel eine Vergrößerung des $L_{\mathrm{CW}}(N(\hat{\boldsymbol{x}}^{\mathbf{qa}}), t)$-Wertes erreicht werden konnte. In diesem Falle, sofern $L_{\mathrm{CW}}(N(\hat{\boldsymbol{x}}^{\mathbf{qa}}), t)$, wie zu Beginn einen negativen Wert hat, ist der Angriff gescheitert, da das Bild, wie von *Marabou* prognostiziert, von N als nicht-t klassifiziert wird.

Ein dem Angreifer wohl erfreulicheres Abbruchkriterium des Verfahrens ist offensichtlich, dass das Abstandsmaß $L_{\mathrm{CW}}(N(\hat{\boldsymbol{x}}^{\mathbf{qa}}), t)$ den Wert 0 überschreitet: Das nunmehr *adversariales Bild*, kurz $\hat{\boldsymbol{x}}^{\mathbf{a}}$, genannte Bild wird, obwohl es innerhalb des von *Marabou* als sicher zertifizierten Radius um $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ liegt, dennoch als t klassifiziert. Ein Angreifer hat sich, den Angriff zusammenfassend, damit ein Zertifikat von *Marabou* erschlichen, dass die Robustheit des quasisicheren Bildes $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ bis inklusive der Robustheitsgrenze von ϵ^l zertifiziert. Da das adversariale Bild $\hat{\boldsymbol{x}}^{\mathbf{a}}$, wie sich ohne großen Aufwand, etwa für unsere Institution des Szenarios, zu verursachen, nachprüfen lässt, in den als sicher zertifizierten ϵ^l-Grenzen liegt, und somit, gemäß Zertifikat, nicht als t klassifiziert werden dürfte, wäre dem Angreifer in diesem Fall und in diesem Szenario also der – ungerichtete – *Certificate Spoofing Attack* gelungen.

Um einen *gerichteten* Angriff zu fahren, in welchem der Angreifer das Ziel t nicht „auf gut Glück“ sondern gezielt auswählt, könnte man geneigt sein, anzunehmen, das Kontrastbild würde sich für die diesbezügliche Zielauswahl anbieten: Wenn wir ein als 0 klassifiziertes Ausgangsbild erzeugen wollen, für das sich ein Zertifikat für die nicht-1 erschleicht lässt, so die Überlegung, müssten wir doch einfach das Originalbild $\hat{\boldsymbol{x}}^{\mathbf{o}}$, welches die 0 zeigt, mit einem Kontrastbild $\hat{\boldsymbol{x}}^{\mathbf{c}}$, welches die 1 zeigt und vom Netz N auch entsprechend klassifiziert wird, überblenden können, wobei dann, so die Annahme, das durch $\gamma^\top$ ermischte Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}} = \gamma^\top \cdot \hat{\boldsymbol{x}}^{\mathbf{o}} + (1 - \gamma^\top) \cdot \hat{\boldsymbol{x}}^{\mathbf{c}}$ gerade noch so als 0 klassifiziert wird, aber das mit dem um δ^γ kleinereren $\gamma^\perp$ ermischte Bild $\gamma^\perp \cdot \hat{\boldsymbol{x}}^{\mathbf{o}} + (1 - \gamma^\perp) \cdot \hat{\boldsymbol{x}}^{\mathbf{c}}$ gerade nicht mehr als 0, sondern als 1 klassifiziert wird. Diese Art der Lösung des Vagheitsproblems – Wann zeigt eine Bild noch eine Null und wann schon eine Eins? – wird von Neuronalen Netzen, vielleicht im Unterschied zur menschlichen Klassifikation, allerdings sehr kreativ entschieden. Dass bei den mit einem Gammawert $0 \leq \gamma^\top \leq 1$ aus $\hat{\boldsymbol{x}}^{\mathbf{o}}$ und $\hat{\boldsymbol{x}}^{\mathbf{c}}$ ermischten Bildern $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ immer die ursprüngliche Klassifizerung erhalten bleibt, dafür garantiert die bei der Binärsuche stets eingehaltenen Invariante. Nichts garantiert jedoch, dass dies auch für die mit einem Gammawert $0 \leq \gamma^\perp \leq 1$ aus $\hat{\boldsymbol{x}}^{\mathbf{o}}$ und $\hat{\boldsymbol{x}}^{\mathbf{c}}$ ermischten

Bildern $\gamma^{\perp} \cdot \hat{\boldsymbol{x}}^{\mathbf{o}} + (1 - \gamma^{\perp}) \cdot \hat{\boldsymbol{x}}^{\mathbf{c}}$ bezüglich der ursprünglichen Klassifizierung, hier: als die Ziffer 1, für $\hat{\boldsymbol{x}}^{\mathbf{c}}$ gilt. Tatsächlich „erkennt“ das Neuronale Netz N, während der Gammawert von 1 kontinuiertlich auf 0 gesenkt wird, auf kreative Weise zuweilen, je nach Netz und Bildkombinationen, alle möglichen Ziffern außer der ursprünglichen, für $\gamma = 1$, Ziffer Null und der finalen, für $\gamma = 0$, Ziffer Eins. Ein auf diese Weise gerichteter Angriff, ein *Targeted Certificate Spoofing Attack*, ist demnach zwar im Einzelfall möglich, hängt aber davon ab, ob das Netz die Eigenschaft hat, für die kontinuierliche Überblendung des Originalbildes durch das Kontrastbild auch eine kontinuiertliche Änderung der Klassifikation bereitzuhalten; andernfalls ist diese Form des Angriffs bloß eine Variante der Attacke „auf gut Glück“. Zu verifizieren, ob ein Netz für das entsprechende Original- und Kontrast-Bild diese Eigenschaft besitzt, wäre eine Aufgabe für einen verifizierten Verifizierer Neuronaler Netze.

9.4 Anwendung des Jia-Rinard-Exploits auf Marabou

Abschließend wollen wir dokumentieren, dass ein, wie im letzten Abschnitt beschriebener und in Abbildung 29 in seiner Vollständigkeit zusammengefasster, von Jia und Rinard inspirierter, Angriff auf *Marabou* bei entsprechender Wahl der Parameter „erfolgreich“ durchgeführt werden kann, was die Falsifikation *Marabou*s in einem praxisnahen Beispiel impliziert und die Bedenklichkeit seiner Verwendung in sicherheitskritischen Kontexten verdeutlicht.

Zunächst wollen wir aber die Parameterwerte für den in diesem Abschnitt zu dokumentierenden Angriff auf *Marabou* diskutieren. Unser Netz N wird ein doppeltpräziser, also auf 64-Bitfließkommazahlen basierender, *MNIST*-Klassifizierer sein, mit, selbstredend, $(28 \times 28) = 784$ Knoten in der Eingabeschicht und 10, der Anzahl der unterscheidbaren Ziffern, Knoten in der Ausgabeschicht sowie einem einzigen *Hidden Layer*, welcher 10 Knoten umfasst. Es handelt sich dabei um das bereits in Kapitel 2.5 vorgestellte Modell `model784x10x10_float64_9355`. Von den insgesamt $10\,000$ Testbildern des *MNIST*-Datensatzes vermag es $9\,355$ davon korrekt zu klassifizieren. Als Originalbild wählen wir eine handgeschriebene Ziffer Null des *MNIST*-Trainingsdatensatzes, konkret die Ziffer an Index-Position $54\,927$ dieses Datensatzes sowie eine handgeschriebene Ziffer Eins an Index-Position $6\,356$ desselben Datensatzes. Die Auswahl der Indizes ist ästhetisch begründet: Die Überblendung beider Ziffern erinnert an ein „Φ“, wie Φιλία [Philía], „Freundschaft“, was einen höheren kulturellen Wert verspricht, als die Überblendung der 5 mit einer 1.

Als Präzision für die ϵ-Binärsuche, wählen wir $\epsilon^{\text{precision}} := 10^{-6}$ als Mindestdistanz für den Abstand des „sicheren“ Radius ϵ^{l} zum „unsicheren Radius“ ϵ^{u}; man vergleiche hierfür auch Abbildung 30. Diese Wahl lässt sich dadurch rechtfertigen, dass wir die Einstellung einer maximalen Robustheit – über den gleich zu diskutierenden Parameter δ^{γ} – in ähnlicher Größe vornehmen möchten. Es erhellt, im Hinblick auf Abbildung 30, dass je größer der Parameter $\epsilon^{\text{precision}}$ gewählt wird, desto schwieriger der Angriff wird, da der Abstand zwischen $\hat{\boldsymbol{x}}^{\mathbf{qu}}$ und $\hat{\boldsymbol{x}}^{\mathbf{qa}}$ größer wird. Wird der Wert für $\epsilon^{\text{precision}}$, in Relation zu δ^{γ}, aber zu groß gewählt, etwa als 10^{-5},

so findet *Marabou* aus der Binärsuche nur die, in der Tat, „sichere“ Untergrenze $\epsilon^l = 0$, welche unangreifbar ist. Unsere Parameterwahl ergibt sich also als der größte mögliche Wert, der Größenordnung nach, für den, bei der, gleich vorzustellenden, Wahl des Parameterwertes für δ^γ, *Marabou* nicht $\epsilon^l = 0$ als Ergebnis der Binärsuche berechnet.

Für den Parameterwert δ^γ nun, welcher über die Binärsuche nach dem bildmischenden Faktor $\gamma^\top$, der das quasisichere Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}} = \gamma^\top \cdot \hat{\boldsymbol{x}}^{\mathbf{o}} + (1 - \gamma^\top) \cdot \hat{\boldsymbol{x}}^{\mathbf{c}}$ erzeugt, welches, wie im letzten Abschnitt besprochen, höchstens die Robustheit von δ^γ haben kann, empfehlen, Jia und Rinard – auf deren entsprechendem Aufsatz, [JR20], der vorgestellte Angriff letztendlich, wie erinnerlich, basiert – einen Wert von 10^{-7}, allerdings vor dem Hintergrund der Überblendung eines Ziffernbildes mit einem gänzlich weißen Bild, was eine leicht größere Mindestrobustheit des ermischten Bildes implizieren könnte. Je kleiner der Wert dieses Parameters, desto kleiner die maximale Robustheit und – so das entsprechende Ergebnis des Aufsatzes [JR20] – desto erfolgsversprechender der Angriff auf den Verifizierer. Wir wählen einen *größeren* als den von Jia und Rinard empfohlenen Wert und setzen $\delta^\gamma := 10^{-6}$.

Für die Inkrementierungspräzision ι, welche, wie im letzten Abschnitt diskutiert, angibt, wie stark wir an den Pixeln des quasi-adversarialen Bildes $\hat{\boldsymbol{x}}^{\mathbf{qa}}$ „wackeln“, um daraus, hoffentlich, schließlich das adversariale Bild $\hat{\boldsymbol{x}}^{\mathbf{a}}$ zu erzeugen, wählen wir initial $\iota := 10^{-12}$. Um das Verfahren zu beschleunigen, skalieren wir diesen Wert, nach jedem Wackel- oder *Jiggle*-Durchlauf – in dem für jeden Pixel des Bildes einmal eine Inkrementierung, dann, gegebenenfalls, eine Dekrementierung um ι, wie im letzten Abschnitt dargelegt, probiert wird, um dadurch eine L_{CW}-Distanzvergrößerung, im Sinne der Formel 92, zu erreichen –, in welchem bei mindestens einem Pixel eine Distanzvergrößerung erreicht werden konnte, um den Faktor 2 nach oben. Kann erstmalig keine Distanzvergrößerung „erwackelt“ werden, so halbieren wir von dort an für jeden derartigen Fall – in einem Durchlauf durch alle Pixel des Bildes konnte keine Distanzvergrößerung erreicht werden – den ι-Wert bis, als Abbruch-Kriterium, der ι-Wert den Ausgangswert von $\iota = 10^{-12}$ unterschritten hat. Tatsächlich wollen wir auch dann noch „weiterwackeln“, wenn die Distanz längst positiv – und der Angriff bereits als erfolgreich feststeht – geworden ist, um gegebenenfalls feststellen zu können, wie viel „Vorsprung“ wir der Klassifizierung von t vor den anderen Klassen auf diese Weise verschaffen können. Es sei betont, dass die Wahl des ι-Parameters, keinerlei Einfluss auf die Berechnungen *Marabou*s hat, da zum Zeitpunkt an welchem wir durch das *Jiggling* versuchen, aus dem quasiadversarialen Bild das adversariale Bild zu erzeugen, alle Berechnungen *Marabou*s bereits abgeschlossen sind. Unsere Wahl des ι-Parameters bedarf daher keiner Rechtfertigung.

Um den gleich dokumentierten Angriff noch überzeugender zu machen, gestatten wir *Marabou*, sich bei der Berechnung des Zertifikats *um eine ganze Größenordnung* verrechnet haben zu dürfen. Konkret verschieben wir den von *Marabou* für das quasisichere Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ als sicheren Radius – innerhalb dessen, ausgehend vom quasisicheren Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ kein Bild $\hat{\boldsymbol{x}}^{\mathbf{a}}$ existiert, das um höchstens ϵ^l Einheiten pro Pi-

xel von jenem abweicht und als t klassifiziert wird – bestimmten Wert ϵ^l, nachdem dieser von *Marabou* durch die Binärsuche errechnet wurde, um eine Dezimalkommastelle nach rechts, sprich: wir setzen $\epsilon^l := (\epsilon^l/10)$.

Damit ist die Wahl der Parameter spezifiziert und wir begleiten den im folgenden dokumentierten Angriff kommentierend. Zunächst werden die beiden Ausgangsbilder, das Originalbild und das Kontrastbild, englisch *Blending Image*, noch einmal, wie in Abbildung 31, vorgestellt.

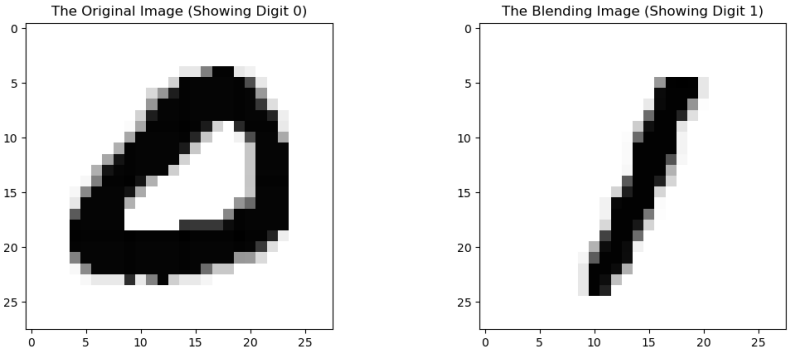


Abbildung 31: Vor dem Angriff. Links: Das Originalbild. Rechts: Das Kontrastbild.

Es folgt die Binärsuche nach δ^γ, woraus schließlich, wie in Abbildung 32, nebst seiner Klassifikation, dargestellt, das quasi-sichere Bild, englisch *Quasi-Safe Image*, ermischt wird.

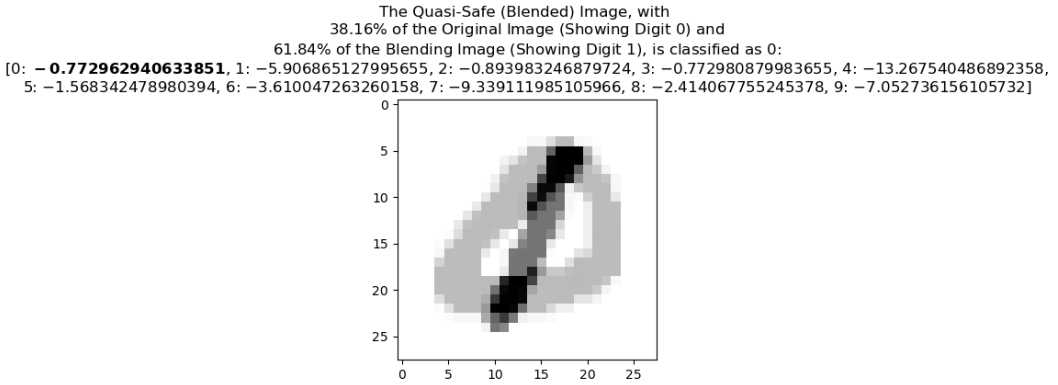


Abbildung 32: Aus dem Originalbild und dem Kontrastbild wurde das abgebildete quasi-sichere Bild gemischt, welches gerade noch so als die Ziffer 0 klassifiziert wird, aber fast schon als die Ziffer 3.

Es lässt sich, aus dem dargestellten Klassifikationsvektor entnehmen, dass es noch

als Null klassifiziert wird, das Netz darin aber schon fast auch die Ziffer Drei erkennt. Damit steht zugleich fest, dass der Angriff auf das Ziel, englisch *target*, $t := 3$ gehen wird. Folglich handelt es sich künftig um einen *untargeted attack*, da als das ursprüngliche, gerichtete, Ziel die Ziffer 1, das Label des Kontrastbildes, gegolten haben mag. Es folgen einige graphisch aufbereitete Abstände der Pixelwerte der verschiedenen Bilder untereinander auf die in Abbildung 33 dargestellte, aufbereitete Weise, jeweils in Form einer *Heatmap*.

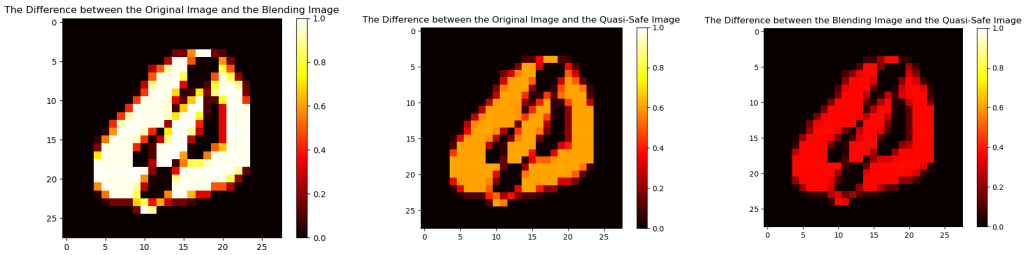


Abbildung 33: Pixelweise Abstände zwischen verschiedenen Bildern. Links: Abstand zwischen Originalbild und Kontrastbild. Mitte: Abstand zwischen Originalbild und dem quasi-sicheren Bild. Rechts: Abstand zwischen Kontrastbild und dem quasi-sicheren Bild.

Nun beginnt *Marabou* mit der Binärsuche nach ϵ^l und ϵ^u und einem Gegenbeispiel der Robustheit des Netzes für das quasi-sichere Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$. Dieses Gegenbeispiel, wird schließlich in Abbildung 34 präsentiert. Es ist das quasi-unsichere Bild $\hat{\boldsymbol{x}}^{\mathbf{qu}}$, welches, wenn man *Marabou* glauben schenken mag, auch – hier insbesondere: nur – als die Ziffer 3 klassifiziert wird. Für ϵ^l wurde im Zug der Binärsuche konkret der, zwecks Darstellung von uns hier auf sechs Nachkommastellen gerundete, Wert $\epsilon^l = 2.861023 \cdot 10^{-6}$ ermittelt – welcher den um eine Größenordnung kleineren Radius $(\epsilon^l/10) = 2.861023 \cdot 10^{-5}$, also lediglich 10% des ohnehin schon von *Marabou* als sicher zertifizierten Radius, impliziert, mit Bezug auf welchen unser Angriff läuft – sowie $\epsilon^u = 3.814697e \cdot 10^{-6}$, wiederum auf sechs Nachkommastellen gerundet.

Da der Unterschied zwischen dem quasi-sicheren und dem quasi-unsicheren Bild pixelweise so gering ist, dass beide Bilder mit bloßem Auge nicht unterschieden werden können, wird durch die in Abbildung 35 gezeigte *Heatmap* die vollzogene Änderung verdeutlicht. Eine derartige „Änderung“ zwischen einem ursprünglichen Bild $\boldsymbol{x^1} := (x_0^1, \ldots, x_{783}^1)^\top$ und einem veränderten Bild $\boldsymbol{x^2} := (x_0^2, \ldots, x_{783}^2)^\top$ bezeichnet dabei, hier und im Folgenden stets, den jeweils pixelweise absoluten Abstand, wodurch wiederum ein Differenzen- oder Abstands-Bild impliziert wird: $\boldsymbol{x^3} := (|x_0^1 - x_0^2|, \ldots, |x_{783}^1 - x_{783}^2|)^\top$.

Dieses quasi-unsichere Bild wird nun – durch die entsprechende Forcierung seiner einzelnen Pixelwerte in den, bezogen auf das quasi-sichere Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$, *extrasicheren* Radius $\epsilon^l/10$ – zum quasi-adversarialen Bild überführt. Zusätzlich dazu ist

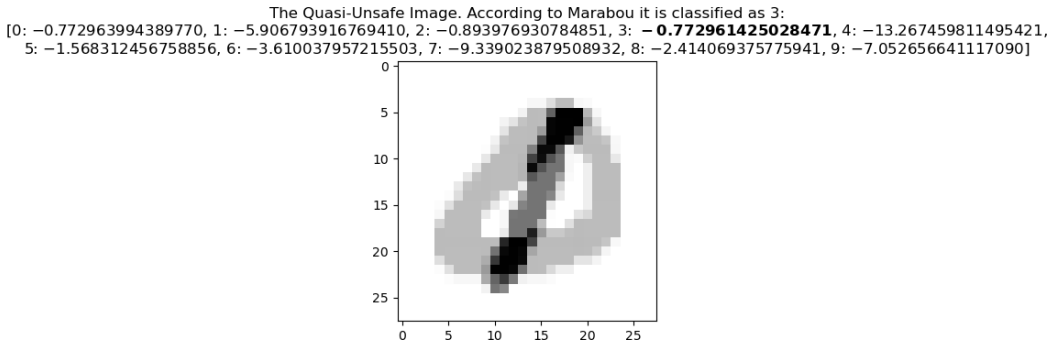


Abbildung 34: Das quasi-unsichere Bild innerhalb des unsicheren Radius ϵ^u um $\hat{\boldsymbol{x}}^{\mathbf{qs}}$, wird als das Ziel $t = 3$ klassifiziert.

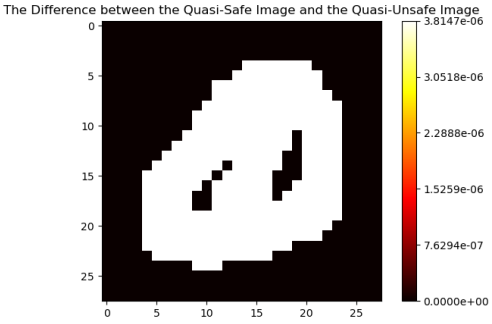


Abbildung 35: Eine *Heatmap* des auf den Maximalwert skalierten Abstands zwischen dem quasi-sicheren und dem quasi-unsicheren Bild.

in Abbildung 36 die *Heatmap* der vollzogenen Änderung mitangegeben.

Nun beginnt, für unseren Angriff, der langwierigste Teil des Verfahrens: Die „Verwackelung“ der Pixel, in der Hoffnung, ein adversariales Bild $\hat{\boldsymbol{x}}^{\mathbf{a}}$ zu erhalten. Am Ende der Prozedur ist ein solches, wie in Abbildung 37 abgebildetes, adversariales Bild, welches als $t = 3$ klassifiziert wird, tatsächlich gefunden worden. Diese, der Abbildung entnehmbare, Klassifikation ist recht deutlich: Der Wert -0.772941, die Klassifikation für die Ziffer Drei, und der Wert -0.773007, die Klassifikation für die Ziffer Null, unterscheiden sich, allerdings natürlich in der gegenteiligen Richtung, stärker voneinander als die entsprechenden Klassifikationen für das quasi-sichere Ausgangsbild des Angriffs $\hat{\boldsymbol{x}}^{\mathbf{qs}}$. Das Netz ist sich, anschaulich zusammengefasst, also „sicherer“, dass das adversariale Bild die Ziffer Drei zeigt (und nicht die Ziffer Null), als dass das quasi-sichere Bild die Ziffer Null (und nicht die Ziffer Drei) zeigt.

Seine in Abbildung 38 als *Heatmap* aufbereitete Differenz zum quasi-adversarialen Bild zeigt deutlich das Resultat des *Jigglings*.

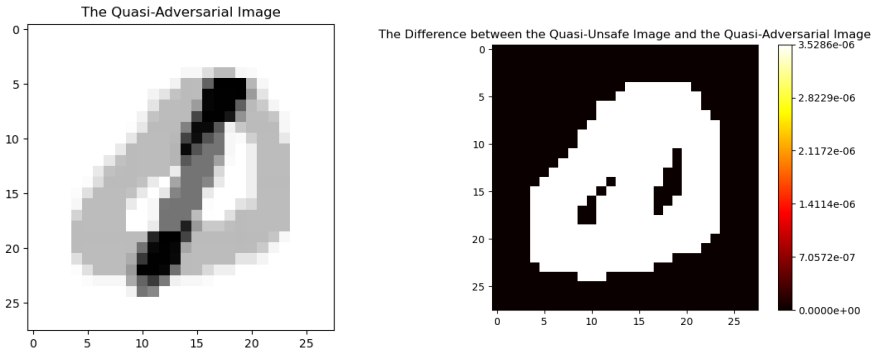


Abbildung 36: Das quasi-adversariale Bild und eine *Heatmap* des auf den Maximalwert skalierten Abstands zwischen dem quasi-unsicheren und dem quasi-adversarialen Bild.

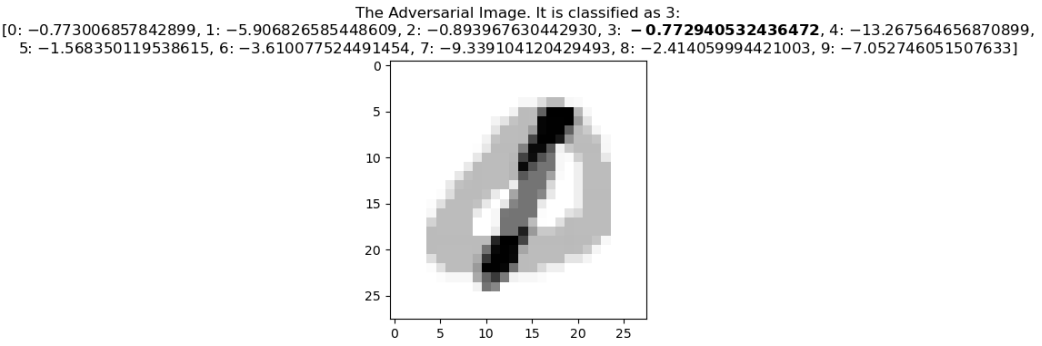


Abbildung 37: Das adversariale Bild wird, obwohl es innerhalb des von *Marabou* als sicher bescheinigten ϵ^l-Radius um $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ liegt, als $t = 3$ klassifiziert.

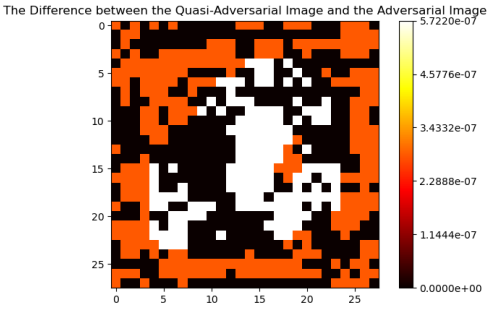


Abbildung 38: Die *Heatmap* des auf den Maximalwert skalierten Abstands zwischen dem quasi-adversarialen und dem adversarialen Bild offenbart die Positionen und den jeweiligen Betrag der Pixel-„Verwackelung“.

Die endlich in Abbildung 39 wiederum als *Heatmap* aufbereitete Differenz zwischen dem Ausgangsbild des Angriffs, dem quasi-sicheren Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ und dem finalen Bild, welches dem Zertifikat *Marabou*s zuwiderläuft, das ist, das adversariale Bild $\hat{\boldsymbol{x}}^{\mathbf{a}}$, zeigt, dass der Angriff erfolgreich war, selbst wenn wir für die pixelweise Abweichung des adversarialen Bildes vom quasi-sicheren Bild nur 10% des von *Marabou* als sicher bescheinigten ϵ^l-Radius verwenden.

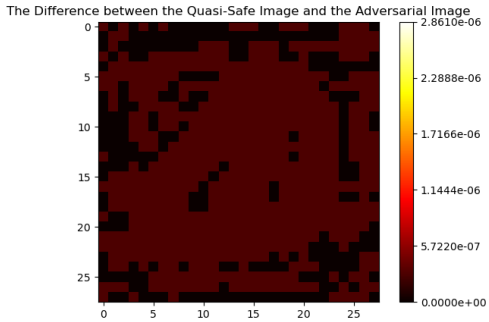


Abbildung 39: Die *Heatmap* des auf ϵ^l als Maximalwert skalierten Abstands zwischen dem quasi-sicheren und dem adversarialen Bild stellt heraus, dass die um eine Größenordnung schärferen $(\epsilon^l/10)$-Grenzen bei der „Verwackelung“ eingehalten wurden.

Dass die in Abbildung 39 als nicht-schwarz dargestellten Pixel schließlich *alle* nahe dieses 10%-Wertes der Farbskala in Dunkelrot erscheinen, sollte uns nicht wundernehmen, da wir unseren Angriff immerhin so konzipiert hatten, dass so lange „weitergewackelt“ werden soll, bis letztlich keine Distanzvergrößerung mehr erreicht werden kann.

Der Angriff war erfolgreich. Ein Angreifer hat von *Marabou* ein Zertifikat für das quasisichere Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ erhalten, wonach dieses bis zu einer Pixelverrauschung von $\epsilon^l = 2.861023 \cdot 10^{-6}$ sicher nicht als Klasse $t = 3$ klassifiziert wird. Der Institution präsentiert der Angreifer nun das quasi-sichere Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$ und das von *Marabou* dafür ausgestellte Zertifikat. Anschließend wird das adversariale Bild $\hat{\boldsymbol{x}}^{\mathbf{a}}$ vorgelegt, zum Zwecke der Eingabe in den institutionseneigenen Klassifizierer, der – so die eingangs skizzierte Bedrohungslage des Szenarios – das Bild $\hat{\boldsymbol{x}}^{\mathbf{a}}$ *unter keinen Umständen* als t klassifizieren darf. Da aber das Bild $\hat{\boldsymbol{x}}^{\mathbf{a}}$ sogar besonders sicher zu sein scheint – schließlich, so stellt die Institution schnell fest, verwendet es nur 10% der maximal für das Bild $\hat{\boldsymbol{x}}^{\mathbf{qs}}$, laut Zertifikat, zulässigen ϵ^l-Verrauschung –, steht, aus Sicht der Institution, einer Eingabe des adversarialen Bildes $\hat{\boldsymbol{x}}^{\mathbf{a}}$ in den institutionseneigenen Verifizierer nichts entgegen, welches prompt als t klassifiziert wird. Damit hat die Verwendung von *Marabou* als Verifizierer durch ein fälschlicherweise ausgestelltes Unbedenklichkeitszertifikat in der – durch unser Szenario angedeuteten – Praxis Schaden verursacht, was – die vorliegende Arbeit abschließend – einer, von der Institution des Szenarios teuer erkauften, erneuten Falsifikation *Marabou*s entspricht.

Literatur

[Alb21] ALBARGHOUTHI, AWS: *Introduction to Neural Network Verification*. Foundations and Trends® in Programming Languages, 7(1–2):1–157, 2021.

[Ari20] ARISTOTELES: *Metaphysik, Buch I–VII. Übersetzt und erläutert von Dr. theol. Eugen Rolfes*. Verlag von Felix Meiner, Leipzig, Zweite Auflage, 1920.

[Bim20] BIMBÓ, KATALIN: *Combinatory Logic*. In: ZALTA, EDWARD N. (Herausgeber): *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2020 Auflage, 2020.

[BJS04] BAZARAA, MOKHTAR S., JOHN J. JARVIS und HANIF D. SHERALI: *Linear Programming and Network Flows*. Wiley-Interscience, 2004.

[BM07] BRADLEY, AARON R. und ZOHAR MANNA: *The Calculus of Computation — Decision Procedures with Applications to Verification.* Springer, 2007.

[Bol96] BOLTZMANN, LUDWIG: *Vorlesungen über Gastheorie, I. Theil: Theorie der Gase mit einatomigen Molekülen, deren Dimensionen gegen die mittlere Weglänge verschwinden*. Verlag von Johann Ambrosius Barth (Arthur Meiner), Leipzig, 1896.

[Dan90a] DANTZIG, GEORGE B.: *The Diet Problem*. Informs, Journal On Applied Analytics, 12(4):43–47, July–August 1990.

[Dan90b] DANTZIG, GEORGE B.: *Origins of the Simplex Method*, Seite 141–151. Association for Computing Machinery, New York, NY, USA, 1990.

[DOW55] DANTZIG, GEORGE BERNARD, ALEX ORDEN und PHILIP S. WOLFE: *The Generalized Simplex Method for Minimizing a Linear form under Linear Inequality Constraints*. Pacific Journal of Mathematics, 5(2):183–195, 1955.

[Fey97] FEYERABEND, PAUL: *Zeitverschwendung*. Suhrkamp, Berlin, 1997.

[Flo67] FLOYD, ROBERT W.: *Assigning Meanings to Programs*. Proceedings of Symposium on Applied Mathematics, 19:19–32, 1967.

[Fre79] FREGE, GOTTLOB: *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Verlag von Louis Nebert, Halle, 1879.

[Fre84] FREGE, GOTTLOB: *Die Grundlagen der Arithmetik. Eine logisch mathematische Untersuchung über den Begriff der Zahl*. Verlag von Wilhelm Koebner, Breslau, 1884.

[Fre19] FREGE, GOTTLOB: *Der Gedanke*. Beiträge zur Philosophie des Deutschen Idealismus, 2:58–77, 1918-1919.

[GBC16] GOODFELLOW, IAN, YOSHUA BENGIO und AARON COURVILLE: *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[Hoa69] HOARE, C. A. R.: *An Axiomatic Basis for Computer Programming*. Communications of the ACM, 12(10):576–583, Oktober 1969.

[IEE19] IEEE: *Standard for Floating-Point Arithmetic*. IEEE Std 754-2019 (Revision of IEEE 754-2008), Seiten 1–84, 2019.

[JR20] JIA, KAI und MARTIN C. RINARD: *Exploiting Verified Neural Networks via Floating Point Numerical Error*. CoRR, abs/2003.03021, 2020.

[KBD+17] KATZ, GUY, CLARK W. BARRETT, DAVID L. DILL, KYLE JULIAN und MYKEL J. KOCHENDERFER: *Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks*. In: *Proceedings of the 29th International Conference on Computer Aided Verification, Part I*, Seiten 97–117. Springer, 2017.

[KHI+19] KATZ, GUY, DEREK A. HUANG, DULIGUR IBELING, KYLE JULIAN, CHRISTOPHER LAZARUS, RACHEL LIM, PARTH SHAH, SHANTANU THAKOOR, HAOZE WU, ALEKSANDAR ZELJIĆ, DAVID L. DILL, MYKEL J. KOCHENDERFER und CLARK BARRETT: *The Marabou Framework for Verification and Analysis of Deep Neural Networks*. In: DILLIG, ISIL und SERDAR TASIRAN (Herausgeber): *Computer Aided Verification*, Seiten 443–452, Cham, 2019. Springer International Publishing.

[Kle67] KLEENE, STEPHEN COLE: *Mathematical Logic*. Wiley, New York [u.a.], 1967.

[Knu74] KNUTH, DONALD E.: *Structured Programming with go to Statements*. Computing Surveys, 6:261–301, 1974.

[KS08] KROENING, DANIEL und OFER STRICHMAN: *Decision Procedures: An Algorithmic Point of View*. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin, Heidelberg, Erste Auflage, 2008.

[LBH15] LECUN, Y., Y. BENGIO und G. HINTON: *Deep Learning*. Nature, 521:436–444, 2015.

[ML23] MARGOLIS, ERIC und STEPHEN LAURENCE: *Concepts*. In: ZALTA, EDWARD N. und URI NODELMAN (Herausgeber): *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2023 Auflage, 2023.

[NH10] NAIR, VINOD und GEOFFREY E. HINTON: *Rectified Linear Units Improve Restricted Boltzmann Machines*. In: *ICML 2010*, Seiten 807–814, 2010.

[Nie15] NIELSEN, MICHAEL A.: *Neural Networks and Deep Learning*. Determination Press, 2015. http://neuralnetworksanddeeplearning.com/.

[Pop65] POPPER, KARL RAIMUND: *Conjectures and Refutations: The Growth of Scientific Knowledge*. Routledge, New York, 1965.

[PT10] PULINA, LUCA und ARMANDO TACCHELLA: *An Abstraction-Refinement Approach to Verification of Artificial Neural Networks*. In: *International Conference on Computer Aided Verification*, 2010.

[Qui50] QUINE, WILLARD VAN ORMAN: *Methods of Logic*. Harvard University Press, New York, NY, USA, 1950.

[Ram27] RAMSEY, FRANK P.: *Facts and Propositions*. Aristotelian Society Supplementary Volume 7, 1:153–170, 1927.

[Ros57] ROSENBLATT, FRANK: *The Perceptron — A Perceiving and Recognizing Automaton*. Technischer Bericht 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York, January 1957.

[Rud76] RUDIN, WALTER: *Principles of Mathematical Analysis*. International Series in Pure and Applied Mathematics. McGraw-Hill, 1976.

[Sch24] SCHÖNFINKEL, M.: *Über die Bausteine der mathematischen Logik*. Mathematische Annalen, 92:305–316, 1924.

[Sch01] SCHÖNING, UWE: *Logic for Computer Scientists*, Band 8 der Reihe *Progress in Computer Science and Applied Logic*. Birkhäuser, Boston [u.a.], Vierte Auflage, 2001.

[SDD+18] SESHIA, SANJIT A., ANKUSH DESAI, TOMMASO DREOSSI, DANIEL FREMONT, SHROMONA GHOSH, EDWARD KIM, SUMUKH SHIVAKUMAR, MARCELL VAZQUEZ-CHANLATTE und XIANGYU YUE: *Formal Specification for Deep Neural Networks*. Technischer Bericht UCB/EECS-2018-25, EECS Department, University of California, Berkeley, May 2018.

[Sen81] SEN, AMARTYA: *Poverty and Famines: An Essay on Entitlement and Deprivation*. Clarendon Press, Oxford, 1981. New York: Oxford University Press, 1981; New Delhi: Oxford University Press, 1982.

[Smi03] SMITH, PETER: *An Introduction to Formal Logic*. Cambridge University Press, New York, 2003.

[SZS+14] SZEGEDY, CHRISTIAN, WOJCIECH ZAREMBA, ILYA SUTSKEVER, JOAN BRUNA, DUMITRU ERHAN, IAN J. GOODFELLOW und ROB FERGUS: *Intriguing properties of neural networks.* In: BENGIO, YOSHUA und YANN LECUN (Herausgeber): *ICLR (Poster)*, 2014.

[Tar36] TARSKI, ALFRED: *Über den Begriff der logischen Folgerung*. In: *Actes du Congrès International de Philosophie Scientifique, Fasc. VII*, Band 394 (Actualités Scientifiques et Industrielles), Seiten 1–11. Librairie scientifique Herrmann & Cie, Paris, 1936.

[Wey12] WEYL, HERMANN: *Levels of Infinity: Selected Writings on Mathematics and Philosophy*. Mineola, New York, Dover Publication Inc., 2012.

[Wit22] WITTGENSTEIN, LUDWIG: *Tractatus Logico-Philosophicus*. London: Routledge, 1981, 1922.

[Wit53] WITTGENSTEIN, LUDWIG: *Philosophische Untersuchungen*. Suhrkamp Verlag, Frankfurt am Main, 1953.

[WR27] WHITEHEAD, ALFRED NORTH und BERTRAND RUSSELL: *Principia Mathematica*. Cambridge University Press, 1925–1927.

[Zak04] ZAKON, ELIAS: *Mathematical Analysis, Volume I*. The Zakon Series on Mathematical Analysis. The Trillia Group, West Lafayette, Indiana, USA, 2004.

[Zus60] ZUSE, KONRAD: *Entwicklungslinien einer Rechengeräte-Entwicklung von der Mechanik zur Elektronik*. Konrad Zuse Internet Archive, Document - ZIA ID: 0086:1–28, ca. 1960.