Bachelorarbeit

Ansätze zur Situationsbewertung

Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Lehrstuhl Technologie der Informationssysteme

Vorgelegt von:
Daniel Philipp Alexander Niecke

Betreuender Hochschullehrer: Prof. Dr. rer. nat. habil. Bernhard Thalheim
Betreuer: Marina Tropmann-Frick

Kiel, 21.09.2016

Aufgabe

Name, Vorname:	Niecke, Daniel Philipp Alexander
Immatrikulations-Nr:	1012491
Studiengang:	Wirtschaftsinformatik
Betreuender Hochschullehrer:	Prof. Dr. rer. nat. habil. Bernhard Thalheim
Betreuer:	Marina Tropmann-Frick
Institut:	Institut für Informatik
Arbeitsgruppe:	Technologie der Informationssysteme

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt und keine anderen, als die angegebenen Quellen und Hilfsmittel, verwendet habe.
Die eingereichte schriftliche Fassung der Arbeit entspricht der auf dem elektronischen Speichermedium.
Weiterhin versichere ich, dass diese Arbeit noch nicht als Abschlussarbeit an anderer Stelle vorgelegen hat.

.....
Daniel Philipp Alexander Niecke

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einführung

In der vorliegenden Arbeit möchte ich Ansätze zur Situationsbewertung erläutern und versuchen, einen Überblick über die Thematik zu geben. In diesem Kapitel werde ich sowohl auf die Motivation hinter dieser Arbeit, als auch auf die Ziele und die Struktur der Arbeit eingehen.

1.1. Motivation

Situationen sind in unserer Welt allgegenwärtig. Der Mensch kann ein Zeitintervall in eine große Menge Situationen einteilen und tut dies intuitiv, ohne darüber nachdenken zu müssen, wie er eine Situation von einer anderen unterscheidet. In den meisten Situationen handelt ein Mensch zudem vollkommen intuitiv, ohne, dass sein Oberbewusstsein einen starken Einfluss darauf nimmt. Mit zunehmender Komplexität der Situation, durch eine Vielzahl von Informationen über die Situation, die aus den unterschiedlichsten Quellen stammen, gelangt der menschliche Verstand jedoch bald an seine Grenzen. In einer solchen Situation ist damit zu rechnen, dass immer mehr Informationen ignoriert werden, da keine Kapazitäten bereitstehen diese zu verarbeiten. Dies führt dazu, dass immer mehr spontan gehandelt wird, was schlussendlich zu Fehlern führen kann.
Beispiele für solche komplexen Situationen sind digitale Märkte. Hier sind vor allem die Krisensituationen wie zuletzt 2008 [Pip10] von Interesse. Bei einer Krise in einem solchen System ist es für einen menschlichen Verstand praktisch unmöglich, die Informationen aus allen bereitstehenden Quellen zu verarbeiten. Auf Grund dieser mangelnden Verarbeitungskapazitäten entstehen Fehleinschätzungen. Hier können Systeme, die die Informationen in der Geschwindigkeit verarbeiten können in der sie entstehen, Abhilfe schaffen.
Des Weiteren können durch Umweltkatastrophen extrem kritische Situationen entstehen. Durch Erdbeben und Überflutungen sind häufig große Regionen und somit viele Menschenleben in Gefahr. Dies ist eine enormere Belastung für Rettungskräfte direkt vor Ort, sowie für organisatorische Einheiten, da Entscheidungen in kürzester Zeit getroffen werden müssen, ohne alle Informationen berücksichtigen zu können. Eben jene Belastung führt dazu das Fehler gemacht werden, durch die im schlimmsten Fall noch mehr Menschenleben gefährdet werden. Es ist also nötig zusätzliche Systeme zu entwickeln, die von diesem Druck nicht betroffen sein können.
Folgerichtig müssen solche Umweltkatastrophen nicht immer natürlichen Ursprungs sein. Ebenso können durch den Menschen Umweltkatastrophen, zum Beispiel Reaktorunglücke verursacht werden. Solche Katastrophen hätten in der Vergangenheit

teilweise vermieden werden können, wenn Warnhinweise bemerkt, bzw. beachtet worden wären. Um zu vermeiden das solche Fehler passieren, sind Systeme nötig, die dem Nutzer klar und leicht verständlich aufzeigen, welche Gefahren aktuell vorliegen.

1.2. Ziel

In dieser Arbeit wird auf die beiden etablierten Ansätze des Situationen Kalküls und der Situationen Theorie, so wie auf Künstliche Neuronale Netze eingegangen. Diese drei Bereiche sollen in ihren Grundzügen beschrieben werden. Dabei wird auch auf die Vor- und Nachteile im Bezug auf die Situationsbewertung eingegangen.
Die etablierten Ansätze sollen im Weiteren als Grundlage für ein Framework genutzt werden, mit dem die Situationsbewertung möglichst abstrakt beschrieben werden kann. Das Framework soll also möglichst unabhängig vom späteren Einsatzbereich sein. Zudem werde ich versuchen, das Framework so zu entwerfen, dass es möglichst einfach auf Maschinen implementiert werden kann und eine geringe Laufzeit aufweist. Durch die abstrakte Beschreibung des Frameworks soll es ermöglicht werden, dass dieses in möglichst vielen Bereichen eingesetzt werden kann und dennoch der Vorgang der Situationsbewertung vereinheitlicht wird.
Dabei ist es nicht die Idee hinter dem Framework eine künstliche Intelligenz aufzubauen die eigenständig Entscheidungen treffen kann. Es sollen lediglich Empfehlungen für einen Nutzer generiert werden, wenngleich der Nutzer des Frameworks theoretisch eine weitere Maschine sein könnte. Gleichfalls wird in dieser Arbeit auf keine konkrete Implementierung eingegangen, sondern lediglich auf die zugrunde liegende Logik. Somit sollte eine Implementierung in einer beliebigen Sprache relativ schnell möglich sein.
Das Ergebnis des Frameworks soll eine übersichtliche Menge von Indikatoren sein, an denen ein späterer Nutzer erkennen kann, ob die bewertete Situation zum Beispiel sicher ist oder ob davon auszugehen ist, dass die Situation sich ohne Einwirken des Nutzers negativ verändern wird. Es soll also eine Reduktion von theoretisch unbegrenzt vielen Informationen auf einige wenige stattfinden, ohne das relevante Informationen verloren gehen.

1.3. Struktur

Im anschließenden Kapitel wird auf das Situationen Kalkül, die Situationen Theorie, sowie Künstliche Neuronale Netze eingegangen. Hierbei werden die drei Themenbereiche kurz beschrieben und ich werde aufzeigen, welche Vor- und Nachteile sich aus den jeweiligen Ansätzen in Bezug auf die Situationsbewertung ergeben.
Im 3. Kapitel werde ich das Framework zur Situationsbewertung aufbauen. Dabei werden in Kapitel 3.1 die grundlegenden Begriffe aus Kapitel 2.1 noch einmal in der Form beschrieben, wie sie für das Framework verwendet werden. Darauf folgt in Kapitel 3.2 die Struktur des Frameworks und in Kapitel 3.3 die Bewertungsfunktion, welche verwendet wird um die Situation zu bewerten. Zum Abschluss von

Kapitel 3 werde ich noch ein paar mögliche Erweiterungen des Frameworks aufzeigen, die aufgrund des Umfangs dieser Arbeit nicht mehr umgesetzt werden konnten. Das 4. Kapitel umfasst ein kleines Anwendungsbeispiel, in dem drei Situationen mit Hilfe des Frameworks bewertet werden.
Zum Schluss findet im 5. Kapitel eine Diskussion der Ergebnisse dieser Arbeit statt, in der ich auf mögliche Einsatzgebiete, sowie Vorteile und Grenzen des Frameworks eingehen werde.

2. Abgrenzung und theoretische Grundlage

In diesem Kapitel wird auf grundlegende Begriffe wie Wirklichkeit, Situation und Universum eingegangen. Des Weiteren werden die beiden theoretischen Ansätze Situationen Kalkül *(engl. situation calculus)* und Situationen Theorie *(engl. situation theorie)* zum Umgang mit Situationen beschrieben. Zusätzlich wird auf sogenannte Künstliche Neuronale Netze (KNN) eingegangen, da diese ebenfalls für die Situationsbewertung genutzt werden können. Die Künstlichen Neuralen Netze stellen zwar genau genommen keinen neuen Ansatz dar, sind aber gerade in den letzten Jahren in vielen Anwendungsbereichen wieder interessant geworden [ZGL16][NYC15], da mittlerweile die benötigten Rechenleistungen für komplexe Anwendungsfälle teilweise bereitgestellt werden können.
Da es sich beim Situationen Kalkül und der Situationen Theorie um mathematische Formalismen handelt, werden diese auch formal beschrieben, wohingegen auf die KNN aus einem anwendungsorientierten Blickwinkel eingegangen.

2.1. Abgrenzung der Situation

In diesem Abschnitt wird eine grobe Abgrenzung der Begriffe Wirklichkeit, Situation und Universum erfolgen. Diese Begriffe sind für die vorliegende Arbeit von essentieller Bedeutung und werden in späteren Kapiteln mit den dann zur Verfügung stehenden Werkzeugen präziser abgegrenzt.

2.1.1. Wirklichkeit

Den Begriff der Wirklichkeit werde ich einteilen in die *echte* Wirklichkeit, also die Welt, in der wir Menschen uns befinden und die *fiktiven* Wirklichkeiten, wie sie zum Beispiel in einem Computer erschaffen werden. Der wichtigste Unterschied dieser beiden Arten von Wirklichkeit ist, inwieweit es dem Betrachter möglich ist, die Welt zu hinterfragen und auf einer abstrakten Ebene zu betrachten.
Die *echte* Wirklichkeit ist einzigartig und hat eine so große Menge an möglichen Ausprägungen, dass ich sie im Folgenden als unendlich annehmen werde. Auch die Möglichkeiten eines Anwenders in dieser Welt zu agieren oder reagieren werden im Folgenden als unendlich angenommen. Des Weiteren nehme ich an, dass diese Wirklichkeit kein Ende hat, selbst das Ausscheiden eines Beobachters oder Anwenders beendet diese Wirklichkeit nicht.
Anders als bei der echten Wirklichkeit, kann es beliebig viele *fiktive* Wirklichkeiten

geben. Sie können von einem Anwender erschaffen und somit auch beendet werden. Diese Eigenschaft werde ich im Folgenden als Gebrechlichkeit bezeichnen. Zudem ist es möglich eine fiktive Wirklichkeit in ihren Ausprägungen einzugrenzen. Diese Eingrenzung kann durch den Anwender selbst vorgenommen werden, da er die Gesetzmäßigkeiten, die einer fiktiven Wirklichkeit zu Grunde liegen, wählen kann. Eine fiktive Wirklichkeit ist zum Beispiel ein Computer-Programm solange es ausgeführt wird. Innerhalb des Programmes sind alle möglichen Ausprägungen durch den Quellcode eingegrenzt und sobald das Programm terminiert ist die Wirklichkeit beendet.
Ein Beobachter kann die echte Wirklichkeit nur subjektiv betrachten, da er immer Teil dieser Wirklichkeit ist. Wohingegen er eine fiktive Wirklichkeit objektiv betrachten kann, wenn er nicht Teil dieser fiktiven Wirklichkeit ist. Ist der Beobachter allerdings ein Teil dieser Wirklichkeit, dann kann er auch diese einzig subjektiv betrachten.

2.1.2. Situation

Eine Situation ist eine Ausprägung einer echten oder fiktiven Wirklichkeit. Somit kann eine Situation durch eine Menge von Parametern beschrieben werden.
Die Beschreibung durch Parameter ist nur bei fiktiven Wirklichkeiten vollständig möglich, muss jedoch nicht zwingend vollständig sein. Im Allgemeinen ist davon auszugehen, dass eine Reduktion auf eine teilweise Beschreibung deutlich sinnvoller ist. Hierbei können auch abgeleitete Parameter eingesetzt werden, die nicht in der Situation direkt gemessen werden können, sondern aus anderen Parametern geschlussfolgert werden, also indirekt gemessen werden.
Die echte Wirklichkeit kann immer nur teilweise erfasst werden. Deshalb können die Situationen der echten und der fiktiven Wirklichkeiten als der Art nach gleich behandelt werden.
In jeder Situation gibt es zusätzlich Aktionen, die ein Anwender ausführen kann, um eine Situation zu verändern und somit eine neue Situation zu erreichen. Zusätzlich gibt es Events, die ohne Einfluss eines Anwenders ablaufen und die Situation verändern und somit zu einer neuen Situation führen. Wenn man sich in einer fiktiven Wirklichkeit befindet und keine Aktionen sowie Events mehr möglich sind hat man eine abschließende Situation erreicht und die Wirklichkeit zerbricht.

2.1.3. Universum

Im Folgenden werde ich alle Situationen, Aktionen, Events, Anwender, etc. in einem Universum zusammenfassen. Hierdurch kann abgrenzen werden, was Teil der Betrachtung ist und was nicht.
Zum einen wäre es möglich, alle Elemente eines Universums explizit zu beschreiben, zum anderen wäre es möglich das Universum auf abstrakter Ebene zu entwerfen und somit auch die unendlichen Mengen durch endliche Anzahlen von Eigenschaften zu

beschreiben.
Da eine explizite Beschreibung aller Situationen, Aktionen, Events, Anwender, etc. sogar bei sehr starken Eingrenzungen extrem aufwendig ist, werde ich diesen Ansatz hier nicht verfolgen.
Stattdessen werde ich versuchen das Universum möglichst abstrakt zu beschreiben. Deshalb wird nicht die Menge der Situationen, die in der betrachteten Wirklichkeit möglich sind, in das Universum aufgenommen, sondern nur die Parameter mit denen wir die Situationen beschreiben wollen. Des Weiteren gehe ich davon aus, dass alle Aktionen und Events möglich sind, die Parameter in einer beliebigen Stärke verändern, so lange die Parameter Teil des Universums sind. Als Anwender gelten solche Elemente, die in der Lage sind Aktionen auszuführen.

2.2. Situationen Kalkül

Das Situationen Kalkül, welches von McCarthy 1963 in einer Arbeit [McC63] vorgestellt und über die Jahre stetig weiter entwickelt wurde [MH69][McC02], gilt als einer der Formalismen, um Situationen und die Dynamik zwischen Situationen zu beschreiben. Gerade, weil es sich um die Grundlage vieler Ansätze für den Umgang mit Situationen handelt, wird es auch für die Bewertung von Situationen aufgegriffen. Allerdings ist die Idee hinter dem Situationen Kalkül weniger die Beschreibung von Situationen, als die Beschreibung der Dynamik zwischen Situationen. In der „block world“ [McC02] geht es zum Beispiel darum, eine Welt mittels des Situationen Kalküls zu beschreiben, in der es nur gleich große Blöcke gibt und einen Roboter, der nur Blöcke bewegen kann, die frei sind. Die Komplexität der beschriebenen Welt ist jedoch relativ gering und nicht mit alltäglichen Situationen zu vergleichen. Zusätzlich tritt in dieser, durch das Situatuionen Kalkül beschriebenen Welt, das sogenannte Rahmenproblem *(engl. frame problem)* [Rei91] auf. Dieses beschreibt die mangelnden Möglichkeiten mit Hilfe des Situationen Kalküls eine vollständige Menge an Situationen und Aktionen zu beschreiben, sodass es durch eine beliebige mögliche Verkettung von Aktionen immer zu einer in der Menge enthaltenen Situation führt.

2.2.1. Struktur

Im Folgenden wird die Grundstruktur des von McCarthy vorgestellten Situationen Kalküls [McC02] beschrieben. Dabei werden geeignete Einschränkungen, Erweiterungen und Veränderungen eingeführt, durch die das Situationen Kalkül für die Situationsbewertung besser verwendet werden kann.

Situation

Eine Situation s beschreibt einen kompletten Zustand des gesamten betrachteten Universums zu einem bestimmten Zeitpunkt [MH69]. Bei Reiter wird eine Situation als bestimmte Reihenfolge von allen Aktionen, die in der Reihenfolge zu der zu

beschreibenden Situation geführt haben [Rei01]. Da die vollständige Beschreibung der Wirklichkeit weder möglich noch sinnvoll ist, wird das betrachtete Universum auf die wenigen Parameter eingegrenzt, die beobachtet werden können und sollen. Es wird also ein Universum verwendet, wie es im vorherigen Kapitel beschrieben wurde.
Die Menge aller Situationen nennen wir $\mathcal{S}$ und die einzelnen Situationen sind mit s_i für $0 \leq i \leq |\mathcal{S}|$ eindeutig bestimmbar. Bei der Definition von Reiter kann die Ursprungssituation nicht frei gewählt werden, da es möglich sein muss, alle Situationen von der Ursprungssituation zu erreichen. Ansonsten ist ein Beschreiben der jeweiligen Situation nicht mehr möglich. Durch die Definition von McCarthy gilt diese Eingrenzung nicht und die Ursprungssituation $S0 = s_0$ kann beliebig gewählt werden. Diese Situation hilft lediglich anzuzeigen, ab wann das Universum beobachtet wurde. Hierdurch kann es allerdings passieren, dass Situationen aus $\mathcal{S}$ von einer Ursprungssituation unerreichbar sind.

Fluents

Die Fluents sind die Grundlage um Situationen zu beschreiben und können entweder aussagenlogisch oder funktional sein. Mit diesen beiden Arten von Fluents kann eine einzelne Situation im betrachteten Universum vollständig beschrieben werden.
Sei $\mathcal{S}$ die Menge aller Situationen, die in dem betrachteten Universum möglich sind, und

$$pfluent : \mathcal{S} \rightarrow \{true, false\}$$

eine Abbildung. Dann ist $pfluent$ ein aussagenlogischer Fluent. Dieser Fluent bildet jede Situation s auf $true$ oder $false$ ab, je nachdem ob s die durch $pfluent$ beschriebene Bedingung erfüllt oder nicht.
Sei $\mathbb{M}$ eine beliebige Menge von Zahlen und

$$tfluent : \mathcal{S} \rightarrow \mathbb{M}$$

eine Abbildung. Dann ist $tfluent$ ein funktionaler Fluent. Im Gegensatz zum aussagenlogischen Fluent geben funktionale eine Ausprägung an. Mit einem funktionalen Fluent kann zum Beispiel die Temperatur einer Situation an einem Ort angegeben werden.
Anders als von McCarthy ursprünglich postuliert werde ich keine zusätzliche Abbildung $Holds$ verwenden um anzugeben, welchen Wert ein Fluent für eine Situation hat, stattdessen werde ich verkürzend einfach $fluent(s) = x$ schreiben. Wobei $fluent$ definiert sein muss für eine Menge Situationen $\mathcal{S}$, $s \in \mathcal{S}$ und x ein Element des Bildes der Abbildung $fluent$ sein muss.

Aktionen

Ich werde im Folgenden nicht im Detail darauf eingehen, wie ein Element a beschaffen sein muss, damit es eine Aktion darstellt, sondern werde annehmen, dass es Objekte gibt, die als Aktion in einer Situation ausgeführt werden können. Da der

Schwerpunkt dieser Arbeit lediglich auf der Bewertung von Situationen und nicht auf dem Berechnen von Ablaufplänen liegt, fällt die Auswirkung dieser Einschränkung gering aus.
Sei $\mathcal{A}$ eine Menge von Aktion und

$$result : \mathcal{A} \times \mathcal{S} \rightarrow \mathcal{S}$$

eine Abbildung, dann gilt für alle $s, s' \in \mathcal{S}$ und $a \in \mathcal{A}$ für die gilt $result(a, s) = s'$ das s' aus s resultiert, wenn die Aktion a in der Situation s ausgeführt wird.
Die Abbildung

$$next : \mathcal{S} \rightarrow \mathcal{S}$$

mit $next(s) = s'$ für $s, s' \in \mathcal{S}$ definiert, dass auf die Situation s die Situation s' folgt.

2.2.2. Dynamik

Dynamik bezeichnet den Wechsel von einer Startsituation s_o zu einer Ziel-Situation s_i. Hierbei kann $s_0 = S0$ gewählt werden und somit immer mit der Ursituation gestartet werden. Zwischen s_0 und s_i können beliebig viele Situationen liegen, die durch Aktionen erreicht wurden. Auf diese Dynamik wird im Folgenden allerdings nur kurz eingegangen, da sie für die grundlegende Situationsbewertung nicht von Relevanz ist. Lediglich für komplexere Situationsbewertungsverfahren, die auch im Detail auf die Historie, also die vorherigen Situationen eingehen, ist die Betrachtung der Dynamik wichtig.
Grundsätzlich kann in jeder Situation immer eine Aktion ausgeführt werden, ansonsten wäre man an einer Situation angekommen, die ein absolutes Ende darstellt. Dies ist nicht sinnvoll, wenn man versucht Situationen aus der echten Wirklichkeit abzubilden. Auch wenn man argumentieren könnte, dass in jeder Situation nur genau eine Aktion aufgrund des deterministischen Verhaltens der echten Wirklichkeit möglich ist, werde ich davon ausgehen, dass in einer Situation auch durchaus verschiedene Aktionen möglich sind. Dies liegt daran, dass unser Universum nur einen Ausschnitt der Wirklichkeit zeigt und man wegen der fehlenden Informationen nicht mehr exakt sagen kann, welches die eine Aktion ist, die möglich ist. Bei fiktiven Wirklichkeiten gilt, dass grundsätzlich verschiedene Aktionen möglich sind. Dies ist abhängig von der Definition der fiktiven Wirklichkeit.
Daher gehen ich grundsätzlich bei allen Wirklichkeiten davon aus, dass in einer Situation eine beliebige Menge von Aktionen möglich ist. Jede Aktion hat dabei die Möglichkeit bestimmte Ausprägungen in einem bestimmten Ausmaß zu verändern. Dabei muss eine Aktion nicht zwingend von einem Akteur ausgeführt werden, sondern kann auch *zufällig* ablaufen. Immer wenn ein Akteur keinen Einfluss auf eine Aktion hat handelt es sich um ein Event.
Immer wenn eine Aktion oder ein Event in einer Situation stattfindet, gelangt man zu einer anderen Situation. Die Veränderungen der Situationen werden durch die Fluents dargestellt. In der erreichten Situation können die Fluents andere Ergebnisse haben als vorher. Allerdings kann auch nicht ausgeschlossen werden, dass die

Fluents gleich bleiben. Dies kann dadurch begründet werden, dass sich Eigenschaften der Situation verändert haben, die durch das verwendete Universum nicht abgebildet werden können.

2.2.3. Erweiterungen

Im Folgenden werden ein paar Erweiterungen des Situationen Kalküls, sowie darauf basierende Systeme, beschrieben. Es gibt unzählige Erweiterungen und Vorschläge zur Anpassung des Kalküls, sodass hier nicht auf alle eingegangen werden kann, sondern nur auf eine kleine Auswahl.

Reiters Lösung zum Rahmenproblem Eine der wichtigsten Erweiterungen ist die von Reiter vorgestellte Lösung des Rahmenproblems [Rei91]. Diese Erweiterung wird in den meisten späteren Arbeiten als selbstverständlich angesehen und verwendet, da sie das Rahmenproblem des Situationen Kalküls immer dann löst, wenn die Auswirkungen aller Aktionen bekannt sind. Für den Umgang mit Alltagssituationen ist dies im Allgemeinen jedoch nicht ausreichend, da die meisten Auswirkungen eher wage Vermutungen oder völlig unbekannt sind.

Zeit im Situationen Kalkül Allen stellt in *Twords a General Theory of Action and Time* [All84] ein Framework vor, um mit Aktionen umgehen zu können, die einen zeitlichen Bezug haben. Dabei beschränkt sich Allen nicht auf das Situationen Kalkül. Dies ist ein großer Vorteil verglichen mit dem Situationen Kalkül, in dem nur Aktionen abgebildet werden können, die keinen zeitlichen Bezug haben.
Um diese Möglichkeiten auch im Situationen Kalkül bereitzustellen haben Pinto und Reiter eine zusätzliche Erweiterung vorgeschlagen, die genau auf die Kernfunktionen von Allens Arbeit eingeht [PR95].

Golog Golog ist eine logische Programmiersprache, welche von Levesque et al [LRL+97] im Jahr 1997 vorgestellt wurde. Die Sprache wurde in Anlehnung an ALGOL entwickelt. Beim Entwurf der Sprache wurde das Situationen Kalkül, mit einigen Erweiterungen, als Grundlage verwendet. Anweisungen in Golog werden in Aussagen des Situationen Kalküls übersetzt, welche ausschließlich aus primitiven Aktionen und Fluents bestehen, die vom Entwickler definiert wurden. Diese Übersetzungen können dann direkt mit dem Situationen Kalkül ausgewertet bzw. überprüft werden. Der Vorteil dieses Verfahrens ist, dass es keine feste Menge von primitiven Anweisungen gibt, sondern diese können vom Entwickler beliebig entworfen werden. Allerdings muss für die Auswertung die initiale Situation vollständig beschrieben sein, was für viele Anwendungsbereiche ein großes Hindernis darstellt.
Ein interessantes Anwendungsbeispiel haben Ferrein et al in ihrer Arbeit über Roboter-Fußball beschrieben [FFL05]. In dieser Arbeit konnte gezeigt werden, dass Golog auch für zeitkritische Anwendungen, wie das Steuern von Robotern in einem Fußballspiel, möglich ist.

In einer 2000 veröffentlichten Arbeit haben Giacomo et al mit conGolog eine Erweiterung von Golog vorgestellt [DLL00]. Grundlage dieser Erweiterung ist die gleichzeitige *(engl. concurrent)* Ausführung verschiedener Aufgaben, bzw. die Möglichkeit eine Aufgabe zu unterbrechen. Zudem können Berechnungen durchgeführt werden, ohne dass die Startsituation vollständig bekannt ist. Ebenso ist der reaktive Ansatz von conGolog für viele Anwendungsbereiche interessant. Es wird verstärkt davon ausgegangen, dass auf Veränderungen der Umgebung eingegangen werden sollte und sogar muss.

2.2.4. Zusammenfassung

Zum Abschluss werden kurz die Vor- und Nachteile des Situationen Kalküls bzgl. der Situationsbewertung beschrieben und es wird darauf eingegangen, in wie weit das Situationen Kalkül genutzt werden kann, um Situationen zu bewerten.

Vorteile

Da es sich bei dem Situationen Kalkül um einen mathematischen Formalismus handelt, ist es extrem gut auf Rechnern umsetzbar. Ein System, welches das Situationen Kalkül als Grundlage nutzt, kann somit relativ schnell als Anwendung implementiert werden, da teilweise keine Veränderungen nötig sind.
Zudem handelt es sich bei dem Situationen Kalkül um eine in der Informatik relativ alte Technik, die dementsprechend viel diskutiert wurde und für die viele Erweiterungen vorgeschlagen wurden. Durch diesen Reifegrad ist davon auszugehen, dass das Situationen Kalkül eine stabile Basis bildet. Anders als bei jungen Techniken, die starken Veränderungen und Anpassungen unterworfen sind, weil Probleme auffallen, die anfangs nicht bedacht wurden.

Nachteile

Der große Vorteil der strikten mathematischen Formulierungen kann bei der Bewertung von Situationen im alltäglichen Leben auch große Nachteile bringen, da sich diese Situationen meistens nicht in dem Maße beschreiben lassen, wie es der Formalismus verlangt. Hier müsste das System erweitert werden um auch mit ungenauen, unglaubwürdigen und fehlenden Informationen arbeiten zu können und dem Menschen, der mit dem System interagiert, muss immer die nötige Freiheit gegeben werden, auf die Entscheidungen des Systems eingreifen zu können. Bei der Situationsbewertung ist nicht davon auszugehen, dass sich Situationen durch ein gewähltes Universum vollständig beschreiben lassen, wie es bei den Roboter-Steuerungssystemen der Fall ist. Für Roboter-Steuerungssysteme ist es völlig ausreichend nur erlaubte Situationen und Aktionen zu beschreiben, da der Roboter über keinen eigenen Willen verfügt und er im Falle einer unerwarteten Situation in einen fehlerhaften Zustand übergeht, der dann durch das eingreifen eines Menschen behoben wird. Im Alltag ist ein solcher Fehlerzustand keine Option, da alleine das erkennen eines solchen Zustandes teilweise von persönlichen Empfindungen abhängt.

Ein weiteres viel diskutiertes Problem des Situationen Kalküls ist das Problem der Auswirkungen (*ramification*) [Lin08]. Hier geht es darum, dass aufgrund der Struktur des Situationen Kalküls beim Auswerten der prädikatenlogisch aufgebauten Situationen gewisse Informationen über die Situation verloren gehen können. Dies muss in einer Implementierung dringend verhindert werden durch zum Beispiel globale Attribute einer Situation, die zwar durch Aktionen und zufällige Events verändert werden können, aber nicht bei jeder Situation neu ermittelt werden.
Zum Schluss muss noch eine geeignete Lösung für das sogenannte Rahmenproblem (*frame problem*) [Lin08] gewählt werden. Dies gehört, wie das Problem der Auswirkungen, zu einem der ersten Probleme, die für das Situationen Kalkül diskutiert wurden, und geht auf die Komplexität ein, ein Universum an Situationen und dazugehörigen Aktionen bzw. zufälligen Events zu wählen, sodass man keine Situation erreichen kann, die nicht Teil des Universums ist. Um Situationen des Alltags zu beschreiben bietet es sich an, mit dem Universum keine direkten Situationen zu beschreiben, sondern lediglich Parameter, wobei sich Situationen dann wieder aus diesen Parametern mit speziellen Ausprägungen zusammensetzen können.

Auf Grund der hier genannten Vor- und Nachteile ist das Situationen Kalkül nur teilweise anwendbar für die Situationsbewertung. Es bietet einen stabilen Formalismus um Situationen zu beschreiben, an dem man sich für die Implementierung orientieren kann. Da das Situationen Kalkül in seinem Grundkonzept allerdings für abgeschlossene Welten und vor allem, die in diesen Welten möglichen Handlungen gedacht ist, bietet es keine vollständige Grundlage für die Situationsbewertung.

2.3. Situationen Theorie

Die Situationen Theorie von Barwise und Perry ist ein weiterer mathematischer Ansatz zur Beschreibung von Situationen [BP81]. Anders als beim Situationen Kalkül stehen hier jedoch die Situationen als solches mehr im Vordergrund. In der Situationen Theorie wird zwischen mehreren Arten von Situationen unterschieden, es gibt somit nicht den einen Typ von Situationen. Hierauf wird im Späteren noch genauer eingegangen.
In der Situationen Theorie wird jedoch keine Dynamik explizit abgebildet. Es gibt keine explizite Möglichkeit Aktionen oder Events zu definieren. Stattdessen wird davon ausgegangen, dass sämtliche Aktionen und Events bekannt sind und es wird nur an gegeben, ob eine Aktion oder ein Event in einer Situation stattfindet. Die Situationen Theorie befindet sich daher auf der konzeptionellen Ebene im Bezug auf Situationen.

2.3.1. Struktur

Im Folgenden wird die Struktur von Devlin [Dev91] verwendet, welche auf den Arbeiten von Barwise und Perry zur Situation Theorie aufbaut.

In der Situationen Theorie unterscheidet man zwischen mehreren Arten von Situationen, die durch sogenannte Infons beschrieben werden. Die Infons übernehmen die Aufgabe der Fluents im Situationen Kalkül. Um Eingrenzungen vorzunehmen werden Constraints verwendet, welche ebenfalls in bestimmte Gruppen eingeteilt werden.

Situation

Alle Situationen werden in sechs verschiedene feste Gruppen eingeteilt. Allerdings kann eine Situation zu verschiedenen Zeitpunkten durchaus anderen Gruppen angehören.
Die Grundlage stellen *reale Situationen*, hierbei handelt es sich um Situationen aus der echten Wirklichkeit, welche auf Grund der Informationsmenge nicht beschrieben werden können. Somit sind reale Situationen nur theoretisch verwendbar. Alle anderen Situationenarten können im Gegensatz zu den realen Situationen beschrieben werden, da sie eine Reduktion einer realen Situation darstellen. Nur reale Situationen können eindeutig identifiziert werden. Bei allen anderen Situationen ist dies durch die Reduktion im Allgemeinen nicht mehr möglich, da es sein kann, dass die Eigenschaften verloren gegangen sind, welche die Situation einzigartig machten.
Eine *aktuelle Situation* ist eine Situation, in der sich der Beobachter selbst befindet. Bei einer aktuellen Situation liegt der Betrachtungsschwerpunkt direkt auf der Situation selbst, es gibt also keine Bezüge auf andere Situationen. Wenn eine Menge aktueller Situationen weiter reduziert wird, bzw. auf eine abstrakte Ebene angehoben wird, spricht man von *abstrakten Situationen*. Eine abstrakte Situation ist also eine Verallgemeinerung von aktuellen Situationen.
Aussagesituationen (*engl. utterance situations*) liegen immer dann vor, wenn in einer Situation über eine andere Situation gesprochen wird. Der Fokus liegt hier also nicht auf der Situation in der sich ein Beobachter befindet, sondern auf der Situation, über die gesprochen wird. Die Situation über die in einer Aussagesituation gesprochen wird nennt man *fokale Situation*. In eine fokale Situation ist der Betrachter aus der Aussagesituation also nicht zwingend involviert.
Die letzte Gruppe der Situationen ist die der *Ressourcensituationen*. Alle Situationen auf die sich bezogen wird können dieser Gruppe zugeordnet werden. Anders als bei den fokalen Situationen muss der Fokus allerdings nicht zwingend auf der Ressourcensituation liegen, es reicht völlig aus, wenn die Situation erwähnt wird um zum Beispiel Zusammenhänge zu erklären.

Infons

Die Eigenschaften von Situationen werden in der Situationen Theorie mit Infons dargestellt. Die Infons können eine Situation auf sehr abstrakter Ebene beschreiben und werden als Tupel der folgenden Form definiert

$$<< R, a_1, ..., a_n, 0/1 >> .$$

Wobei R eine Relation ist, die als primitiv und somit als bekannt angenommen wird. Die Elemente $a_1, ..., a_n$ sind die Argumente der Relation. Die letzte Stelle eines Infos ist die Polarität, also 0 wenn das Infon für die Situation nicht gilt und 1, wenn das Infon für die Situation gilt.
Infos sind vergleichbar mit Fluents aus dem Situationen Kalkül, sie übernehmen die gleiche Aufgabe, sind in ihrer Struktur allerdings anders. Infons bieten mehr ein Konstrukt zum Modellieren von Informationen über eine Situation, weniger eine Möglichkeit zur mathematischen Berechnung.

Typen und parametrisierte Infons

Eine Erweiterung des einfachen Infons bieten die parametrisierten Infons an. Es werden Infons um Elemente aus festen Gruppen erweitert. Dies wird verwendet um unabhängig von der Relation des Infos bestimmte Elemente in den Infon aufzunehmen. Die folgenden Typen können verwendet werden um Infons zu parametrisieren.

- TIM : temporaler Typ
- LOC : räumlicher Typ
- IND : Typ eines Individuums
- REL^n : Typ einer n-stelligen Relation
- SIT : Typ einer Situation
- INF : Typ eines Infons
- TYP : Typ eines Typs
- PAR : Typ eines Parameters
- POL : Typ einer Polarität

Angenommen, es liegt eine Situation vor, die durch

$$<< sieht, Claudia, e, Mensa, heute, 1 >>$$

beschrieben werden kann, wobei e eine Situation ist, die durch

$$<< liest, Arne, Mensa, heute, 1 >>$$

beschrieben werden kann. Dann sind *sieht* und *liest* vom Typ REL^2 bzw. REL^1, *Claudia* und *Arne* sind vom Typ IND, e ist vom Typ SIT, *Mensa* ist vom Typ LOC, *heute* ist vom Typ TIM und *1* ist vom Typ POL.

Constraints

Die Constraints in der Situationen Theorie werden in drei Klassen eingeteilt. Diese Klassen werden im Folgenden kurz beschrieben. Auf eine genaue Definition der Constraints wird hier verzichtet.
Die erste Klasse bilden die *nomic Constraints* durch die Naturgesetze abgebildet werden. Nomic Constraints gelten somit in jeder Situation und können nicht umgangen werden. Hierbei wird ein intuitives Verständnis der Naturgesetze angenommen. Durch die nomic Constraints werden auch die meisten *reflexiven Constraints* bestimmt. Bei reflexiven Constraints handelt es sich um meistens sehr abstrakte Constraints, die auf eine Vielzahl von Constraints zurückgreifen und diese zusammenfassen. Ein anschauliches Beispiel für reflexive Constraints bietet der Regen. Wenn es regnet kann daraus gefolgert werden, dass der Boden draußen nass wird bzw. nasser wird. Würde man dies nicht abstrakt ausdrücken, müsste man auf eine Vielzahl von Constraints zurückgreifen. Unter anderem, dass sich Regentropfen, auf Grund der Gravitation, von den Wolken zum Erdboden bewegen, dass das Auftreffen von Wasser auf einer Oberfläche dazu führt, dass diese als nass bzw. nasser gesehen wird und sogar, dass Regentropfen im Allgemeinen als Wassertropfen aufgefasst werden. Auch auf die für den Regen nötigen Wolken wurde nicht eingegangen, es wurde angenommen, dass es Wolken geben muss, weil es regnet.
Die letzte Klasse bilden die *konventionellen Constraints*, bei denen es sich um moralische oder soziale Normen handelt. Viele konventionelle Constraints können auch als reflexive Constraints aufgefasst werden, allerdings basiert die Schlussfolgerung bei den konventionellen Constraints auf gesellschaftlichen Erwartungen und nicht auf Naturgesetzen. Somit ist es möglich, dass konventionelle Constraints nicht überall gleich sind, wohin gegen die nomic Constraints an jedem Ort zu jeder Zeit ihre Gültigkeit behalten.

2.3.2. Zusammenfassung

Zum Abschluss werden kurz die Vor- und Nachteile der Situationen Theorie bzgl. der Situationsbewertung beschrieben und es wird darauf eingegangen, in wie weit die Situationen Theorie genutzt werden kann um Situationen zu bewerten.

Vorteile

Die Situationen Theorie kann als eine Art Datenstruktur für Situationen verwendet werden, da der Schwerpunkt auf der Beschreibung von Situationen liegt und nicht auf der Dynamik zwischen Situationen. Damit würde sich die Situationen Theorie vor allem für einfache Situationsbewertungen eignen, die nicht auf die Historie einer Situation eingehen. Zudem sind die Möglichkeiten, die durch die Infons gegeben sind, sehr weitreichend. Hier werden wenige Eingrenzungen durch die Situationen Theorie gemacht, sodass die Modellierungsmöglichkeiten umfassend sind.
Des Weiteren bietet die Situationen Theorie eine Gliederung für Situationen und

Constraints an, wodurch diese auf einer abstrakten Ebene zu Mengen zusammengefasst werden können.

Nachteile

Gerade durch die Gliederung von Situationen und Constraints kann es allerdings zu Problemen kommen. Für die Situationsbewertung ist es wohl möglich nicht von Interesse die Situationen zu gliedern, da alle Situationen unabhängig von ihrer Einteilung bewertet werden. Zudem ist die Einteilung der Constraints gerade im Zusammenhang mit verschiedenen Domänen nicht ausreichend. Deshalb müsste diese voraussichtlich um einige Gruppen erweitert werden.
Wenn Situationen aus völlig verschiedenen Bereichen bewertet werden sollen, sollte die Grundlage der Bewertung möglichst abstrakt sein, was durch die Gliederungen von Situationen und Constraints, sowie die Parametrisierung der Infons verhindert werden könnte.
Für komplexe Situationsbewertungen müssten zusätzlich noch Aktionen und Events als eigene Elemente eingebaut werden, damit auch auf die Abläufe, die zu einer bestimmten Situation geführt haben, eingegangen werden kann. Dies wäre ansonsten nur mit einer extrem großen Menge von Infons möglich, welche den Umgang mit Situationen unnötig komplex machen würden.

2.4. Künstliche Neuronale Netze

Der Bereich der Künstlichen Neuronalen Netze (KNN) ist extrem weit und wird von verschiedenen Wissenschaftlern teilweise sehr unterschiedlich eingeteilt. Eine, von den meisten anerkannte Gliederung der KNN, ist daher leider nur auf sehr abstrakter Ebene möglich. Zum Beispiel lassen sich die meisten KNN in überwacht lernende und unüberwacht lernende Netze einteilen. In dieser Arbeit wird aus jedem dieser zwei Bereiche je ein Vertreter vorgestellt und es wird gezeigt, wie man diese Netze für die Situationsbewertung einsetzt kann. Dabei sollen die KNN nicht verwendet werden um die gesamte Situation zu bewerten, sondern lediglich um Situationen aus der Perspektive einer Domäne zu bewerten. Dies macht nur Sinn, wenn zu der Domäne keine klaren Regeln zur Bewertung existieren und es auf Grund des Aufwandes nicht lukrativ erscheint für die Domäne händisch Regeln zu entwerfen.
Die Idee der neuronalen Informationsverarbeitung existiert schon relativ lange. Das Dartmouth Summer Research Project von 1956 wird häufig als die Geburtsstunde der „Künstlichen Intelligenz“ gesehen [Kra93]. Wenig später beschrieb Frank Rosenblatt in einer Arbeit das sogenannte Perzeptron [Ros58], welches in dieser Arbeit aufgegriffen werden soll. Allerdings setzte nach der ersten Euphorie eine Phase der Stagnation ein, da die Forschung weit hinter ihren hochgesteckten Zielen zurückblieb [Kra93]. Viele Wissenschaftler waren anfangs davon überzeugt, dass man mit neuronaler Informationsverarbeitung und künstlichen Intelligenzen auf lange Sicht sämtliche Probleme lösen könnte. Doch eine KI, die sich wie ein Mensch verhält ist bis heute extrem komplex und konnte nie vollständig umgesetzt werden. Auch wenn

gerade in den letzten Jahren interessante KIs entstanden sind, wie Watson von IBM [Fer12], so haben diese Systeme noch starke Eingrenzungen, welche unter anderem durch den enormen Rechenaufwand entstehen.
Deshalb sollen auch bei der Situationsbewertung KNN nur für Teilbereiche eingesetzt werden und nicht als allgemeine Lösung. Es ist zwar in der Theorie möglich ein KNN zu verwenden um eine komplette Situation zu bewerten, die Bewertung wäre allerdings wenig transparent, da die Entscheidungen eines KNN nur schwer nachvollzogen werden können. Hinzu kommt das viel diskutierte *Value Learning Problem* [Soa15]. Dieses besagt, dass egal wie viele Grenzen man einer künstlichen Intelligenz setzt, man läuft immer Gefahr, dass sie trotzdem nicht im Interesse des Erfinders handelt. Zum einen kann dies dadurch kommen, dass die KI versucht Lücken in ihren Einschränkungen zu finden um die Aufgaben für die sie entwickelt wurde immer effizienter zu lösen oder weil die Trainingsdaten fehlinterpretiert werden ohne, dass der Erfinder eine Möglichkeit hat, dies zu bemerken.

2.4.1. Lernregeln

Jedes KNN muss erst eine Trainingsphase durchlaufen bevor es genutzt werden kann. In einer solchen Trainingsphase werden Trainingsdaten in das Netz eingegeben und aufgrund der zuvor festgelegten Lernregel werden die Gewichtungen der einzelnen Verbindungen innerhalb des Netzes angepasst. Allerdings werden nicht zwingend alle Gewichtungen angepasst. Es gibt eine Fülle von Lernregeln, die hier nicht alle beschrieben werden sollen. Es wird je eine Lernregel für überwachtes und eine für unüberwachtes Lernen erklärt, die die später beschriebenen KNN verwenden.
Beim überwachten Lernen ist es nötig, dass zu den Trainingsdaten auch die gewünschten Ergebnisse bekannt sind. Dem KNN werden nacheinander die einzelnen Datensätze zum Verarbeiten gegeben. Nachdem ein Datensatz verarbeitet wurde wird geprüft, ob das KNN das gewünschte Ergebnis geliefert hat. Ist dies der Fall, dann werden keine Veränderungen vorgenommen. War das Ergebnis allerdings nicht das gewollte, so werden die Gewichtungen im KNN angepasst.
Das unüberwachte Lernen hat einen grundsätzlichen Unterschied zum überwachten Lernen. Dem KNN wird nämlich nicht mitgeteilt welches Ergebnis erwartet wird. Dadurch kann das KNN im Laufe der Trainingsphase eigene Klassen von Situationen finden und ist nicht an vorgegebene Klassen gebunden. Dieser Unterschied ist besonders interessant für die Situationsbewertung. Man kann das KNN zum Beispiel nur mit positiven Situationen trainieren. In der darauf folgenden Anwendung kann geprüft werden, ob die aktuelle Situation zu einer der entstandenen Klassen passt und somit auch positiv ist, ohne das diese Klassen manuell definiert werden mussten.

Delta-Regel

Die Delta-Regel [Hof93] wird von überwacht lernenden KNN verwendet. Hierbei wird die Abweichung zwischen Soll- und Ist-Ergebnis betrachtet. Zu jeder gewichte-

ten Verbindung im KNN lässt sich mit Hilf der Delta-Regel berechnen, um welchen Wert diese Gewichtung angepasst werden muss. Im Aufbau des KNN wird festgelegt, welche Gewichtungen von dieser Veränderung in der Trainingsphase betroffen sind. Es wird die Differenz aus Soll-Wert S und Ist-Wert A berechnet und mit der Eingabe des Neurons E gewichtet. Zusätzlich wird ein sogenannter Lerngeschwindkeitskoeffizient α verwendet, welcher festlegt, wie schnell das KNN seine Gewichtungen anpasst.

Wettbewerbslernen

Beim Wettbewerbslernen [Hof93] konkurrieren die Neuronen darum, welches Neuron bzw., welcher Bereich an Neuronen durch einen Lernschritt beeinflusst wird. Anders als bei der Delta-Lernregel für überwachtes Lernen gilt beim Wettbewerbslernen also das „Winner-Takes-it-All"-Prinzip.
Anhand verschiedener Distanz-Maße lässt sich das Gewinner-Neuron zu einem Trainingsdatensatz bestimmen. Es ist zum einen möglich das Gewinner-Neuron anhand des Skalarproduktes aus Eingabe- und Gewichtsvektoren zu berechnen. In dem man das Neuron wählt, welches das größte Skalarprodukt aufweist. Zum anderen kann auch die Euklidische Distanz verwendet werden. Bei beiden Verfahren ist es sinnvoll nicht nur die Gewichtungen des Gewinner-Neurons anzupassen, sondern auch die der Nachbarneuronen. Während man beim ersten Verfahren die direkten Nachbarn des Gewinner-Neurons wählen kann, bietet es sich beim zweiten Verfahren an, als Faktor der Gewichtsanpassung die Euklidische Distanz zu verwenden.
Nachdem das Gewinner-Neuron und die Nachbarschaft bestimmt wurden können die Gewichtungen der Neuronen mit folgender Gleichung angepasst werden:

$$\delta w_j = \alpha h_j(\vec{x} - \vec{w_j})$$

Wobei α genau wie bei der Delta-Regel den Lernkoeffizienten darstellt, $\vec{x}$ ist der Eingabevektor. Zudem gilt: $h_j = 0$ für den Fall, das Neuron j weder das Gewinner-Neuron, noch ein direkter Nachbar davon ist, ansonsten gilt: $h_j = 1$. Bei Verwendung der Euklidischen Distanz entspricht h_j eben dieser Distanz für das Neuron j. $\vec{w_j}$ ist die alte Gewichtung, während δw_j die Veränderung der Gewichtung ist.

2.4.2. Netzarten

Im Folgenden sollen die beiden Netzarten des Perzeptrons und die des Kohonennetzes beschrieben werden. Diese eignen sich beide für die Situationsbewertung und repräsentieren dabei jeweils den Bereich des überwachten und den des unüberwachten Lernens.
Das Perzeptron als Muster-Assoziator wird häufig verwendet, um Bilder oder ähnliche Strukturen zu identifizieren. Dies ist leicht auf eine Situation übertragbar, indem man die Situation als Bild auffasst. Beim Bild hat man zum Beispiel zu jedem Bildpunkt Informationen über dessen Beschaffenheit, während man bei der Situation zu

jedem Parameter Informationen über die Ausprägung hat. Das Kohonennetz ist in Anlehnung an die Sinneswahrnehmung des Menschen entstanden. Durch das Trainieren entsteht eine Karte, auf der abgelesen werden kann, an welchem Punkt häufig Reize auftreten. Die Verknüpfung zur Situationsbewertung ist daher etwas komplexer. Wie weiter oben bereits beschrieben, kann ein Kohonennetz mit Informationen über zum Beispiel nur positive Situationen trainiert werden. Die entstehende Karte zeigt dann, an welchem Punkt der „Reiz“ stattfinden muss, damit eine neue Situation ebenfalls als positiv eingestuft wird.

Perzeptron

Das von Frank Rosenblatt entworfene [Ros58] Perzeptron ist ein spezieller Muster-Assoziator [Hof93], der über eine Vorverarbeitungsschicht und eine Muster-Assoziator-Schicht verfügt.
In der Vorverarbeitungsschicht werden die eingehenden Signale in den S-Zellen (Verteilungsneuronen) aufgenommen und an die A-Zellen (McCulloch-Pitts-Neuronen) weitergegeben. Die Verbindungen zwischen den S- und A-Zellen werden bei der Initialisierung des Perzeptrons zufällig gewählt und danach nicht mehr verändert. Wichtig ist hier, dass jede S-Zelle mit einer A-Zelle verbunden sein muss, aber nicht andersherum. Daher sollen deutlich weniger A-Zellen als S-Zellen verwendet werden. Durch diese Schicht werden also die Eingangssignale aggregiert.
In der Muster-Assoziator-Schicht befinden sich die R-Zellen (McCulloch-Pitts-Neuronen), wobei jede A-Zelle mit jeder R-Zelle verbunden ist. Jede R-Zelle generiert einen Output-Wert. Das heißt die Granularität der Klassifizierung ist direkt von der Anzahl der R-Zellen abhängig. Die Gewichtungen der Verbindungen zwischen allen A- und R-Zellen werden anfangs frei gewählt und dann durch die Trainingsphase angepasst. Ein Perzeptron kann theoretisch die Eingangsmuster in 2^n verschiedene Klassen einteilen, wobei n der Anzahl der R-Zellen entspricht. Durch die vollständige Vernetzung der A- und R-Zellen steigt der Rechenaufwand allerdings sehr schnell an.
Wir werden im Folgenden nur ein Perzeptron betrachten mit genau einer R-Zelle.
Abbildung 2.1 zeigt einen beispielhaften Aufbau eines Perzeptrons mit acht Eingangsignalen, vier A-Zellen und einer R-Zelle.

Trainingsphase Während der Trainingsphase werden Eingabevektoren verwendet um das Perzeptron zu konditionieren. Die Eingabevektoren bestehen aus Parametern einer Situation und den erwarteten Ergebnissen in Form von 0 oder 1 für jede R-Zelle des Perzeptrons. Es ist extrem wichtig, dass für jede Situation alle Parameter bekannt sind, damit der Eingabevektor vollständig ist. Dieser Vorgang muss sehr häufig mit möglichst vielen Trainingsdaten wiederholt werden, damit sich die Gewichtungen der Verbindungen zwischen den A- und R-Zellen korrekt einstellen.

Der Output a_i einer R-Zelle i kann wie folgt berechnet werden:

$$a_i = \begin{cases} 0 & \text{für } \epsilon_i < 0 \\ 1 & \text{für } \epsilon_i \geq 0 \end{cases} \text{ mit } \epsilon_i = \sum_{j=0}^{n} w_{ij} e_j$$

Wobei j die Anzahl der A-Zellen ist, die mit der R-Zelle i verbunden sind.
Anhand des Outputs einer R-Zelle findet nun das Anpassen der Gewichtungen w_{ij} zwischen allen j A-Zellen und der R-Zelle i mit Hilfe folgender Gleichung statt:

$$\delta w_{ij} = \alpha(s_i - a_i)E_j$$

Wobei s_i der erwartete Output der R-Zelle i ist und E_j der Output einer A-Zelle j ist. Zudem ist $\alpha > 0$ der Lerngeschwindigkeitskoeffizient, welcher am Anfang festgelegt werden muss. Dieser entscheidet, wie stark ein Lernschritt die Gewichtungen beeinflussen kann.
Da $s_i, a_i \in 0, 1$ ist diese Lernregel äquivalent zur Delta-Lernregel [Hof93], welche in Abschnitt 2.4.1 beschrieben wurde.

Anwendung Im Anschluss an die Trainingsphase kann das Perzeptron verwendet werden um Situationen aus der Perspektive einer Domäne zu bewerten. Hierbei werden die selben Schritte durchgeführt, wie bei der Trainingsphase, allerdings werden die Gewichtungen zwischen den A- und R-Zellen nicht mehr verändert. Wenn lediglich eine R-Zelle verwendet wird bekommt man am Ende der Bewertung 0 ($false$) oder 1 ($true$).
Wenn beispielsweise alle positiven Situationen mit dem erwarteten Ergebnis 1 angelernt wurden und alle negativen mit dem erwarteten Ergebnis 0, dann wäre eine Situation positiv, wenn das Perzeptron sie in der Anwendung mit 1 bewertet. Dies wird sehr komplex, wenn mehrere R-Zellen verwendet werden. Man könnte damit die Situationen zwar besser in Gruppen einteilen und hätte somit präzisere Aussagen als positiv und negativ, doch werden dann auch deutlich mehr Trainingsdaten benötigt. Zudem wächst die Gefahr, dass die Klassifizierung des Perzeptrons nicht der gewünschten Klassifizierung entspricht, weil Trainingsdaten falsch ausgewählt wurden.

Laufzeit Durch die Reduktion von Eingangssignalen in der Vorverarbeitungsschicht ist die Laufzeit eines Perzeptrons auch bei vielen verwendeten Parametern geringer, als wenn nur eine Muster-Assoziator-Schicht verwendet wird. Zudem verbessert sich die Laufzeit der Trainingsphase, da in unserem Fall nur eine R-Zelle vorhanden ist. In jedem Trainingsschritt müssen also nur so viele Gewichtungen angepasst werden, wie es A-Zellen gibt, da jede A-Zelle mit der einen R-Zelle verbunden ist. Sowohl die Laufzeit in der Trainingsphase, als auch in der Anwendung wächst maximal linear mit der Anzahl der verwendeten Parameter.

Das Perzeptron hat dieselben Grenzen wie der Muster-Assoziator, es kann also nur

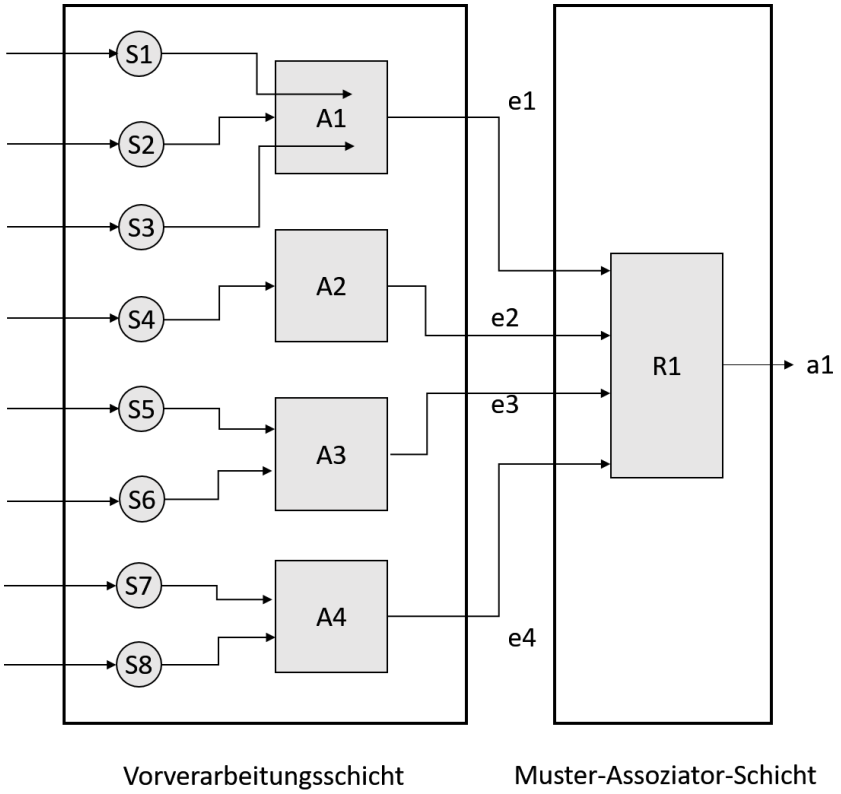


Abbildung 2.1.: Perzeptron mit acht Eingangszellen (S-Zellen), vier Vorverarbeitungszellen (A-Zellen) und einer Muster-Assoziator-Zelle (R-Zelle)

Muster lernen, deren Eingangsmuster linear unabhängig sind [Hof93]. Des Weiteren können mit Hilfe eines Perzeptrons nur solche Situationen bewertet werden, für die alle Parameter bekannt sind. Hier muss auf viele Parameter verzichtet werden, die nicht immer verfügbar sind, was eine starke Einschränkung der Bewertung mit sich bringt.
Zudem ist es leider nicht möglich eine Aussage über die Sicherheit eines Ergebnisses zu treffen, oder über die Berechnung allgemein.

Kohonennetze

Auch Kohonennetze könnten für die Bewertung einer Situation durch eine Domäne verwendet werden. Hierbei würde man das Kohonennetz lediglich mit Situationen trainieren, die entweder positiv oder negativ sind. Bei Verwendung einer zweidimensionalen Output-Schicht sollten sich dann Cluster bilden. Nach dem Training kann eine Situation durch das Kohonennetz bewertet werden, indem man die Parameter der Situation als Eingabevektor an das Netz gibt und dann prüft, ob die Erregung des Netzes in einem der beim Training entstandenen Cluster liegt. Ist dies der Fall,

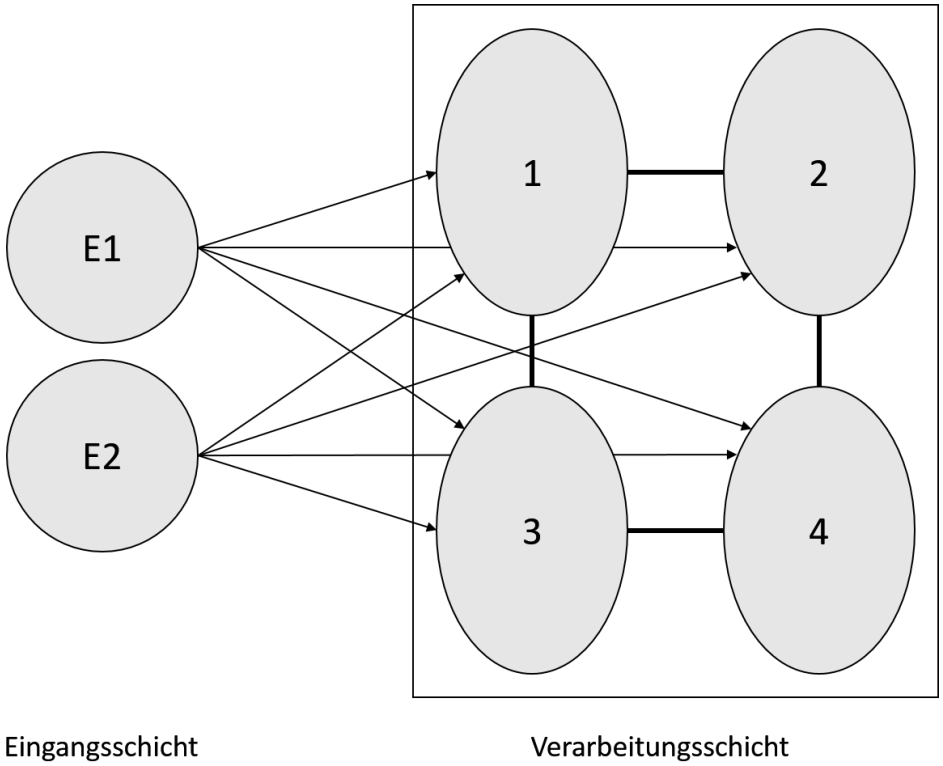


Abbildung 2.2.: Kohonennetz mit zwei Eingangsneuronen und vier Ausgangsneuronen

so handelt es sich um eine Situation, die zu den vorher angelernten Situationen passt. Die häufigste Form des Kohonennetzes ist jene mit einer zweidimensionalen Output-Schicht, welche in der Abbildung 2.2 dargestellt wird. Die Situationen werden als n-dimensionale Vektoren an die Eingabeneuronen des Netzes gegeben. Die Eingabeneuronen sind in der Abbildung 2.2 als E1 und E2 bezeichnet. Die Anzahl der Parameter der Situation werden durch die Dimension des Eingabevektors und somit durch die Menge der Eingabeneuronen festgelegt. Durch das Wettbewerbslernen, welches in Abschnitt 2.4.1 beschrieben wurde, findet eine Anpassung der Positionen der Neuronen auf der Verarbeitungsschicht (hier mit 1 bis 4 bezeichnet) statt. Die Verbindungen zwischen den Neuronen auf der Verarbeitungsschicht stellen dabei die Nachbarschaftsbeziehung dar. In einer konkreten Anwendung werden deutlich mehr Eingabeneuronen und Ausgabeneuronen verwendet. Zusätzlich zu der normalen Trainingsphase müssen vor der ersten Anwendung noch die Cluster auf der Verarbeitungsschicht bestimmt werden. Da die Anzahl der im Kohonennetz entstehenden Cluster nicht vorher bekannt ist, ist DBSCAN ein geeigneter Algorithmus um die Cluster zu identifizieren [EKSX96].

Trainingsphase Die Trainingsphase des Kohonennetzes läuft sehr ähnlich zu der des Perzeptrons ab. Nachdem das Kohonennetz mit zufälligen Gewichtungen zwischen allen Neuronen der Eingabe- und der Verarbeitungsschicht initialisiert wurde, werden nacheinander die Trainingsdaten angewendet um es zu konditionieren. Jeder Trainingsdatensatz beinhaltet einen Eingabevektor $\vec{x}$ der aus den Parametern der Situation besteht. Anders als beim Perzeptron wird hier kein erwartetes Ergebnis mitgegeben, da es sich um unüberwachtes Lernen handelt. Dieser Eingabevektor wird an die Eingangsschicht des Kohonennetzes angelegt. Da das Kohonennetz das Wettbewerbslernen verwendet, wird nun geprüft bei welchem Neuron es sich um das Gewinner-Neuron handelt und welche Neuronen die Nachbarschaft dieses bilden. Dies kann, wie in Abschnitt 2.4.1 beschrieben, zum einen durch die Skalarprodukte aus Gewichts- und Eingabevektor oder durch die Euklidische Distanz geschehen. Nachdem sowohl Gewinner-Neuron als auch Nachbarschaft bekannt sind, können die Gewichtungen dieser Neuronen mit der ebenfalls in Abschnitt 2.4.1 beschrieben Gleichung angepasst werden.
Beim Trainingsprozess ist zu beachten, dass α nicht wie bei der Delta-Regel konstant sein darf, sondern im Laufe des Trainings langsam verringert werden muss, damit das Kohonennetz konvergiert [Koh82].
Zum Schluss müssen noch die Cluster auf der Output-Schicht mittels DBSCAN ermittelt werden.

Anwendung Bei der anschließenden Anwendung des Kohonennetzes wird eine Situation wieder durch einen Eingabevektor dargestellt, der aus den Parametern der Situation besteht. Dieser Eingabevektor wird nun an die Eingangsschicht angelegt und man kann das Gewinner-Neuron bestimmen. Liegt dieses Gewinner-Neuron in einem der Cluster, welche in der Trainingsphase bestimmt wurden, so passt die Situation zu den vorher angelernten. Waren zum Beispiel die Trainingsdaten positive Situationen, so handelt es sich bei der gerade bewerteten Situation ebenfalls um eine positive. Des Weiteren wäre es möglich zu bestimmen, ob die Situation im Kern oder am Rand eines Clusters liegt, bzw. wie weit sie von dem nächsten Cluster entfernt ist.

Laufzeit Der Rechenaufwand eines Kohonennetzes zur Situationsbewertung ist deutlich größer als der eines Perzeptrons. Da bei einem Kohonennetz keine Vorverarbeitungsschicht vorhanden ist, in der die Eingabesignale reduziert werden, wächst die Laufzeit auch deutlich schneller an. Durch die vollständige Vernetzung von Ein- und Ausgabeneuronen, muss für alle $n \times m$ Kombinationen aus n Eingangsneuronen und m Ausgangsneuronen geprüft werden, ob es sich um das größte Skalarprodukt aus Gewichts- und Eingabevektor handelt bzw. welche Euklidische Distanz die beiden Vektoren haben, um das Gewinner-Neuron zu ermitteln.

Der Vorteil des Kohonennetzes gegenüber dem Perzeptron ist die feinere Klassifizierung der Situationen, denn es muss nicht anfangs festgelegt werden, wie viele

Klassen von Situationen es gibt. Durch das ausschließliche Verwenden von zum Beispiel positiven Situationen in der Trainingsphase lernt das Kohonennetz lediglich, welche Situationen positiv sind und klassifiziert diese dann selbst. Leider ist es danach nicht möglich genauere Aussagen über die einzelnen Klassen zu machen, man kann höchstens bestimmen, ob eine neue Situation im Kern eines Clusters liegt oder wie weit sie von diesem entfernt ist, bzw. wie weit sie von der Grenze eines Clusters entfernt ist.
Leider geht die Präzision der Klassifizierung mit höherem Rechenaufwand einher. Umso genauer die Klassen auf der Verarbeitungsschicht eingeteilt werden sollen, umso mehr Neuronen benötigt die Verarbeitungsschicht, wodurch die Anzahl der Verbindungen zwischen Eingangs- und Verarbeitungsschicht dramatisch ansteigt.

2.4.3. Zusammenfassung

Vorteile

Der Vorteil eines KNN für die Bewertung von Situationen aus der Perspektive einer Domäne ist, dass man so auch Domänen einbeziehen kann, für die noch keine klar definierten Regeln existieren. Zudem könnte das System auch nach der Trainingsphase weiter lernen, in dem es beobachtet, welche Situationen vom Nutzer akzeptiert werden und welche eben nicht. Es wäre theoretisch auch denkbar alle Domänen mit KNN darzustellen. Hierbei würde man sich allgemein den Aufwand sparen Regeln und Gesetzmäßigkeiten für die Domänen zu entwickeln und zu überprüfen. Diese Aufgabe würde komplett von der Trainingsphase des KNN übernommen werden und somit größtenteils automatisch ablaufen. Das Aufbereiten der Trainingsdaten würde allerdings einen Teil des ersparten Aufwandes kompensieren und kann sogar womöglich aufwendiger sein. Dieser Ansatz sollte in der Praxis jedoch vermieden werden, da KNN auch einige zum Teil gravierende Nachteile mit sich bringen.

Nachteile

Der für jede Art von KNN anfallende Aufwand zur Aufbereitung der Trainingsdaten ist je nach Umfang der Domäne im schlimmsten Fall ähnlich groß oder sogar größer, als beim Entwickeln von Regeln. Vor allem bei Ansätzen des überwachten Lernens ist der Aufwand der Aufbereitung von Trainingsdaten enorm groß, wenn eine feinere Klassifizierung vorgenommen werden soll. Des Weiteren gibt es keine Gewissheit, dass die Trainingsdaten korrekt ausgewählt wurden, um das KNN so zu konditionieren, wie es beabsichtigt war. Diese als *Value Learning Problem* [Soa15] bekannte Schwäche von künstlichen Intelligenzen wird schon lange diskutiert.
Auch die Laufzeit von Künstlichen Neuronalen Netzen ist eines der immer noch aktuellen Probleme. Bei den meisten Arten von KNN steigt die Anzahl der Rechenschritte sowohl in der Trainingsphase als auch in der Anwendung exponentiell und lässt sich somit auch mit leistungsstärkerer Hardware nur bedingt ausgleichen. Da die Bewertungen von Situationen in Echtzeit benötigt werden, ist die Zeit für die

Bewertung auf wenige Millisekunden limitiert.
Aus eben diesen Nachteilen ergeben sich auch teilweise rechtlich schwer zu beantwortende Fragen: Wer haftet, wenn ein KNN eine Situation falsch bewertet oder die Bewertung zu spät fertig ist und dadurch Schäden entstanden sind?
Hier könnten die aktuellen Diskussionen zu autonomen Fahrzeugen [MD15] [LTL] von Interesse sein, da die Ergebnisse für viele autonome Systeme wegweisend sein dürften.

Der Lösungsansatz mit einem KNN ist bei der Situationsbewertung nur teilweise zielführend, da die Menge der Eingabeparameter nicht fest ist. Man kann also nur solche Teile der Situation bewerten, für die die Parameter zu jeder Zeit bekannt sind oder man geht das Risiko ein, in einigen Fällen keine Bewertung zu erhalten, wenn die Informationen über die Situation zu lückenhaft sind. Lediglich Domänen, die nicht durch einfache Regeln und Gesetzmäßigkeiten dargestellt werden können, bieten sich an für die Berechnung mit Hilfe eines KNN. Alle anderen Domänen sollten durch Regeln und Gesetzmäßigkeiten manuell entworfen werden.

3. Situationsbewertung

Nach dem im vorherigen Kapitel einige Grundlagen und Ansätze zum Umgang mit Situationen vorgestellt wurden, soll in diesem Kapitel ein Framework entwickelt werden um Situationen abzubilden und zu bewerten. Das Framework soll dabei explizit nicht die Möglichkeit bieten Abläufe von Aktionen, zum Erreichen einer bestimmten Situation, zu berechnen. Der Schwerpunkt des Frameworks liegt auf der Bewertung von Situationen unter Einbeziehung verschiedener Regeln. Damit soll es ermöglicht werden ungewöhnliche Situationen, durch vordefinierte Regeln, automatisch zu erkennen.

3.1. Grundlage

In diesem Abschnitt wird anfangs erneut auf die drei Punkte Wirklichkeit, Situation und Universum eingegangen. Dabei werden diese in einer Form beschrieben, die als Grundlage für das Framework dienen soll.

3.1.1. Wirklichkeit

Wie im vorherigen Kapitel gehe ich von zwei Arten von Wirklichkeiten aus. Der einen *echten* Wirklichkeit und einer unendlichen Menge an *fiktiven* Wirklichkeiten. Ich gehen davon aus, dass es sich bei der echten Wirklichkeit um eine *Open World* handelt. Die Möglichkeiten dieser Wirklichkeit sind praktisch unendlich und können niemals vollständig beschrieben werden. Die fiktiven Wirklichkeiten werden im Folgenden als Gegensatz zur Open World als *Closed World* nach Reiter [Rei77] aufgefasst, da ihre Möglichkeiten endlich sind und zumindest theoretisch vollständig beschrieben werden können.
Der Beobachter, welcher als Anwender bezeichnet wird und für den die Situation bewertet werden soll, ist immer Teil der echten Wirklichkeit, aber nicht zwingend Teil der fiktiven Wirklichkeit. Sobald der Anwender Teil der fiktiven Wirklichkeit ist, die für ihn bewertet werden soll, ist davon auszugehen, dass die Situationen nicht mehr vollständig beschrieben werden kann, weil es sich nicht um einen objektiven Beobachter handeln kann. Zudem wird davon ausgegangen, dass die echte Wirklichkeit auf fiktive reduziert werden kann und auch reduziert werden muss um Bewertungen vorzunehmen.

3.1.2. Situation

Eine Situation wird im Folgenden als eine Menge von Parametern in einer bestimmten Ausprägung aufgefasst, wobei einige dieser Parameter zu bestimmten Zeitpunkten dem Bewertungssystem unbekannt sein können. Ist dies der Fall, dann werden Sie mit *NULL* gekennzeichnet. Dies geschieht in Anlehnung an den Null-Wert aus der Datenbanklehre. Der NULL-Wert ist also genau so zu verstehen, wie in Datenbanken, weshalb auf die gleiche Problematik geachtet werden muss, wie in Datenbankmanagementsystemen [Kle01]. Die Situationen werden nicht explizit unterteilt, damit sie möglichst abstrakt gestaltet werden können. Eine Einteilung kann auf Grundlage der Ausprägungen der Parameter erfolgen, die eine Situation beschreiben.
Die Parameter hingegen werden in *normale* Parameter, welche im Folgenden einfach Parameter genannt werden, und *abgeleitete* Parameter eingeteilt. Bei den normalen Parametern handelt es sich um solche, die durch geeignete Methoden sowohl in der echten als auch in den fiktiven Wirklichkeiten gemessen werden können. In den fiktiven Wirklichkeiten handelt es sich hierbei nicht zwingend um ein Messen im herkömmlichen Sinne, sondern kann z.B. bei Computerprogrammen auch durch das Auslesen bestimmter Variablen passieren. Die abgeleiteten Parameter hingegen können nicht gemessen werden. Sie werden durch logische Schlussfolgerung unmittelbar aus den normalen Parametern abgeleitet. Auf abgeleitete Parameter wird in dieser Arbeit jedoch nicht weiter eingegangen.
Ebenfalls wird auf die Dynamik zwischen Aktionen nicht explizit eingegangen. Es wird angenommen, dass zwischen je zwei Situationen eine Aktion oder ein Event, bzw. eine Abfolge von Aktionen oder Events stattgefunden hat. Eine Situation ist somit in sich geschlossen und es muss nicht auf andere Situationen zurückgegriffen werden um sie zu beschreiben. Eine Reihenfolge von Situationen kann somit lediglich durch zum Beispiel einen Parameter, der die Zeit angibt, umgesetzt werden.

3.1.3. Universum

Da das Konstrukt des Universums in der Situationsbewertung nicht explizit benötigt wird, wird hier nur eine sehr grobe Definition gegeben. Das Universum umfasst alles, was für die Situationsbewertung nötig ist oder nötig sein könnte. Da sich die gesamte Situationsbewertung in der echten oder einer fiktiven Wirklichkeit abspielt und auch alle Regeln als Teil dieser Wirklichkeiten aufgefasst werden können, kann das Universum als die Menge aller möglichen Wirklichkeiten verstanden werden.

3.2. Struktur

In diesem Abschnitt wird auf die Struktur der Situationen, der Domänen bzw. Naturgesetze und der Person eingegangen. Diese Strukturen bilden die Grundlage für die Bewertungsfunktion im nächsten Abschnitt.

3.2.1. Situation

Eine Situation besteht aus Parametern, die Informationen über die Situation enthalten, und aus Indikatoren, welche die Ergebnisse der Situationsbewertung widerspiegeln. Es ist möglich, dass zwei verschiedene Situationen, innerhalb der Situationsbewertung, als identisch aufgefasst werden. Dies kann daran liegen, dass sich die Situationen in der Wirklichkeit zwar unterscheiden, aber die Informationen der Situation, in der die Unterschiede liegen, nicht bei der Bewertung betrachtet werden. Die Abbildung 3.1 stellt eine Situation schematisch mit ihren Einzelteilen, die hier beschrieben werden, dar und zeigt auch die Verbindung zur Wirklichkeit auf.

Parameter

Die Parameter sind die Informationen, die in die Situationsbewertung eingehen. Eine Situation hat für jeden Parameter, der durch das Universum definiert wurde eine Ausprägung zu jedem Zeitpunkt. Es ist jedoch möglich, dass es Informationen gibt, für die durch das Universum kein Parameter definiert ist. Diese Informationen werden deshalb nicht betrachtet und die Situation ist dadurch eine Reduktion der Wirklichkeit. Sollte ein Messen einer Information nicht möglich sein oder die Ausprägung des Parameters durch andere Gründe nicht bekannt sein, wird dies mit der Ausprägung *NULL* gekennzeichnet. Im Folgenden werden auch Relationen zwischen zwei oder mehr Objekten der Wirklichkeit als Parameter dargestellt. Somit kann jede Situation allein durch Parameter dargestellt werden.
In Anlehnung an die Infons der Situationen Theorie in der Form von Devlin [Dev91], werden die Parameter einer Situation als Tupel der folgenden Form dargestellt:

$$<< R, a_1, ...a_n, b_1, ..., b_m, P >> .$$

Wobei R eine n-stellige Relation ist und $a_1, ..., a_n$ die Attribute der Relation sind. $b_1, ..., b_m$ sind Elemente, die Teil jedes Parameters sein können, unabhängig davon was durch die Relation R definiert wird. Des Weiteren handelt es sich bei P um eine Erweiterung der Polarität der Situationen Theorie. In der Situationen Theorie wäre $P \in \{0, 1\}$ und zeigt somit nur an, ob der Infon für die Situation zutrifft oder nicht. Dies wird dargestellt als $P \in \{true, false\}$ für Parameter, die keinen eigenen Wert haben, also nur Relationen zwischen Objekten der Wirklichkeit angeben oder einfache Ja-/Nein-Informationen, wie zum Beispiel „es regnet“. Möchte man jedoch eine Ausprägungsstärke des Parameters angeben, wie es bei den funktionalen Fluents des Situationen Kalküls möglich ist, so gilt $P \in \mathbb{R}$. Damit lassen sich Informationen darstellen wie „die Temperatur beträgt 24 Grad Celsius“. Solche Informationen ließen sich auch mit Parametern darstellen, deren Polarität nur $true$ oder $false$ sein kann, allerdings könnten dann keine allgemeinen Relationen wie „die Temperatur beträgt x Grad“ verwendet werden.
Die Elemente $b_1, ..., b_m$ können einen Typ der folgenden Liste haben:

- TIM - Angabe eines Zeitpunktes

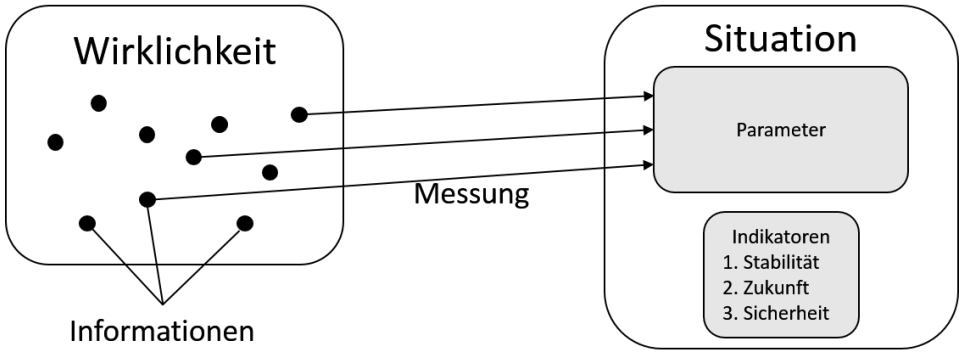


Abbildung 3.1.: Aufbau einer Situation und Ableitung aus der Wirklichkeit

- LOC - Angabe eines Ortes
- INF - Angabe einer Menge von Parametern auf den sich der Aktuelle bezieht

Jedes Element dieser Liste kann nur einmal in einem Parameter vorkommen und wenn es nicht vorkommt wird es als *NULL* angenommen.

Indikatoren

Ein Indikator ist ein Wert, der bei der Situationsbewertung berechnet wird. Da einige Indikatoren für jede Situation berechnet werden, werden diese explizit zur Situation gezählt. Durch die Domänen und Person kann die Menge der Indikatoren allerdings noch erweitert werden.
Ein Indikator wird als Tupel der Form (Indikator-Wert, Sicherheit) dargestellt. Der erste Teil gibt die konkrete Ausprägung des Indikators an, während der zweite Wert angibt mit welcher Sicherheit dieses Ergebnis berechnet wurde. Umso geringer die Sicherheit eines Indikators ist, umso weniger sollte diesem Wert vertraut werden. Die Sicherheit eines Indikators selbst sollte nicht mit dem Indikator Sicherheit verwechselt werden, da dieser angibt, wie sicher eine Situation für alles ist, was von dieser betroffen ist. Sowohl für den Indikator-Wert, als auch für die Sicherheit gilt, dass sich diese in dem Intervall zwischen 0 und 1 bewegen.
Die folgenden Indikatoren gehören zu jeder Situation:

- Stabilität - Gibt an, wie stabil die Situation ist, also wie wahrscheinlich es ist, dass sich ohne Eingreifen des Anwenders die Situation ändert.
- Zukunft - Gibt an, wie wahrscheinlich es ist, dass eine nicht durch den Anwender hervorgerufene Änderung positiv ist.
- Sicherheit - Gibt an, wie sicher die Situation für alles von ihr betroffene ist.

3.2.2. Domänen

Durch Domänen wird Wissen bereitgestellt, das in die Situationsbewertung einbezogen wird. Eine Domäne besteht aus verschiedenen Skalen und Regeln, die es ermöglichen die Parameter einer Situation auf Indikatoren abzubilden und wird in Abbildung 3.2 dargestellt. Die Auswirkungen einer Regel auf einen bestimmten Indikator kann zusätzlich durch Gewichtungen reduziert werden. Die Domänen lassen sich in die folgenden vier Gruppen einteilen.

- Wissenschaftsdomänen - Physik, Biologie, Betriebswirtschaftslehre...
- Gesetzesdomänen - dt. Zivilrecht, dt. Strafrecht, dt. Datenschutz...
- Organisationsdomänen - Verwaltung, Demokratie, Infrastruktur...
- Kulturdomänen - regionale Domänen, ethnische Domänen, religiöse Domänen...

Eine Domäne könnte also Regeln beinhalten um eine Situation aus Sicht der deutschen Datenschutzgesetze zu bewerten oder Regeln um sie aus Sicht der Finanzwirtschaft zu bewerten. Eine Bewertung durch eine Domäne die eine Kultur repräsentiert ist deutlich komplexer als bei den ersten beiden, da eine Kultur sich sehr schwer eingrenzen lässt und die Regeln häufig sehr personenspezifisch sind. Daher werde ich im Anwendungsbeispiel nicht weiter auf diese Art der Domänen eingehen.
Jede Domäne kann neben den Regeln auch eigene Indikatoren enthalten. Dies ist nötig um weitere Bewertungsaspekte für die Situation zu schaffen. Zum Beispiel können aus der Domäne des deutschen Datenschutzes verschiedene Indikatoren für Datenschutz und Datensicherheit folgen. Die Datensicherheit könnte durch die drei Variablen *Verfälschungsgefahr*, *Zerstörungsgefahr* und *Gefahr der unzulässigen Weitergabe* [WMR10] abgebildet werden. Damit ist ein präziseres Reagieren auf die Situation möglich, da die Daten nicht nur auf Standardindikatoren aggregiert werden, sondern viel mehr die einzelnen Teilbereiche durch die für sie relevanten Informationen dargestellt werden.

Skalen

Bevor die Regeln einer Domäne definiert werden können benötigt man Skalen, die die gemessenen Werte aus der Wirklichkeit in Zahlen übersetzen. Diese Übersetzung könnte natürlich auch in jeder Regel separat durchgeführt werden. Um die Regeln übersichtlicher zu gestalten, ist es allerdings sinnvoll dies separat zu machen.
Eine Skala ist also eine Abbildung der Form

$$\mathcal{S} : E \to A.$$

Wobei E die Menge der, in der Wirklichkeit gemessenen, Werte ist und A eine Menge von Zahlen. Wenn zum Beispiel in der Wirklichkeit eine Beziehung zwischen zwei Objekten erfasst werden soll, dann wird ein bestehen dieser Beziehung durch *true* und ein nicht bestehen durch *false* dargestellt. Die Abbildung $\mathcal{S}$ ist dann so zu

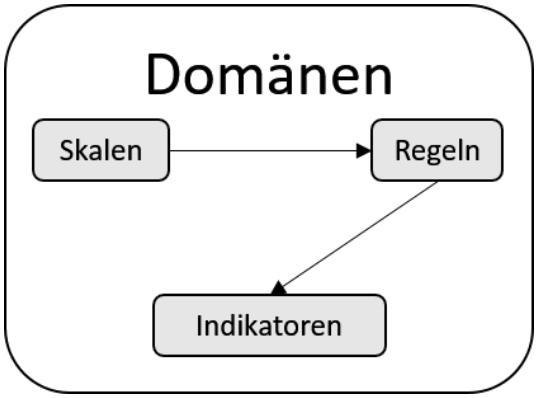


Abbildung 3.2.: Aufbau einer Domäne

definieren, dass sie die Menge $E = \{true, false\}$ auf die Menge $A = \{1, 0\}$ abbildet. Wenn es mehr Möglichkeiten als *true* oder *false* gibt, sind Urbild und Bild von $\mathcal{S}$ geeignet zu erweitern.
Des Weiteren kann die Situationsbewertung durch Skalen teilweise vereinheitlicht werden. Nimmt man an, man hat eine Domäne, die als Parameter die Temperatur in Grad Kelvin benötigt. Dann ist es möglich verschiedene Skalen zu definieren in denen jeweils Grad Celsius oder Grad Fahrenheit in Grad Kelvin umgerechnet werden.

Regeln

Die Regeln werden in die drei Gruppen *einfache* Regeln, *stufenweise* Regeln und *quantifizierte* Regeln eingeteilt, wie bei Cios et al [CPSK07]. Die jeweiligen Regelarten werden hier jeweils nur kurz erklärt, danach wird auf die Bedingungen *(engl. condition)* und die Schlussfolgerungen *(engl. conclusion)* der Regeln eingegangen. Zum Abschluss dieses Abschnittes wird noch auf die Gewichtungen, Sicherheit und Elastizitäten von Regeln eingegangen.

einfache Regeln IF $condition_1 \wedge ... \wedge condition_n$ THEN $conclusion$
Durch einfache Regeln können einfache, meist abstrakte Zusammenhänge dargestellt werden. Zum Beispiel „wenn es regnet, wird der Boden draußen nass“. Dies ist in sofern eine sehr abstrakte Aussage, da hier von Regen gesprochen wird, allerdings nicht woraus dieser Regen genau besteht oder um welche Menge es sich handelt. Des Weiteren wird auch nicht darauf eingegangen, dass sich Tropfen in der Luft bilden, die zum Erdboden fallen. Dieser an sich nicht triviale Vorgang wird als „es regnet“ zusammengefasst.

stufenweise Regeln IF $\tau(X)$ THEN $\mu(Y)$
Mit stufenweisen Regeln können funktionale Beziehungen dargestellt werden. Dabei

sind τ und μ Abbildungen die jeweils die Bedingung X und die Schlussfolgerung Y in eine feste Richtung verschieben.
Hierdurch können Beziehungen, wie „für jeden eingeloggten Nutzer werden 100 MB Arbeitsspeicher benötigt“, dargestellt werden. Dies lässt sich theoretisch auch mit einfachen Regeln abbilden, in dem für jede explizite Nutzeranzahl eine Regel definiert wird, die angibt wie viel Arbeitsspeicher bei dieser Anzahl benötigt wird. Die Formulierung von stufenweisen Regeln ist jedoch deutlich schlanker und einfacher zu verändern.

quantifizierte Regel IF *condition* THEN θ(*conclusion*)
Bei der letzten Art von Regeln handelt es sich um solche, die eine Wahrscheinlichkeit ausdrücken können. Wobei θ eine Funktion ist, die mit einer gewissen Wahrscheinlichkeit auf die Schlussfolgerung abbildet oder nicht. Hier bleiben viele Möglichkeiten der Modellierung offen. Die Funktion θ kann zum Beispiel im negativ Fall kein Ergebnis oder ein anderweitig definiertes zurückgeben. Damit lassen sich auch Ungenauigkeiten in den Regeln abbilden, wie „wenn es gewittert, besteht die Gefahr, von einem Blitz getroffen zu werden“.
Eine Kombination von stufenweisen und quantifizierten Regeln wäre zudem auch möglich, um darzustellen, dass eine Gefahr immer wahrscheinlicher oder unwahrscheinlicher wird, umso stärker oder schwächer die Bedingung ausgeprägt ist.

Für alle Regeln gilt, dass sie eine Bedingung und eine Schlussfolgerung haben. Durch die Bedingung wird angegeben, wie die Situation beschaffen sein muss, damit die Regel greift. Da in der Wirklichkeit auf verschiedene Arten Informationen gesammelt und auch dargestellt werden können, ist es nötig, dass diese vorher in eine einheitliche Form gebracht werden. Dies wird durch die Skalen bewerkstelligt. Die Bedingung ist somit ein aussagenlogischer Term, dessen Variablen durch die Skalen der Domäne definiert sind. Dieser Term kann dann zu *true* oder $false$ ausgewertet werden, woraus sich entscheidet, ob eine Regel angewendet wird oder nicht. Im Falle von quantifizierten Regel wird die Bedingung nicht zu *true* oder *false* ausgewertet, sondern zu einer Zahl, die die Ausprägung der Bedingung angibt.
Die Schlussfolgerung einer Regel wird im Folgenden auch Ergebnis einer Regel genannt. Da wir eine Menge von Regeln verwenden für die Bewertung einer Situation, handelt es sich genau genommen um ein Teilergebnis dieser Bewertung. Die Schlussfolgerung ist eine Menge von Tupeln der Form (Indikator-Wert, Gewichtung, Sicherheit), wobei jedes Tupel genau einem Indikator der Domäne oder der Situation zugewiesen sein muss. Der Indikator-Wert gibt an, wie der jeweilige Indikator laut der Regel ausgeprägt ist und der dritte Wert gibt an, wie sicher das Ergebnis der Regel ist. Die Gewichtung eines Ergebnisses gibt an, wie stark der Einfluss auf das Gesamtergebnis ist. Alle drei Werte müssen sich immer im Intervall zwischen 0 und 1 befinden.
Bei der Gewichtung handelt es sich um einen Faktor, der angibt wie relevant der Wert für die gesamte Situation ist. Wird ein Indikator durch mehrere Regeln beeinflusst,

ist es möglich, dass einige Regeln nur einen geringen Einfluss (Bsp.: Gewichtung < 0,25) auf diesen Indikator haben, während andere einen stärkeren Einfluss (Bsp.: Gewichtung > 0,75) haben. Eine Gewichtung muss für jeden Indikator angegeben werden, zu dem eine Regel einen Wert in der Schlussfolgerung hat. Wenn ein Gewicht von 0 angegeben wird, dann beeinflusst die Regel diesen Indikator niemals.
Die Sicherheit zu einem Wert gibt an, wie wahrscheinlich es ist, dass dieser Wert vertrauenswürdig ist. Wenn ein Wert vollständig vertrauenswürdig ist, liegt die Sicherheit bei 1. Es ist in diesem Fall davon auszugehen, dass keine fehlerhaften oder ungenauen Werte in die Berechnung eingegangen sind und alle Parameter, die verwendet wurden, wurden als schlüssig anerkannt.
Diese Unsicherheiten können verschiedene Ursprünge haben. Zum Beispiel ist es sinnvoll in einer quantifizierten Regel eine Unsicherheit einzubauen, da das Ergebnis der quantifizierten Regel an sich schon einer gewissen Unwahrscheinlichkeit unterliegt und somit niemals gesagt werden kann, dass das Ergebnis auf alle Fälle korrekt ist. Allerdings können auch normale oder stufenweise Regeln eine Unsicherheit beinhalten. Dies kann daraus resultieren, dass der Zusammenhang, der durch die Regel abgebildet wird, nicht durch einen Beweis formal sichergestellt werden kann, sondern durch Beobachtung oder womöglich Schätzung und Vermutungen entstanden ist. Die Vertrauenswürdigkeit des Wissens, das hinter einer jeden Regel steckt, sollte durch die Sicherheit der jeweiligen Regel wiedergegeben werden, damit sie in das Ergebnis der Situationsbewertung einfließen kann. Zu guter Letzt kann Unsicherheit auch durch ungenaue, fehlerhafte oder fehlende Parameter entstehen. Die Abbildung dieser Unsicherheit ist jedoch deutlich komplexer und muss durch geeignete Bedingungen für die jeweiligen Regeln geprüft werden.
Neben der Sicherheit ist auch das Konzept einer Elastizität der Regeln nützlich um die Situationsbewertung zu verbessern. Vor allem bei den stufenweisen Regeln kann dieses Konzept eingesetzt werden. Denn hier ist nicht nur die Verletzung einer Regel interessant, also das Abweichen von einem Soll-Wert, sondern es ist zusätzlich auch interessant, wie weit ein Ist-Wert von einem Soll-Wert entfernt ist und wie ausschlaggebend diese Entfernung ist. Um dies an einem Beispiel zu zeigen eignet sich ein theoretisches System, dass bei Überlastung vollständig ausfällt. Zudem wird angenommen, dass nach dem Ausfallen des Systems kein weiterer Schaden am System entstehen kann und der Ausfall als der schlimmst mögliche Fall gesehen wird. Bei diesem System wäre das Überschreiten der Grenze der maximalen Belastung am dramatischsten, da das System in diesem Moment ausfällt. Wird diese Grenze allerdings deutlich überschritten ist der Schaden immer der gleiche, da das System im schlimmsten Fall lediglich völlig ausfallen kann.
Verschiedene Formen der Elastizitäten lassen sich in drei Gruppen einteilen. Die erste und einfachste umfasst alle linearen Elastizitäten, welche nicht explizit in einer Regel zu finden sind. Diese Form der Elastizität liegt standardmäßig bei jeder Regel vor. Der oben beschriebene Fall ließe sich durch eine extreme Form einer logarithmischen Elastizität näherungsweise darstellen. In dem Beispiel müsste durch die Elastizität zwar eine Schranke definiert werden, aber dies ist durch den Logarithmus näherungsweise möglich. Die letzte Gruppe bilden die exponentiellen Elastizitäten,

bei solchen Elastizitäten wächst der Verlust oder Schaden durch ein Abweichen vom Soll-Wert exponentiell mit der Abweichung.

Verwendung von KNN für Komplexe Domänen

Im vorherigen Kapitel wurden Künstliche Neuronale Netze, wie das Perzeptron und das Kohonennetz, beschrieben. Daher soll hier noch einmal darauf eingegangen werden, wie diese für die Situationsbewertung verwendet werden können.
Der Einsatz eines KNN zum Abbilden einer Domäne bietet sich immer dann an, wenn eine Domäne zu komplex ist, als dass man manuell die nötigen Regeln bilden könnte um die Inhalte der Domäne geeignet wiederzugeben. Sollte dieses Problem der Komplexität nicht vorliegen, ist das Verwenden von manuell geschrieben Regeln immer vorzuziehen, da bei einem KNN nicht nachvollzogen werden kann, wie die Bewertung zu Stande kam.
Die Komplexität einer Domäne kann sich durch eine sehr große Menge von Regeln ausdrücken oder dadurch, dass aus dem vorhandenen Wissen einer Domäne keine Regeln abgeleitet werden können, die das Wissen geeignet repräsentieren. Ab wann eine Domäne zu viele Regeln benötigt, um geeignet abgebildet zu werden, kann nicht pauschal festgelegt werden. Es handelt sich also immer um eine Ermessensfrage, ab wann die Menge der Regeln zu groß ist. Hierbei sollte auch immer die Komplexität des KNN betrachtet werden. Wenn die benötigte Zeit für die Trainingsphase des KNN länger als die für das entwerfen der Regeln ist, kann mit Gewissheit gesagt werden, dass ein KNN für die Domäne nicht in Frage kommt. Sobald es jedoch nicht möglich ist für eine Domäne geeignete Regeln abzuleiten, weil zum Beispiel fundiertes Wissen über die Domäne fehlt, ist ein KNN dem manuellen Ansatz vorzuziehen. Bei einem KNN darf jedoch nicht vergessen werden, dass eine Art Mentor benötigt wird, der dem KNN mitteilt, welche Situationen aus den Trainingsdaten positiv sind und welche nicht, bzw. der die Ergebnisse für das Training vorgibt. Auch hier kann ein großer Zeitaufwand entstehen.
Wie im Vorherigen Kapitel erklärt, kann ein Perzeptron verwendet werden um Situationen zu bewerten, indem man die Situation, welche aus Parametern mit bestimmten Ausprägungen besteht, als Muster auffasst. Im Endeffekt prüft das Perzeptron nun, ob die Situation einem bekannten Muster entspricht und kann so die Situation klassifizieren. Dabei ist es möglich Situationen als positiv oder negativ zu klassifizieren. Genauere Einteilungen dazwischen sind mit dem Perzeptron nur sehr schwer möglich.
Bei einem Kohonennetz läuft die Bewertung etwas anders ab. Indem vorher nur positive Situationen verwendet werden, um das Kohonennetz zu trainieren, ergeben sich Cluster in dem Kohonennetz. Nun kann geprüft werden, wo eine Erregung des Netzes durch eine neue noch unbekannte Situation stattfindet und dies kann mit den Clustern verglichen werden. Auch hier ist es leider nicht möglich zu sagen, welcher Parameter oder welche Parameter für einen Cluster verantwortlich sind, allerdings kann jeder Cluster als eigene Klasse von Situationen, die positiv sind, verstanden werden. Hierdurch sind die Möglichkeiten des Kohonennetzes deutlich umfangreicher

als die des Perzeptrons.
Ein nicht zu unterschätzender Vorteil der KNN liegt darin, dass sie während der Situationsbewertung die Möglichkeit des kontinuierlichen Lernens bieten. Das System könnte also vom Anwender lernen, während es bereits Situationen bewertet. Nachdem eine Situation bewertet wurde, wird eine Rückmeldung von dem Anwender eingeholt. Wenn dieser die Situation genau wie das KNN für positiv bzw. negativ hält findet kein weiterer Lernschritt statt. Sollte der Anwender allerdings nicht zu dem gleichen Schluss kommen wie das KNN, kann die Situation als neuer Trainingsdatensatz verwendet werden und eine kurze Trainingsphase eingeschoben werden, damit das System die Situation in Zukunft mit einer höheren Wahrscheinlichkeit so wie der Anwender bewertet. Dies setzt allerdings voraus, dass zwischen jeder Situationsbewertung ein geeignet großes Zeitfenster ist, um die zusätzliche Trainingsphase durchzuführen, ohne dass weitere Situationsbewertungen verzögert werden.

3.2.3. Naturgesetze

Grundsätzlich lassen sich die Naturgesetze als Domäne auffassen, allerdings mit einigen markanten Unterschieden in Herkunft und Anwendung. Daher wird im Folgenden auf die Unterschiede zwischen Domänen und den Naturgesetzen eingegangen.
Als Naturgesetze sollen nur solche Regeln aufgefasst werden, deren Richtigkeit weitgehend als bewiesen gilt, beispielsweise Newtons Gravitationsgesetz, der Energieerhaltungssatz oder das Ohmsche Gesetz [Bak09]. Auch Regeln, die auf mathematischen Sätzen basieren, sollen vereinfachend in die Menge der Naturgesetze aufgenommen werden, unter der Bedingung, dass ihre Gültigkeit in der betrachteten Wirklichkeit gesichert ist. Auf komplexere Abgrenzungen der Naturgesetze wird an dieser Stelle verzichtet, da dies keinen signifikanten Einfluss auf die Situationsbewertung haben wird.
Da die Naturgesetze auf jede Situation anwendbar sind, definieren sie keine eigenen Indikatoren, sondern beziehen sich lediglich auf die Indikatoren die jede Situation besitzt. Auch die Skalen der Naturgesetze haben eine gewisse Sonderstellung. Da es möglich ist an dieser Stelle zentrale Skalen für alle Situationen zu definieren, sollte dies auch gemacht werden. Domänen können zwar auf Skalen der Naturgesetze zurückgreifen, aber nicht anders herum, weil dies bedeuten würde, dass eine spezielle Domäne in jeder Situation angewendet werden muss. Hierdurch würde die spezielle Domäne auf den Rang der Naturgesetze gestellt werden, was vermieden werden sollte.
Für die Regeln der Naturgesetze gilt die gleiche Struktur wie bei allen anderen Domänen und auch die Gewichtung der Regeln bleibt erhalten, da es auch bei solchen Regeln sein kann, dass ihre Auswirkungen nur gering sind. Bezüglich der Sicherheit und der Elastizität von Regeln unterscheiden sich die Naturgesetze jedoch von den anderen Domänen.
Die Sicherheit des Ergebnisses eines Naturgesetzes sollte grundsätzlich nicht reduziert werden, außer in zwei Sonderfällen. Zum einen bei Regeln, die sich nicht deterministisch verhalten, wie zum Beispiel der radioaktive Zerfall von Atomen [JAW+00]

oder bei Regeln, auf die ungenaue Werte angewendet wurden. Beim ersten Fall ist davon auszugehen, dass solche Regeln eher selten von Relevanz sind und bei dem zweiten Fall ist vorausgesetzt, dass der ungenaue Wert als solcher identifiziert wurde. Wie ein ungenauer Wert erkannt wird, soll an dieser Stelle nicht weiter beschrieben werden, da dies später aufgegriffen wird.
Eine Elastizität ist bei Regeln der Naturgesetze nicht zwingend notwendig und wird daher hier auch nicht weiter betrachtet. Die Verwendung einer Elastizität bei Naturgesetzen wäre nur sinnvoll, wenn man davon ausgeht, dass die Messtechniken, mit denen die Parameter der Situation bestimmt werden, eine Ungenauigkeit besitzen. Eine solche Unterstellung ist sicher nicht abzuweisen, könnte allerdings auch mit geeigneten Skalen ausgeglichen werden.
Für die Naturgesetze ist die Verwendung eines KNN nicht sinnvoll, da davon ausgegangen wird, dass alle Regeln sowohl bekannt sind, als auch unveränderlich. Ein KNN würde hier womöglich durch ungeeignete Trainingsdaten falsche Schlüsse ziehen und somit die Grundlage der Situationsbewertung verfälschen.

3.2.4. Personen

Die Darstellung einer Person wird in Anlehnung an Jaakkola und Thalheim [JT14] beschrieben. Hier wird ein *user model* in Profile und Portfolios eingeteilt. Auf der Seite der Profile gibt es das Bildungsprofil, das Arbeitsprofil, das Persönlichkeitsprofil und das Sicherheitsprofil, während sich die Seite der Portfolios in Aufgaben, Einbeziehung des Nutzers, Zusammenarbeit und Restriktionen einteilt. Im Folgenden werden nicht alle Bereiche aufgegriffen, da dies den Rahmen dieser Arbeit sprengen würde. Die Person, für die die Situation bewertet wird, wird in dieser Arbeit ebenfalls als Persona bezeichnet, da es sich lediglich um eine verallgemeinerte Form der Person handelt. Der Aufbau dieser Persona wird in Abbildung 3.3 dargestellt.
Die verschiedenen Profile können durch Regeln, wie in den Domänen, abgebildet werden. Somit ist es möglich, dass für eine Persona nicht nur durch bestimmte Domänen Situationen bewertet werden, sondern auch durch eigene Erfahrungen und Vermutungen, die für die Persona hinterlegt sind. Es findet also nicht zwingend eine Rechtfertigung der Regeln statt. Zusätzlich kann eine Persona durch das Bildungs-, Arbeits- und Sicherheitsprofil bestimmte Domänen in die Situationsbewertung einbeziehen. Diese Domänen werden bei jeder Situationsbewertung aus der Perspektive dieser Persona verwendet, da sie fester Bestandteil der Persona sind.
Des Weiteren müssen durch eine Persona bestimmte Präferenzen und Grenzwerte bezüglich der Indikatoren definiert werden. Diese lassen sich als Teil des Persönlichkeitsprofils einstufen. Die Präferenzen und Grenzwerte, die durch das Persönlichkeitsprofil gegeben sind, dürfen sich allerdings nur auf die Indikatoren der Situation und auf solche, die durch Domänen der Profile definiert werden, beziehen.
Auch die Portfolios können Domänen einbeziehen und Präferenzen bzw. Grenzwerte vorgeben. Die Domänen, die durch Portfolios vorgegeben werden, sind grundsätzlich austauschbar, da sich zum Beispiel die Rolle oder die Berechtigungen einer Persona, welche Teil des Portfolios des Nutzers ist, dynamisch ändern können.

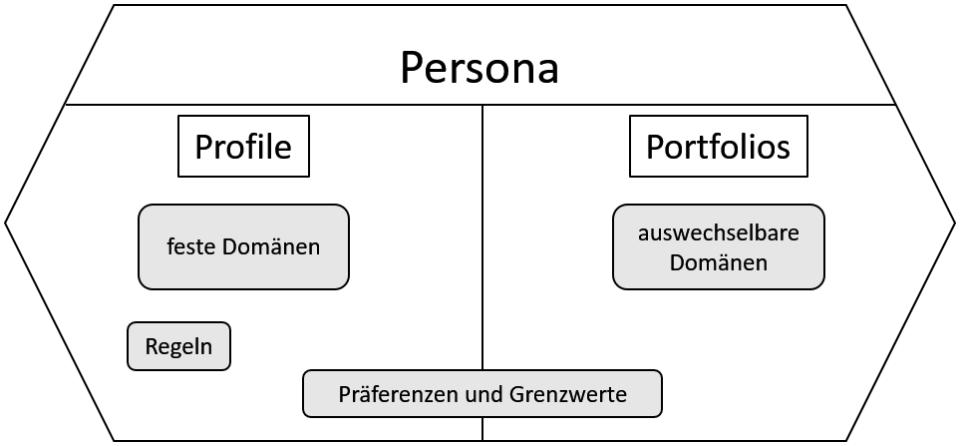


Abbildung 3.3.: Aufbau der Persona

Kulturen werden nicht explizit als Kultur definiert, sondern sollen durch das abstrakte Konstrukt der Domäne dargestellt werden. Bei Kulturen handelt es sich beispielsweise um solche Domänen, die durch das Profil einer Person vorgegeben werden.

3.3. Bewertungsfunktion

Nachdem im vorangegangenen Abschnitt die Strukturen für die Situationsbewertung definiert wurden, werde ich nun die eigentliche Situationsbewertung beschreiben. Diese wird in die drei Hauptschritte *Bewertung durch die Naturgesetze*, *Bewertung durch die Domänen* und *Bewertung durch die Persona* eingeteilt. Danach werden die Informationen zusammengefasst und an Hand der Präferenzen und Grenzwerte des Anwenders ausgewertet. Am Ende einer jeden Situationsbewertung steht die Reaktion, welche zu einer neuen Situation führt. Zu beachten ist, dass sogar dann eine Veränderung stattfinden kann, wenn nichts unternommen wird. Auch ohne das Einwirken des Betrachters ist davon auszugehen, dass sich einige Parameter der Situation verändern haben. Abbildung 3.4 zeigt den Ablauf der Situationsbewertung grafisch. Bei der Darstellung ist zu beachten, dass die Naturgesetze anders dargestellt werden, als die Domänen, da diese eine Sonderstellung haben, auf die im vorherigen Abschnitt bereits eingegangen wurde.

3.3.1. Bewertung durch die Naturgesetze

Am Anfang der Situationsbewertung wird die Situation durch die Naturgesetze bewertet, dass heißt, die Skalen und Regeln der Naturgesetze werden auf die Parameter der Situation angewendet. Die Naturgesetze werden vor allen anderen Domänen angewendet, da es sein kann, dass einige Domänen auf Skalen der Naturgesetze zurück-

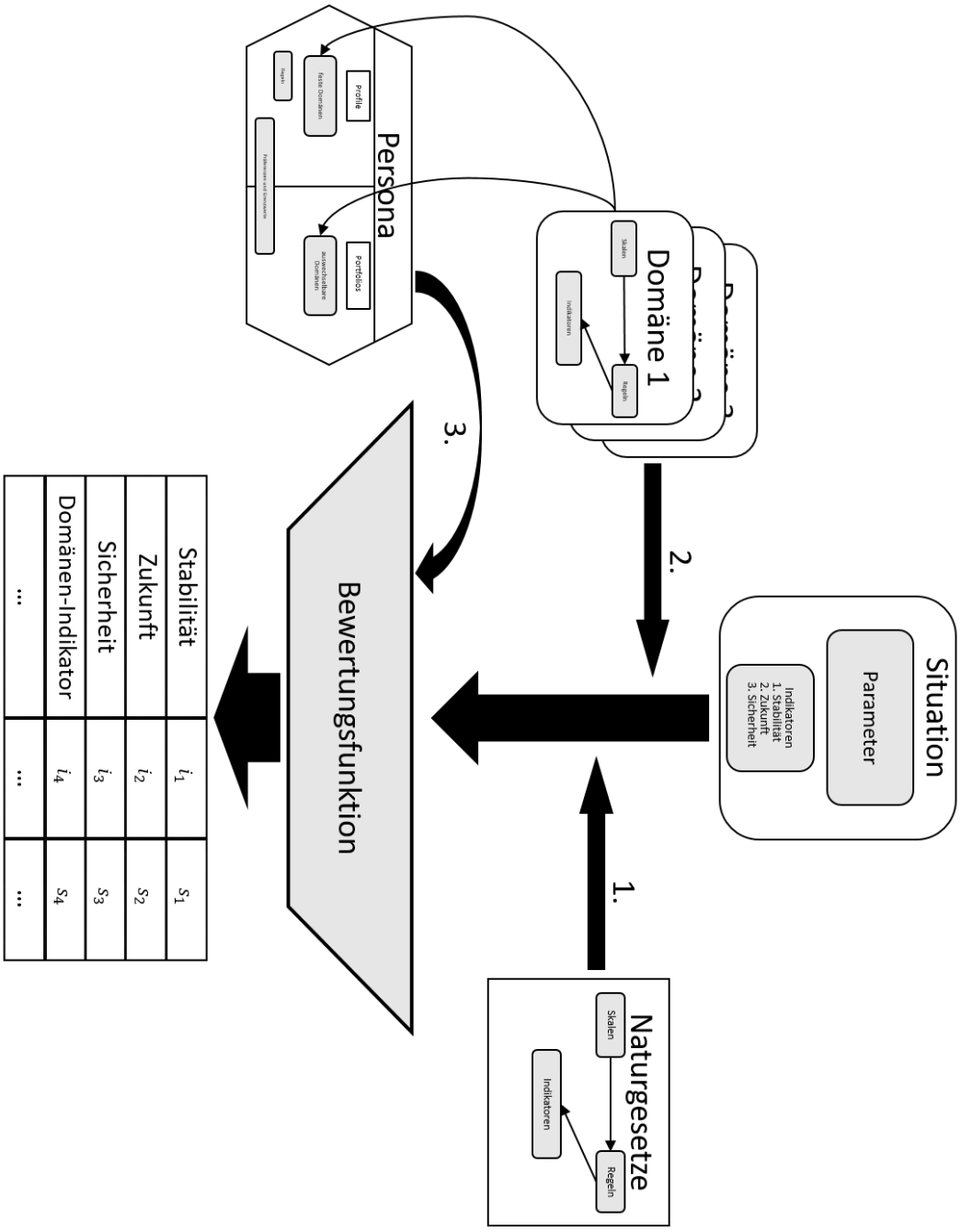


Abbildung 3.4.: Schematische Darstellung der Situationsbewertung

greifen. Daher ist es sinnvoll diese bereits vollständig auszuwerten. Die Bewertung kann in zwei Schritte gegliedert werden, wobei im ersten Schritt die Vorbereitung der Daten stattfindet, gefolgt von der Auswertung der aufbereiteten Daten.
Bei der Vorbereitung der Daten können verschiedene Schritte erfolgen um die Qualität der Daten zu erhöhen. Dieser Vorgang wird von Aggarwal als *Data Preparation* beschrieben [Agg15] und soll hier nicht weiter verfolgt werden. Es werden lediglich die Skalen auf die verschiedenen Parameter angewendet, weil davon ausgegangen wird, dass die Parameter bereits in der bestmöglichen Version vorliegen.
Nehmen wir an, E ist eine Menge von Parametern, die die Situation beschreiben und es gibt eine Menge von Skalen der Form $\mathfrak{S} : E \to A$, die für die Naturgesetze geeignet definiert wurden und die Parameter der Situation (E) auf eine Menge von Zahlen (A) abbilden. Dann lassen sich alle für die Naturgesetze relevanten Werte als $\mathfrak{S}(e)$ darstellen, wobei $e \subseteq E$ gilt.
Nachdem alle für die Naturgesetze relevanten Parameter in Werte überführt wurden, können die Regeln der Naturgesetze auf eben diese Werte angewendet werden. Wir können die Bedingung einer Regel als eine Abbildung $\mathfrak{C} : \mathfrak{S}_1 \times ... \times \mathfrak{S}_n \to \{true, false\}$ schreiben. Diese bildet die Variablen, also die Bilder vorher festgelegter Skalen, auf einen Wahrheitswert ab, der entscheidet, ob die Bedingung erfüllt ist oder nicht. Bei stufenweisen Regeln wird nicht auf einen Wahrheitswert sondern auf eine Zahl abgebildet, welche angibt wie stark die Bedingung ausgeprägt ist. Hier muss die Menge des Bildes der Bedingung speziell definiert werden.
Die Regeln selbst lassen sich nun ebenfalls als Abbildung $\mathfrak{L} : \{true, false\} \to (i, w, s)$ auffassen. Bei stufenweisen Regeln gilt auch hier, dass die Wahrheitswerte durch eine Menge von Zahlen zu ersetzen sind. Das Bild einer Regel, also die Schlussfolgerung, ist ein Tupel bestehend aus drei Elementen. Das erste Element (i) gibt die Ausprägung für den Indikator an, während das zweite Element (w) die Gewichtung dieser Schlussfolgerung für das Gesamtergebnis angibt und das dritte Element (s) gibt die Sicherheit der Schlussfolgerung an. Auf Regeln, die auf mehr als einen Indikator Auswirkung haben wird hier aus Platzgründen nicht eingegangen. Eine Verwendung solcher Regeln ist aber durchaus sinnvoll.

3.3.2. Bewertung durch die Domänen

Die Bewertung durch die Domänen läuft im Prinzip genau wie die Bewertung durch die Naturgesetze ab, da eine Domäne lediglich eine allgemeinere Form darstellt. Allerdings kann durch eine Domäne auf Skalen der Naturgesetze zurückgegriffen werden, da die Skalen der Naturgesetze auf jede Situation angewendet werden. Eine Domäne hingegen wird durch die Persona, für die die Situation bewertet wird, gewählt. Das bedeutet, dass nicht jede Domäne auf jede Situation angewendet werden muss, obgleich dies möglich wäre. Die Domänen können sich sogar bei einer Persona zwischen verschiedenen Situationen unterscheiden. Dies passiert immer, wenn eine Persona eine Situation in einer anderen Rolle oder ähnlichem betrachtet, da auch durch die Rolle einer Persona Domänen vorgegeben werden können.

Nachdem die Parameter der Situation mit Hilfe der Skalen der Domänen und Naturgesetze überführt wurden, können alle Regeln der Domänen ausgewertet werden, genau wie dies der Fall bei den Naturgesetzen ist.

3.3.3. Bewertung durch die Persona

Die Bewertung durch die Persona darf nicht verwechselt werden mit der Bewertung durch die Person. Letztere wird durch die natürliche Person durchgeführt und kann auf Grund mangelnder Informationen nicht technisch abgebildet werden, weil es sich um eine extrem große Menge von abzubildenden Informationen handelt und nicht alle Arten von Informationen bekannt sind. Lediglich die Bewertung durch die Persona kann technisch abgebildet werden, da die Persona von einer Person abstrahiert wurde und dabei auf eine darstellbare Menge von Regeln und Eigenschaften reduziert wurde.
Die Anwendung der Regeln geschieht genau wie bei den Naturgesetzen und den restlichen Domänen. Der Unterschied liegt also einzig im Ursprung der Regeln, welcher vorher bereits erklärt wurde.
Da diese Bewertung auf der Intuition einer Person aufbaut sollten die Ergebnisse vorsichtig interpretiert werden. Es ist wahrscheinlich sogar sinnvoll die gesamte Situation einmal ohne und einmal mit den Ergebnissen dieses Teils zu bewerten, um die Ergebnisse zu vergleichen. Wenn diese Ergebnisse deutlich von einander abweichen, sollte zu erst geprüft werden, ob die Abweichungen an fehlerhaften Schlussfolgerungen der Persona liegen.

3.3.4. Zusammenfassung der Ergebnisse und Auswahl der relevanten Informationen

Nachdem alle Auswertungen vorgenommen wurden, können die Teilergebnisse aggregiert werden. Beim Zusammenfassen der Teilergebnisse wird die Struktur dieser verändert, da die Ergebnisse nur noch aus Tupeln der Form (Indikator-Wert, Sicherheit) bestehen. Diese Veränderung entsteht auf Grund der Anwendung der Gewichtungen. Da die Gewichtungen beim Aggregieren auf die jeweiligen Indikator-Werte und Sicherheiten angewendet werden, entfallen diese.
Die Ergebnisse für die verschiedenen Indikatoren lassen sich dann in einer zweidimensionalen Ergebnismatrix darstellen. Die Ergebnismatrix hat generell zwei Spalten, wobei die erste Spalte die konkrete Ausprägung des Indikators angibt und die zweite Spalte die Sicherheit des jeweils berechneten Ergebnisses. Jede Zeile der Ergebnismatrix entspricht also einem aggregierten Ergebnis bzw. steht somit für genau einen Indikator.
Angenommen, bei der Bewertung für den Indikator *Sicherheit*, welcher an m-ter Stelle in der Ergebnismatrix stehen soll, wurden insgesamt n Teilergebnisse berechnet. Dann haben diese alle die Form $(i_{m,j}, w_{m,j}, s_{m,j})$ mit $1 \leq j \leq n$.
In Abbildung 3.5 werden die Formeln dargestellt, mit denen der Indikator Sicherheit und analog auch alle anderen Indikatoren berechnet werden. Die Ergebnismatrix,

$$
\begin{array}{l}
\\
Indikator(m-1) \\
Sicherheit(m) \\
Indikator(m+1) \\
\\
\end{array}
\begin{pmatrix}
\vdots & \vdots \\
\sum_{j=1}^{n} i_{m-1,j} \cdot W_{m-1,j} & \sum_{j=1}^{n} s_{m-1,j} \cdot W_{m-1,j} \\
\sum_{j=1}^{n} i_{m,j} \cdot W_{m,j} & \sum_{j=1}^{n} s_{m,j} \cdot W_{m,j} \\
\sum_{j=1}^{n} i_{m+1,j} \cdot W_{m+1,j} & \sum_{j=1}^{n} s_{m+1,j} \cdot W_{m+1,j} \\
\vdots & \vdots
\end{pmatrix}
$$

Für die Gewichtungen gilt folgendes:

$$w_m = \sum_{j=1}^{n} w_{m,j} \qquad W_{m,j} = \frac{w_{m,j}}{w_m}$$

Abbildung 3.5.: Zusammenfügen der Teilergebnisse und Erstellung der Ergebnismatrix

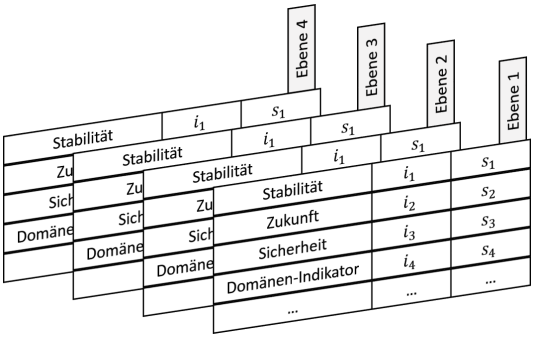
welche in Abbildung 3.6a schematisch dargestellt wird, könnte um zusätzliche Dimensionen erweitert werden, um weitere Informationen zu den Ergebnissen bzw. über die Aussagefähigkeit der Ergebnisse, darzustellen. Zum Beispiel könnte eine dritte Dimension die Ergebnisse nach der Sicherheit, wie in Abbildung 3.6b dargestellt, einteilen. Hier würden in jeder der l Ebene der dritten Dimension alle Teilergebnisse aufgenommen werden, deren Sicherheit über einem festen Wert x_k, mit $1 \leq k \leq l$, liegt. Dies hätte den Vorteil, dass ein Nutzer nach der Bewertung entscheiden kann, wie sicher die Informationen sein müssen, damit sie betrachtet werden. Die Möglichkeiten des Nutzers, die bereitgestellte Bewertung zu hinterfragen und geeignete Werte zu wählen, würde damit verbessert werden.
Zum Abschluss können die Präferenzen der Person auf die Ergebnismatrix angewendet werden. Nehmen wir an, ein Nutzer betrachtet die Ergebnismatrix aus Abbildung 3.5 und er hat einen oberen und unteren Grenzwert für den Indikator Sicherheit festgelegt. Wenn nun der Wert des Indikators Sicherheit, also der Eintrag in der ersten Spalte bei Sicherheit innerhalb dieser Grenzwerte liegt, dann ist es aus Sicht des Systems nicht sinnvoll hier eine Aktion, mit dem Ziel die Situation bezüglich ihrer Sicherheit zu verändern, auszuführen. Sollte der Indikator allerdings einen dieser Grenzwerte überschreiten wird hier ein Handeln für sinnvoll erachtet.

3.3.5. Reaktion

Der Nutzer selbst hat die Entscheidung zu Handeln oder nicht, da das Situationsbewertungssystem lediglich informativen Charakter hat. Es wird keine Aktion automatisch ausgewählt, um eine möglicherweise offensichtlich fehlerhafte Bewertung nicht zum Anlass eines Einschreitens zu nehmen. Neben der Gefahr der fehlerhaften

Stabilität	i_1	s_1
Zukunft	i_2	s_2
Sicherheit	i_3	s_3
Domänen-Indikator	i_4	s_4
...	...	...

(a) zweidimensional



(b) dreidimensional

Abbildung 3.6.: Darstellung der zwei- bzw. dreidimensionalen Ergebnismatrix

Bewertung spielen in vielen Ländern auch die jeweiligen Gesetze eine wichtige Rolle in Bezug auf die Reaktionsmöglichkeiten von Systemen. Es gibt immer noch viele ungeklärte Fragen bezüglich der Haftung von sogenannten autonomen Systemen, wie sie zum Beispiel in Autos verbaut werden [LTL]. In der rechtlichen Diskussion über autonome Fahrzeuge gilt in Europa immer noch der Grundsatz, dass vollständig autonome Fahrzeuge gegen das Wiener Übereinkommen von 1968 und auch gegen die Überarbeitung von 2014 verstoßen [JM15]. Daher ist es nötig am Ende der Situationsbewertung eine klare Grenze zu ziehen und keine Aktion mehr auszuführen, wenn gleich die Informationen theoretisch von einem weiteren System direkt genutzt werden können, um eigenständig auf die Situation zu reagieren. Die Probleme von Systemen, die eigenständig Entscheidungen treffen, werden hier nicht im Detail ausgeführt, es sei auf die Arbeit der Daimler und Benz Stiftung verwiesen [MD15], die sich ausgiebig mit der Thematik am Beispiel autonomer Fahrzeuge beschäftigt. Die klaren Grenzen helfen zusätzlich die Modularität von großen Systemen zu verbessern, da die Messung der Situation und die Reaktion von beliebigen separaten Systemen durchgeführt werden können. Bei sauber definierten Schnittstellen entsteht hier die Möglichkeit des einfachen Austauschs von Teilkomponenten.
Nachdem der Nutzer eine Entscheidung getroffen hat, also reagiert hat, indem er eine Aktion ausführt oder indem er keine Aktion ausführt, besteht noch die Chance aus dem Verhalten des Nutzers zu lernen, also die Situationsbewertung zu verbessern. Wenn der Ratschlag des Systems befolgt wurde ist davon auszugehen, dass die Situation korrekt bewertet wurde. Ist dies jedoch nicht der Fall und es lässt sich eindeutig bestimmen, welche Domäne eine fehlerhafte oder unpassende Einschätzung lieferte, dann kann diese Domäne angepasst werden. Falls die betroffene Domäne durch ein KNN abgebildet wird ist dieses in Form eines neuen Trainingsschrittes möglich, wobei die gerade bewertete Situation als Trainingsdatensatz dient. Bei allen anderen Domänen ist eine derartige Anpassung nicht so einfach möglich. Um den Umfang dieser Arbeit nicht zu sprengen wird auf diese Anpassung nicht mehr eingegangen.

3.3.6. Verhalten bei falschen oder fehlenden Informationen

Bei der Bewertung von Situationen aus einer von der echten Wirklichkeit abgeleiteten fiktiven Wirklichkeit ist davon auszugehen, dass niemals alle Informationen vollständig vorhanden sein werden, wohingegen bei anderen fiktiven Wirklichkeiten eher davon auszugehen ist, dass alle Informationen über eine Situation vorhanden sind. Da es allerdings in beiden Fällen möglich ist, dass die Informationen fehlen oder falsch sind, muss auch darauf eingegangen werden, was in diesen Fällen zu tun ist. Ich werde lediglich auf fehlende Informationen eingehen, da falsche Informationen schwer bis unmöglich zu identifizieren sind. Wenn eine Information fehlt kann man dies als Sonderfall ansehen, da grundlegend davon ausgegangen wird, dass alle Informationen vorhanden sind. Nach Fischer und Grodzinsky kann man auch von einer Ausnahme sprechen [FG93]. Eine Einteilung der Fehler wie bei Fischer und Grodzinsky nach Hardware-, Software- und Benutzerfehlern werde ich nicht vornehmen, da dies im Folgenden keinen Einfluss darauf haben wird, wie mit der Ausnahme zu verfahren ist, auch wenn dies theoretisch möglich wäre.
Es gibt zwei Modelle, wie mit Ausnahmen verfahren werden kann. Einerseits das Terminationsmodell bei dem die Komponente, in der die Ausnahme auftrat, abgeschaltet wird. Andererseits das Wiederanlaufmodell bei dem die Komponente lediglich unterbrochen bzw. angehalten wird und versucht wird die Ausnahme zu behandeln, bevor die Komponente weiter arbeiten kann [Kla12]. Unter Ausnahme ist hierbei die fehlende Information zu verstehen. Ich werde auf einen Ansatz des Wiederanlaufmodells zurückgreifen, da ein Terminieren der Situationsbewertung wegen einer fehlenden Information eine unverhältnismäßige Reaktion wäre. Der schlimmste Fall tritt ein, wenn es keine Möglichkeit gibt die fehlende Information zu erhalten, dann würden die betroffenen Regeln nicht ausgewertet werden können, was vergleichbar mit dem Terminationsmodell ist. Um dies zu vermeiden gibt es zwei Möglichkeiten. Zum einen können Regeln entworfen werden, die bei fehlenden Informationen immer noch auswertbar bleiben und deren Ergebnisse dann zum Beispiel eine geringere Sicherheit aufweisen. Zum anderen könnten die Skalen erweitert werden, damit die fehlenden Informationen auf der Grundlage von anderen Informationen geschätzt werden können. Hier bleibt zu beachten, dass dies zu potenziell unsicheren Ergebnissen führt, was ebenfalls in die Ergebnismatrix eingehen sollte. Bei der Verwendung solcher Skalen muss das System also dahingehend erweitert werden, dass Unsicherheiten, die durch Skalen entstehen, ebenfalls weiter gegeben werden können. Dies ist aktuell nicht möglich.
Fehlerhafte Informationen lassen sich leider meistens nicht auf einfachem Wege erkennen, sodass in diesem Fall auch keine Ausnahmesituation erkannt werden kann. Hier bleibt lediglich die Möglichkeit in den Skalen und Regeln Schranken einzubauen, um Parameter, die außerhalb des vermutlich zulässigen Bereiches liegen, nicht mehr zu betrachten.

3.4. Erweiterungen

Zum Abschluss dieses Kapitels werden kurz einige nützliche Erweiterungen aufgegriffen, die in der Situationsbewertung aus Platzmangel nicht betrachtet wurden. Die hier beschriebenen Erweiterungen sind jedoch auch nur ein kleiner Ausschnitt, da es in der Natur der Sache liegt, dass eine Situationsbewertung, vor allem bei offenen Welten, niemals vollständig sein wird.

3.4.1. Dynamik

In der Situationsbewertung wird aktuell keine Dynamik explizit erfasst. Es ist zwar prinzipiell möglich über die Parameter auch Informationen über vorherige Situationen zu erfassen, allerdings ist dieses Verfahren nur begrenzt umsetzbar, da die Anzahl der Parameter extrem steigen würde und die Situationsbewertung somit nach jeder Situation aufwendiger wird. Hier wäre eine Erweiterung von Interesse, damit explizit erfasst wird, wie es zu der aktuellen Situation kam, oder wie sich die Situation von einer bestimmten Situation aus entwickelt hat. Dieser Ansatz wird in Abbildung 3.7 verdeutlicht. Der Vorteil wäre unter anderem, dass in einer Situation, die schon einmal vorkam, anders reagiert werden kann als bei dem Mal zuvor, wenn festgestellt wurde, dass die Reaktion beim vorherigen Auftreten nicht sinnvoll war. Auch viele Eigenschaften einer Situation bezüglich der Person können erst mit einer Historie, in der ein Ablauf alter Situationen und Aktionen gespeichert ist, vernünftig erfasst werden. Vor allem Emotionen und die menschliche Intuition benötigen Informationen darüber, was vorher passiert ist. Aus der technischen Perspektive wäre es sinnvoll alte Situationen, die bereits ausgewertet wurden, nicht jedes mal erneut auszuwerten, sondern direkt mit den Indikatoren der alten Situationen zu arbeiten und höchstens abstrakte Informationen über eine Situation zu speichern. Damit würde der Rechenaufwand der Situationsbewertung bei weitem weniger stark ansteigen.

3.4.2. Umfangreichere Persona

Bei der Persona der Situationsbewertung sind noch viele Erweiterungen möglich. Aktuell müsste für fast jede Person eine eigene Persona entworfen werden, was einen enormen Zeitaufwand darstellt. Daher wäre es hier sinnvoll die Persona weiter zu zerlegen, um einen modularen Aufbau zu erhalten. Eine Persona könnte nach Jaakkola und Thalheim [JT14] gegliedert werden. Diese Gliederung wurde in Ansätzen bereits verwendet, um die Persona der Situationsbewertung zu strukturieren. Allerdings ist die in dieser Arbeit verwendete Struktur noch sehr wenig präzise. Da eine vollständige Übertragung der Gliederung in die Situationsbewertung allerdings den Rahmen dieser Arbeit gesprengt hätte wurde darauf verzichtet. Innerhalb der Portfolios besteht die Möglichkeit eine höhere Modularität aufzubauen, indem zum Beispiel Rollen oder Aufgaben abstrakt definiert werden und somit immer wieder verwendet werden können, wenn sie für eine Persona relevant werden.

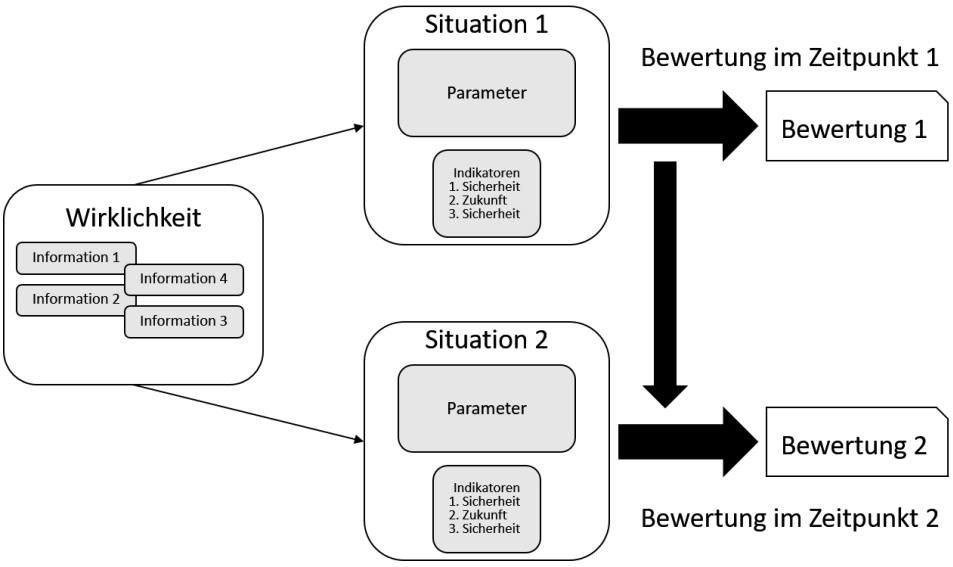


Abbildung 3.7.: Situationsbewertung mit Bezug auf vergangene Situationen

3.4.3. Knowledge Bases

Neben den internen Erweiterungen bezüglich Dynamik und Persona könnte durch die Verwendung von sogenannten Knowledge Bases die Aussagefähigkeit der Situationsbewertung verbessert werden. Als Wissen *(engl. knowledge)* wird hierbei jede Information verstanden, bei der angenommen wird, dass sie korrekt sei und bei der davon auszugehen ist, dass sie für die Situationsbewertung von Interesse sein kann. Dieses Wissen wird weiter unten noch unterteilt.
Zum einen kann dies erreicht werden durch den Aufbau einer eigenen Wissenssammlung (Abbildung 3.8), worauf später kurz eingegangen wird, oder durch das Anbinden von Wissensdatenbanken wie DBpedia [ABK+07]. Vor allem durch das Anbinden solcher Wissensdatenbanken kann die Qualität der Situationsbewertungen verbessert werden, ohne großen Aufwand für die Aufbereitung von Wissen zu verursachen. Je nach Anwendungsbereich der Situationsbewertung ist es ebenfalls sinnvoll nach speziellen Wissensdatenbanken in der Linked Open Data Cloud [PSA14] zu suchen und diese zu verwenden. Durch die Wissensdatenbanken könnten vor allem Informationen über die Situation geprüft werden, sofern diese als Fakten aufgefasst werden können.
Eine Wissensdatenbank speziell für die Situationsbewertung sollte die folgenden Informationen speichern können:

- Fakten - Werden entweder lokal abgespeichert oder bei Bedarf aus anderen Wissensdatenbanken in Form eines Tripels abgefragt. Aus Fakten können Regeln abgeleitet werden oder auch Parameter auf ihre Richtigkeit geprüft werden.

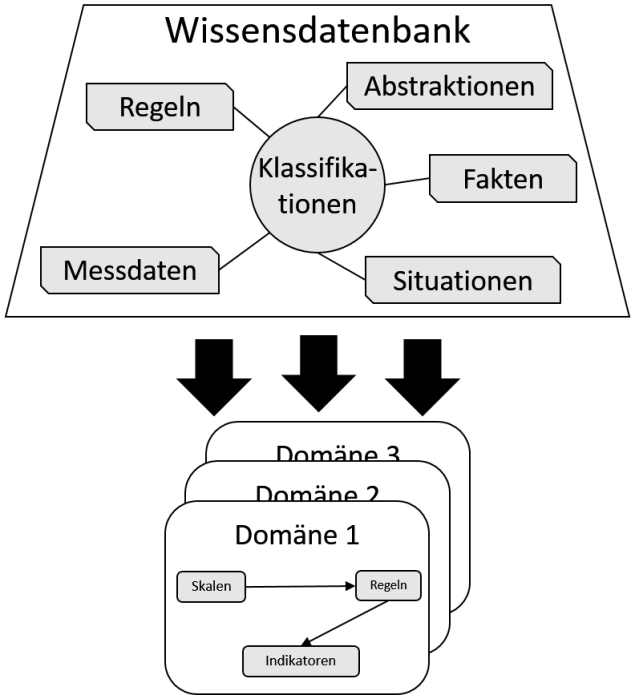


Abbildung 3.8.: Knowledge Base für die Situationsbewertung

- Regeln - Diese haben die gleiche Struktur, wie die Regeln in den Domänen und können somit in beliebige Domänen direkt übernommen werden.
- Situationen - Nachdem eine Situation bewertet wurde kann sie verwendet werden, um ein KNN zu trainieren.
- Messdaten - Aus Messdaten zu bestimmten Parametern können Bereiche für die Ausprägungen der Parameter abgeleitet werden. Liegt eine Ausprägung innerhalb eines so identifizierten Bereichs, kann zumindest gesagt werden, dass die Ausprägung möglich ist.
- Abstraktionen - Mit Abstraktionen können Regeln auf höheren Ebenen definiert werden (Bsp. Eine Schaufel ist ein Werkzeug - ein Werkzeug ist Realkapital - Realkapital ist ein materieller Gegenstand). Nun kann eine Regel für Werkzeuge, Realkapital oder materielle Gegenstände definiert werden anstatt für alle Werkzeuge einzeln.
- Klassifikation - Alle Einträge sollten klassifiziert sein, damit sie einer Domäne zugeordnet werden können, dabei kann ein Objekt in mehreren Klassen enthalten sein

Durch diese Wissensdatenbank wäre es möglich die Regel für die Bewertung zu

generieren und nicht manuell zu entwerfen. Zudem können Parameter geprüft werden und notfalls als sehr unwahrscheinlich markiert werden.

4. Anwendung

Im Anschluss an den Aufbau des Frameworks zur Situationsbewertung wird im folgenden Kapitel der Einsatz des Frameworks an einem Beispiel gezeigt. Hierbei wird ein fiktives Storage-System gewählt. Es ist zu beachten, dass das Storage-System und alle verwendeten Domänen auf einige wenige Merkmale reduziert wurden, um die Bewertung übersichtlich zu halten. Des Weiteren sind jegliche Werte, die in den Regeln verwendet werden, beliebig gewählt und dienen nur der Veranschaulichung. Auf die Verwendung eines KNN zur Bewertung wird hier aus Platzgründen nicht eingegangen.
Als erstes werden die betrachtete Wirklichkeit und die drei zu bewertenden Situationen beschrieben. Im Anschluss werden Naturgesetze und die relevanten Domänen Storage-System, Netzwerkkommunikation und Datenschutz & Datensicherheit dargestellt, gefolgt von der Beschreibung einer Persona. Zum Schluss des Kapitels wird die konkrete Bewertung durchgeführt.

4.1. Wirklichkeit und Situationen

Durch die starke Eingrenzung auf das Storage-System in dem eigene Grundsätze gelten, kann die betrachtete Wirklichkeit als fiktiv angesehen werden. Diese Wirklichkeit ist zwar ein Ausschnitt der echten Wirklichkeit, allerdings um alle Informationen bereinigt, die die echte Wirklichkeit ausmachen. Ich gehe davon aus, dass das Storage-System über die im Folgenden dargestellten Parameter vollständig beschrieben werden kann. Des Weiteren wird angenommen, dass keine der Soft- oder Hardwarekomponenten, aus denen das Storage-System besteht, abstürzen oder ausfallen können. Somit kann die Wirklichkeit nicht plötzlich zerbrechen, also vollständig beendet werden. Die Parameter unterteilen sich in Storage-System und Netzanbindung. Die drei Beispielsituationen sind so gewählt, dass die Ergebnisse der Situationsbewertung möglichst verschieden sind und auf viele Aspekte eingegangen werden kann.

4.1.1. Storage-System und Netzanbindung

Die Kernkomponente des Storage-Systems wird immer als a bezeichnet, während die Netzanbindung des Storage-Systems als b bezeichnet wird. Zusätzlich werden alle externen Objekte, die das Storage-System nutzen können, als c gekennzeichnet.

Storage-System

1. Netto-Speicher $<< net\_storage, a, q, p >>$
   Gibt an, wie viele Speichereinheiten des Systems a Anwendungen oder Nutzern bereitgestellt werden können, wobei p die Anzahl der Speichereinheiten ist. q ist der Parameter Brutto-Speicher.

2. belegter Speicher $<< net\_storage\_in\_use, a, q, p >>$
   Gibt an, wie viel Speichereinheiten (Netto) des Systems a belegt sind, wobei p die Anzahl der belegten Speichereinheiten ist. q ist der Parameter Netto-Speicher.

3. Brutto-Speicher $<< gross\_storage, a, p >>$
   Gibt an, wie viel Speichereinheiten insgesamt im System a verbaut wurden, wobei p die Anzahl der Speichereinheiten ist.

4. max. Lesegeschwindigkeit $<< read\_rate\_max, a, p >>$
   Gibt an, wie hoch die Lesegeschwindigkeit des Systems a maximal sein kann, wobei p die Geschwindigkeit als Speichereinheit pro Zeiteinheit ist.

5. aktuelle Lesegeschwindigkeit $<< read\_rate, a, q, p >>$
   Gibt an, wie hoch die aktuell genutzte Lesegeschwindigkeit des Systems a ist, wobei p die Geschwindigkeit als Speichereinheit pro Zeiteinheit ist. q ist der Parameter max. Lesegeschwindigkeit.

6. max. Schreibgeschwindigkeit $<< write\_rate\_max, a, p >>$
   Gibt an, wie hoch die Schreibgeschwindigkeit des Systems a maximal sein kann, wobei p die Geschwindigkeit als Speichereinheit pro Zeiteinheit ist.

7. aktuelle Schreibgeschwindigkeit $<< write\_rate, a, q, p >>$
   Gibt an, wie hoch die aktuell genutzte Schreibgeschwindigkeit des Systems a ist, wobei p die Geschwindigkeit als Speichereinheit pro Zeiteinheit ist. q ist der Parameter max. Schreibgeschwindigkeit.

8. Puffer-Größe $<< buffer, a, p >>$
   Gibt an, wie groß der Puffer für Schreib- und Lese-Operationen des Systems a ist, wobei p die Anzahl an Speichereinheiten ist.

9. belegte Puffer-Menge $<< buffer\_usage, a, q, p >>$
   Gibt an, wie groß die aktuelle Auslastung des Puffers des Systems a ist, wobei p die Anzahl der belegten Speichereinheiten ist. q ist der Parameter Puffer-Größe.

10. max. Verbindungsanzahl $<< connections, a, p >>$
    Gibt an, wie viele Verbindungen Anwendungen zu dem System a maximal aufbauen können, wobei p die Anzahl der Verbindungen ist.

11. aktuell aktive Verbindungen $<< connections\_in\_use, a, X, q, p >>$
    Gibt an, wie viele Verbindungen aktuell zum System a von allen Anwendungen aus X belegt sind, wobei p die Anzahl der Verbindungen ist. q ist der Parameter max. Verbindungsanzahl.

12. Anzahl der Server-Knoten $<< nodes, a, p >>$
    Gibt an, aus wie vielen Knoten das System a besteht, wobei p die Anzahl der Knoten ist.

13. Anzahl der Verbindungen zwischen Knoten $<< node\_connections, a, q, p >>$
    Gibt an, wie viele Verbindungen zwischen allen Knoten des Systems a existieren und genutzt werden, wobei p die Anzahl der Verbindungen ist. q ist der Parameter Anzahl der Server-Knoten.

Netzanbindung

1. max. Upload-Geschwindigkeit $<< upstream, b, p >>$
   Gibt an, wie hoch die Upload-Geschwindigkeit des Systems b maximal sein kann, wobei p die Geschwindigkeit als Speichereinheit pro Zeiteinheit ist.

2. aktuelle Upload-Geschwindigkeit $<< upstream\_usage, b, q, p >>$
   Gibt an, wie hoch die aktuell genutzte Upload-Geschwindigkeit des Systems b ist, wobei p die Geschwindigkeit als Speichereinheit pro Zeiteinheit ist. q ist der Parameter max. Upload-Geschwindigkeit.

3. max. Download-Geschwindigkeit $<< downstream, b, p >>$
   Gibt an, wie hoch die Download-Geschwindigkeit des Systems b maximal sein kann, wobei p die Geschwindigkeit als Speichereinheit pro Zeiteinheit ist.

4. aktuelle Download-Geschwindigkeit $<< downstream\_usage, b, q, p >>$
   Gibt an, wie hoch die aktuell genutzte Download-Geschwindigkeit des Systems b ist, wobei p die Geschwindigkeit als Speichereinheit pro Zeiteinheit ist. q ist der Parameter max. Download-Geschwindigkeit.

5. Latenz $<< ping, b, c, p >>$
   Gibt an, wie groß die Verzögerungen von Zugriffen zwischen b und c sind, wobei p die Anzahl der Zeiteinheiten ist.

6. max. Verbindungsanzahl $<< connections, b, p >>$
   Gibt an, wie viele Verbindungen Clients zu dem System b maximal aufbauen können, wobei p die Anzahl der Verbindungen ist.

7. aktuell aktive Verbindungen $<< connections\_in\_use, b, Y, q, p >>$
   Gibt an, wie viele Verbindungen aktuell zum System b von allen Clients aus Y belegt sind, wobei p die Anzahl der Verbindungen ist. q ist der Parameter max. Verbindungsanzahl.

4.1.2. Situation 1-3

Es werden drei Situationen, der vorher beschriebenen Wirklichkeit, bewertet. Die Parameter der Situationen sind im Anhang in den Tabellen A.1, A.2 und A.3 aufgelistet. In allen Situationen gelten für a, b und c die Festlegungen aus dem vorherigen Abschnitt. Wohingegen die beiden Mengen X und Y in den jeweiligen Situationen unterschiedlich gegeben sind. Da die konkreten Elemente von X sowie von Y für die Situationsbewertung in diesem Fall nicht relevant sind, wird hier nur auf die jeweiligen Mächtigkeiten eingegangen. Diese sind wie folgt:

1. Situation: $|X| = 15$ und $|Y| = 10$
2. Situation: $|X| = 30$ und $|Y| = 15$
3. Situation: $|X| = 15$ und $|Y| = 16$

4.2. Naturgesetze und Domänen

Nachdem, sowohl die Wirklichkeit als auch drei Beispiel Situationen beschrieben wurden, wird nun auf die für den Anwendungsfall interessanten Naturgesetze und Domänen eingegangen. Natürlich lassen sich neben den drei beschriebenen Domänen noch weitere Domänen, wie zum Beispiel Unternehmensorganisation, heranziehen, um die Situation zu bewerten. Allerdings wird hier nur auf die drei am Anfang dieses Kapitel erwähnten Domänen eingegangen.
Im Folgenden wird davon ausgegangen, dass die jeweiligen Sicherheitswerte und Gewichtungen der Regeln immer 1 entsprechen und diese somit im Anwendungsfall vernachlässigt werden können.

4.2.1. Naturgesetze

Auf die Naturgesetze wird bei dieser Anwendung nicht eingegangen, da die Situation extra so eingegrenzt wurde, dass keine Parameter erfasst werden, die durch Naturgesetze ausgewertet werden können.

4.2.2. Storage-System

Die Domäne für Storage-Systeme wird durch die folgenden Regeln und Indikatoren dargestellt. Neben einigen Regeln, die sicherstellen, dass Maximalwerte nicht überschritten werden, gibt es auch Regeln, die die Ausfallsicherheit und ähnliches prüfen.

Regeln

1. Lese- und Schreibgeschwindigkeit sollten immer unter 90 % der maximalen Lese- und Schreibgeschwindigkeit bleiben.
2. Der Puffer sollte mindestens zu 30% leer sein.

3. Der freie Speicher sollte nicht weniger als 10% des Netto-Speichers betragen.
4. Der Netto-Speicher sollte 70% - 80% des Brutto-Speichers betragen.
5. Es sollten möglichst 10% der Verbindungen frei sein.
6. Die Vernetzung des Storage-Systems sollte möglichst hoch sein, d.h. jeder Knoten sollte im Durchschnitt mit mindestens 2 anderen verbunden sein.
7. Jede Anwendung kann nur maximal drei Verbindungen gleichzeitig nutzen.

Die Übersetzung der hier aufgeführten Regeln befindet sich im Anhang (Tabelle A.5).

Verfügbarkeit Die Domäne von Storage-Systemen wird durch den Indikator Verfügbarkeit repräsentiert, welcher darstellen soll, wie weit die Verfügbarkeit des Systems gesichert ist. Der Indikator kann sich dabei zwischen 0 und 1 bewegen, wobei 0 bedeutet, dass das System komplett ausgefallen ist und 1 angibt, dass ein System vollständig verfügbar ist.

4.2.3. Netzwerkkommunikation

Zusätzlich zu der Domäne für Storage-Systeme wird die Domäne für Netzwerkkommunikation benötigt, da das Storage-System über eine Netzwerkanbindung verfügt. Die Domäne über Netzwerkkommunikation besteht aus Regeln, die denen des eigentlichen Storage-Systems sehr ähnlich sind, da die Netzanbindung des Storage-Systems in unserem Fall lediglich als Zugang zum eigentlich Storage-System dient.

Regeln

1. Up- und Download-Geschwindigkeit sollten immer unter 90% der maximalen Up- und Download-Geschwindigkeit bleiben.
2. Die Latenz sollte niemals höher als 10 Zeiteinheiten sein.
3. Es sollten möglichst 10% der Verbindungen frei sein.
4. Jeder Client kann maximal drei Verbindungen gleichzeitig nutzen.

Die Übersetzung der hier aufgeführten Regeln befindet sich im Anhang (Tabelle A.7).

4.2.4. Datenschutz und Datensicherheit

Die Domäne Datenschutz und Datensicherheit ist deutlich abstrakter als jene zu Storage-Systemen und Netzwerkkommunikation. Hier ist die Domäne auf einige für den Anwendungsfall interessante Regeln komprimiert und soll zeigen, wie Parameter, die von verschiedenen Domänen benötigt werden, auch in einer Domäne zusammen verwendet werden können, um komplexere Zusammenhänge zu erkennen.

Regeln

1. Ohne Redundante Speicherung ist die Zerstörungsgefahr erhöht.
2. Wenn das Speichersystem oder das Netzwerk gestört sind, reduziert dies die Verfügbarkeit.
3. Wenn mehr Clients über das Netzwerk verbunden sind, als Anwendungen mit dem Storage-System verbunden sind, deutet dies auf einen Angriff hin.

Die Übersetzung der hier aufgeführten Regeln befindet sich im Anhang (Tabelle A.9).

Angriffsgefahr Die Domäne Datenschutz und Datensicherheit wird durch den Indikator der Angriffsgefahr repräsentiert. Dieser Indikator kann genau wie die Verfügbarkeit der Storage-Systeme zwischen 0 und 1 liegen. Dabei gibt 0 an, dass das System aktuell angegriffen wird und 1 zeigt, dass das System vollständig sicher ist. Letzteres ist ein abstrakter Zustand, der nach aktuellem Verständnis nicht erreicht werden kann [Her16].

Verfügbarkeit Zusätzlich zur Angriffsgefahr wird die Domäne Datenschutz und Datensicherheit ebenfalls durch den Indikator Verfügbarkeit, der auch beim Storage-System verwendet wird, dargestellt. Diese beiden Indikatoren werden hier zusammengefasst.

4.3. Persona

Für die Situationsbewertung soll die Persona eines Systemadministrators verwendet werden. Dabei wird davon ausgegangen, dass sich der Systemadministrator sowohl für den eigentlichen Kern des Storage-Systems, als auch für die Netzanbindung interessiert, weshalb die beiden vorher beschriebenen Domänen von ihm einbezogen werden. Zusätzlich ist für den Systemadministrator ebenso die Datensicherheit von Interesse, weshalb auch die Domäne Datenschutz und Datensicherheit mit einbezogen wird. Die Grenzwerte der Persona sind in Tabelle A.10 im Anhang aufgeführt. Auf Präferenzen wurde hier verzichtet, da diese lediglich die Reihenfolge der Interpretation verändern würden.
Wie im vorherigen Kapitel beschrieben kann die Persona auch eigene Regeln haben, durch die sie die Situation bewertet. Die folgende Regel soll dies beispielhaft darstellen.

Der freie Speicher sollte nicht weniger als 10 % betragen.

Die Übersetzung der hier aufgeführten Regel befindet sich im Anhang (Tabelle tab:regP).
Diese Regel steht in gewissem Maße im Widerspruch zu der Regel Nr. 3 der Domäne für Storage-Systeme. Das bedeutet, dass für die Persona festgelegt wurde, dass für

Indikator	
Stabilität	0.9
Zukunft	0.9
Sicherheit	0.775
Verfügbarkeit	0.9
Angriffsgefahr	0.9

(a) Situation 1

Indikator	
Stabilität	**0.25**
Zukunft	**0.657**
Sicherheit	0.775
Verfügbarkeit	**0.725**
Angriffsgefahr	0.9

(b) Situation 2

Indikator	
Stabilität	0.9
Zukunft	**0.657**
Sicherheit	**0.6**
Verfügbarkeit	0.85
Angriffsgefahr	**0.3**

(c) Situation 3

Tabelle 4.1.: Ergebnisse der Situationsbewertung

sie ein Mangel an freiem Speicher erst viel später von Interesse ist, als durch die Domäne festgelegt wurde. Wenn das Gewicht der Regel der Persona deutlich stärker gewählt wird, als das der Regel aus der Domäne, kann es dazu kommen, dass die Regel der Domäne in diesem Fall sogar unwichtig wird.

4.4. Bewertung der Situationen

Die Bewertung einer Situation, welche den eigentlichen Kern des Frameworks darstellt, ist verhältnismäßig einfach, nachdem sowohl die Wirklichkeit, als auch die Domänen und die Persona sauber definiert wurden. Da für das hier betrachtete Anwendungsbeispiel keine Naturgesetze definiert wurden entfällt dieser Teil und es können direkt die Regeln der Domänen ausgewertet werden. Eine bestimmte Reihenfolge ist bei der Auswertung der Domänen bzw. ihrer Regel grundsätzlich nicht einzuhalten.
Nachdem die Regeln der Domänen, wie in Abschnitt 3.3.2 beschrieben, angewendet wurden, werden dazu analog die Regeln der Persona, wie in 3.3.3, angewendet. Zum Schluss werden die Ergebnisse, wie in Abschnitt 3.3.4, zusammengefasst. Die Ergebnisse dieses Vorgangs sind in den Tabellen 4.1a, 4.1b und 4.1c aufgeführt. Die Berechnung der hier aufgeführten Werte ist im Detail im Anhang A.8 beschrieben.
Die Ergebnisse der Situation können jetzt im nächsten Schritt interpretiert werden und es können Aussagen darüber getroffen werden, wie eine Aktion geartet sein muss, um die Situation zu verbessern oder sie zumindest zu erhalten.

4.5. Interpretation der Situation

Die Interpretation ist keine Aufgabe der Situationsbewertung und wird durch das Framework nicht unterstützt. Allerdings soll hier kurz auf die Ergebnisse eingegangen werden und beschrieben werden, wie die Ergebnisse verwendet werden können. Für den Fall, dass der Indikator außerhalb der Grenzwerte liegt, kann abgeleitet werden, dass eine Aktion dann sinnvoll ist, wenn sie mit einem akzeptablen Risiko und einer hohen Wahrscheinlichkeit den Indikator so beeinflusst, dass dieser danach innerhalb der Grenzwerte liegt oder zumindest näher an einem der beiden Grenz-

werte, die für den jeweiligen Indikator festgelegt wurden.
Die Indikatoren der ersten Situation (Tabelle 4.1a) liegen alle innerhalb der Grenzwerte, d.h. hier ist keine konkrete Aktion empfehlenswert. Es ist davon auszugehen, dass die Situation sich ohne ein Eingreifen nicht verschlechtert. Natürlich sind hier Aktionen durchaus möglich, die zum Beispiel die Indikatoren näher an den Mittelwert der jeweiligen Grenzwerte bringen. Solche Aktionen sollten hier allerdings nur mit enormer Vorsicht durchgeführt werden, da sie zu ungewollten negativen Nebeneffekten führen können und somit die Situation unnötig unsicher machen könnten.
Anders als in der ersten Situation liegen die Indikatoren *Stabilität*, *Zukunft* und *Verfügbarkeit* in der zweiten Situation (Tabelle 4.1b) außerhalb der Grenzwerte. Das bedeutet, dass hier ein Eingreifen empfohlen wird. Eine Aktion sollte möglichst die Verfügbarkeit des Storage-Systems verbessern, da dies der Indikator ist, der am weitesten vom Grenzwert entfernt ist. Weil auch die Stabilität scheinbar gefährdet ist, ist davon auszugehen, dass die Situation sich ebenso ohne Eingreifen schnell verändern kann. Allerdings ist der Indikator *Zukunft* ebenfalls gering, was bedeutet, dass die ungewollten Veränderungen der Situation sich wahrscheinlich zusätzlich negativ auf die Situation auswirken werden. Daher sollte die Aktion nicht nur die Verfügbarkeit des Storage-Systems verbessern, sondern auch versuchen die Stabilität der Situation zu verbessern, um einer weiteren Verschlechterung der Situation vorzubeugen.
In der letzten Situation (Tabelle 4.1c) scheint das Storage-System sich wieder in einem akzeptablen Zustand zu befinden, jedoch ist die Angriffsgefahr sehr hoch. Hierauf muss auf alle Fälle reagiert werden, da ebenfalls die Indikatoren für *Sicherheit* und *Zukunft* gering sind. Auf diese Situation sollte allerdings nicht mit der gleichen Aktion reagiert werden, die für die zweite Situation bereits verwendet wurde, da die Verfügbarkeit des Systems nicht gefährdet scheint.
Die hier aufgeführten Interpretationen erlauben es bereits mit den Ergebnissen der Situationsbewertung, Aktionen anhand abstrakter Merkmale auszuwählen. Sie eignen sich allerdings nicht, um eine konkrete Aktion auszuwählen. Mit einer größeren Anzahl an Indikatoren lassen sich die Aktionen immer weiter eingrenzen und somit immer konkreter bestimmen.

5. Diskussion

Zum Abschluss der Arbeit wird eine kurze Zusammenfassung gegeben. Dabei werden mögliche Einsatzgebiete, sowie Vorteile und Grenzen des Frameworks dargestellt. Die Einsatzgebiete lassen sich wahrscheinlich noch deutlich stärker ausweiten, da hier nur auf einen kleinen Ausschnitt eingegangen wird. Bei den Grenzen des Systems ist zu beachten, dass es keine Implementierung des hier vorgestellten Frameworks gibt und somit keine Tests unter natürlichen Bedingungen durchgeführt wurden.

5.1. Einsatzgebiete

Das Framework kann an extrem vielen Stellen eingesetzt werden, da es immer wichtiger wird, dass Maschinen in die Lage versetzt werden, Situationen zu bewerten. Als Beispiel seien hier autonome Fahrzeuge, digitale Märkte und vernetzte Maschinen erwähnt. Das menschliche Gehirn tut nichts anderes, als dauerhaft die Umwelt wahrzunehmen, zu bewerten und darauf zu reagieren. Was einem Menschen in jungen Jahren aufwendig beigebracht werden muss, muss ebenso einer Software beigebracht werden. Allerdings verläuft dieser Vorgang bei Software anders als beim Menschen. Hier setzt das Framework an, indem versucht wird die Situationsbewertung auf einer abstrakten Ebene zu beschreiben und somit von konkreten Situationen und den Einsatzgebieten der Software zu entkoppeln. Die Einsatzgebiete sind in diesem Kapitel unterteilt in die Überwachung von großen Systemen und die Unterstützung beim Katastrophenmanagement. Diese Unterteilung stellt keinen Anspruch auf Vollständigkeit und ist auch nicht die einzige Möglichkeit der Einteilung.

5.1.1. Überwachung von großen Systemen

Im Zuge der Globalisierung und zunehmendem technischen Fortschritts, steigt die Zahl an großen und durch ihre Größe komplexen Systemen. All diesen Systemen ist gleich, dass sie zu groß sind, als dass ein Mensch sie vollständig überblicken kann und daher ist der Einsatz des Frameworks in diesen Systemen sinnvoll, da das Framework nicht an die Grenzen eines Menschen gebunden ist. Im Folgenden wird kurz auf die drei Beispiele Rechenzentrum, Netzwerke und Finanzmärkte eingegangen.

Rechenzentren

Rechenzentren sind generell verhältnismäßig komplexe Anlagen, da sie aus einer Vielzahl von Servern, Netzwerkkomponenten, Klimaanlagen, Stromversorgungen und Sicherheitsanlagen bestehen. Die nächsten Jahre ist davon auszugehen, dass die Größe

und die Anzahl an Rechenzentren weiter ansteigen wird [Hin15], somit wird hier die Komplexität voraussichtlich ebenfalls weiter ansteigen. All die Informationen, die in einer solchen Anlage innerhalb kürzester Zeit anfallen, sind deutlich mehr, als ein Mensch verarbeiten kann. Alleine die Überwachung aller Temperaturen und Informationen zur Klimatisierung stellt in großen Rechenzentren eine echte Herausforderung dar. Durch das Framework wäre es möglich Zusammenhänge zwischen den laufenden Anwendungen und der Temperatur frühzeitig zu erkennen und somit die Klimaanlage zu steuern, bevor die Temperatur angestiegen ist.

Netzwerke

Die Komplexität eines Netzwerkes lässt sich daran festmachen, wie viele Knoten und Verbindungen das Netzwerk hat. Bezogen auf das Internet kann man deshalb sagen, die Komplexiät des Internets ist schnell gestiegen und wird voraussichtlich auch weiterhin stark ansteigen [LT12]. Durch die Einführung von zum Beispiel IPv6 [Hag14] ist die Anzahl der möglichen Knoten des Internets deutlich größer geworden und durch Ideen wie das Internet of Things [GBMP13] wird die Anzahl der Knoten voraussichtlich weiter steigen. In Netzwerken wie dem Internet ist es wichtig Engpässe oder Ausfälle früh zu erkennen und die damit verbundenen negativen Auswirkungen auf das gesamte Netzwerk zu identifizieren. Dies gilt ebenfalls für Netzwerke innerhalb der vorher beschriebenen Rechenzentren. Der Nutzen des Frameworks liegt hier in der Möglichkeit enorm viele Beziehungen innerhalb des Netzwerkes zu überwachen und somit Probleme zu erkennen, die nicht offensichtlich sind und wohl möglich lange Zeit verdeckt geblieben wären.

Finanzmärkte

Durch die Globalisierung wurden die Finanzmärkte zunehmend verknüpft und durch technische Fortschritte sind diese zudem schneller geworden. Teilbereiche der Finanzmärkte sind jetzt schon bei einer Geschwindigkeit angekommen, mit der Menschen nicht mehr umgehen können [Men13]. Diese Entwicklung hat schon 2010 zum sogenannten Flash Crash geführt [Mad12]. Finanzmärkte neigen allgemein auf Grund ihrer Komplexität zu Zusammenbrüchen (*engl. crashs*) [Sor09], welche möglichst vermieden werden sollten. Dies setzt voraus, dass man den Markt überwachen kann in der Geschwindigkeit mit der er selbst arbeitet, damit auf mögliche Probleme oder Risiken frühzeitig reagiert werden kann. Hier kann das Framework zum Einsatz kommen, da es nicht nur eine große Menge an Informationen verarbeiten kann, sondern auch, abhängig von der Hardware auf der es eingesetzt wird, mit der nötigen Geschwindigkeit.

5.1.2. Unterstützung beim Katastrophenmanagement

Neben großen Systemen ist auch das Katastrophenmanagement ein Einsatzbereich, der hervorgehoben werden sollte. Hier ist der Unterschied zu den vorher erwähnten

Systemen, dass es sich bei einer Katastrophe um eine dauerhafte Ausnahme handelt. Es ist also selten die Frage, ob gehandelt werden muss, und häufiger die Frage, wie gehandelt werden muss, zu klären. Das Katastrophenmanagement umfasst nach der Definition des IFRC die Organisation und Verwaltung aller Ressourcen und Verantwortungen, um die Auswirkungen von Katastrophen zu reduzieren [IFR]. Da Katastrophen Ausnahmesituationen sind, die die Betroffenen selbst nicht bewältigen können (KatSG §1 Abs. 1), ist es nicht trivial Entscheidungen zu treffen. In einer solchen Situation können sich Parameter schnell ändern und in der Eile können Informationen vergessen werden. Daher ist auch hier die Unterstützung der Entscheidungstreffer durch ein Framework zur Situationsbewertung sinnvoll. Dadurch kann sichergestellt werden, dass Informationen nicht vergessen werden und durch die Domänen ist es möglich, Bereiche zu überwachen, die womöglich außerhalb des Wissens des Anwenders stehen. Zudem wäre es möglich für verschiedene Arten von Katastrophen jeweils Domänen anzulegen, sodass ein Anwender einfach die Domänen auswählen kann und Informationen darüber erhält, wie die Situation zu bewerten ist und auf was besonders eingegangen werden muss. Auch hier bleibt es natürlich in der Verantwortung des Anwenders eine Aktion auszuführen, da das Framework lediglich versucht den Blick des Anwenders auf besonders wichtige Parameter zu lenken.

5.2. Vorteile des Frameworks

Die Vorteile, die ein System zur Situationsbewertung, in den vorgestellten und weiteren Einsatzgebieten, bringen kann, sind vom jeweiligen Einsatzgebiet abhängig. Sie lassen sich allerdings allgemein zusammenfassen, als Automatisierung eines manuellen Vorganges, Reduktion von Komplexität und Vermeidung von emotionalem oder intuitivem Handeln.
Durch das vorgestellte System kann der Vorgang der Situationsbewertung weitestgehend automatisiert werden, da für die gesamte Situationsbewertung eindeutige Regeln vorliegen. Allerdings bezieht sich diese Automatisierung lediglich auf die Bewertung. Nicht auf das vorher notwendige Entwickeln von Domänen etc.
Weil die Ergebnisse der Situationsbewertung durch eine übersichtliche Menge an Indikatoren repräsentiert werden, können sehr große und daher komplexe Systeme vereinfacht werden, wodurch es für einen Anwender einfacher wird Probleme innerhalb des eigentlich komplexen Systems zu erkennen. Deshalb ist es unter Anderem auch möglich besser mit zeitkritischen Anwendungen umzugehen, da dem Anwender eine Menge an Informationsverarbeitung abgenommen wird.
Die Vermeidung von emotionalem und intuitivem Handeln ist zwar nicht vollständig möglich, vor allem wenn ein menschlicher Anwender auf Grund der Situationsbewertung Entscheidungen trifft, allerdings kann solches Handeln in diesen Fällen stark reduziert werden, wenn dies sinnvoll ist. Dadurch wird dafür gesorgt, dass die getroffenen Entscheidungen eher rationaler Natur sind. Aus diesen Vorteilen lässt sich zusammenfassend sagen, dass ein Framework zur Situationsbewertung an vielen Stellen nicht den Grenzen eines einzelnen menschlichen Verstandes unterliegt, sondern

diese überschreiten kann. Vor allem was die Menge der zu verarbeitenden Informationen betrifft und die Eindeutigkeit der Ergebnisse, da eine Situation immer gleich bewertet wird.

5.3. Grenzen des Frameworks

Nachdem auf ein paar mögliche Einsatzgebiete und die Vorteile des Frameworks eingegangen wurde, sollen zum Abschluss die Grenzen des Frameworks aufgezeigt werden. Die Grenzen entstehen zum einen durch Annahmen, welche bei der Entwicklung des Frameworks getroffen wurden und können in gewissem Masse als frei gewählte Grenzen aufgefasst werden. Wohingegen es Grenzen gibt, welche nicht direkt aus Annahmen folgen, sondern aus der Idee der Situationsbewertung. Letztere sind nicht zwingend endgültig und können durch geeignete Methoden womöglich verschoben werden.

5.3.1. Annahmen

Bei der Entwicklung des Frameworks wurden verschiedene Annahmen getroffen, die die Möglichkeiten des Frameworks eingrenzen. Diese Annahmen sind teilweise nötig um die Komplexität des Frameworks zu reduzieren und teilweise um den Rahmen dieser Arbeit einzuhalten.
Die erste Annahme die getroffen wurde, bezieht sich auf die Messung der Situation. Hier wurde davon ausgegangen, dass eine irgendwie geartete Messung stattfindet, allerdings werden nur die Ergebnisse dieser Messung verwendet. Dies kann man als eine Art Black-Box vor der Situationsbewertung verstehen, da innerhalb der Situationsbewertung keine Informationen mehr dazu vorliegen, wie die Informationen gemessen wurden. Dies könnte verbessert werden, durch eine größere Schnittstelle zwischen Messung und Bewertung, in der auch Informationen über die Messung ausgetauscht werden können.
Es wird ebenfalls nicht auf die Situation reagiert. Alles was nach der Bewertung stattfindet befindet sich genau wie bei der Messung in einer Art Black-Box und die einzige Information, die dem Framework mitgeteilt werden kann, ist, ob der Bewertung zugestimmt wurde oder nicht. Allerdings ist keine Schnittstelle in dieser Richtung vorhanden, über die Informationen darüber ausgetauscht werden können, warum reagiert wurde oder eben nicht und wie die Reaktion von dem Anwender selbst eingestuft wurde. Diese beiden Annahmen machen das System nach außen in gewissem Maße blind.
Zudem wurde angenommen, dass sich jegliches Wissen in Domänen aufteilen lässt und durch die drei Arten von Regeln darstellen lässt. Da die drei Regelarten ziemlich abstrakt sind und sich kombinieren lassen ist die Menge an Wissen, die sich durch diese repräsentieren lässt enorm groß, allerdings ist es unwahrscheinlich, dass sich wirklich alles darüber abbilden lässt. Stattdessen wird davon ausgegangen, dass nur Wissen interessant ist, welches sich durch diese Regeln darstellen lässt. Zusätzlich

ist die Verwendung der vorgegebenen Regeln nicht immer der effizienteste Weg Wissen zu repräsentieren, was bedeuten würde, dass diese Annahme die Effizienz des Frameworks einschränkt.
Zu guter Letzt ist die Persona sehr stark eingegrenzt, wodurch sich viele Aspekte einer Person nicht abbilden lassen. Hier wurden bereits Ansätze zur Erweiterung vorgeschlagen, die jedoch auf Grund ihres Umfangs nicht in das Framework eingearbeitet werden konnten.

5.3.2. Grenzen

Die Grenzen des Frameworks verhalten sich anders als die Annahmen, da diese nicht frei wählbar sind, sondern direkt aus dem zu lösenden Problem oder dem zu modellierenden Objekt entstehen.
Sämtliche Grenzen des Systems lassen sich unter dem Problem der Beschreibbarkeit zusammenfassen, da diese niemals vollständig gegeben sein wird. Weder Wirklichkeit noch Domänen oder Persona werden vollständig beschrieben werden können und somit kann die Situationsbewertung nicht vollständig geschehen. Jedoch kann man sich dieser vollständigen Beschreibung immer weiter annähern, sodass die Differenz zwischen dem Modell, welches der Situationsbewertung zugrunde liegt und der Realität immer geringer wird. Eine vollständige Auflösung der Differenzen ist allerdings nicht möglich, da es sich immer nur um ein Modell handeln wird.

A. Daten des Anwendungsfalls

Bei den Skalen der Domänen wird die Punktnotation, wie sie auch im Datenbankbereich bekannt ist, verwendet. D. h. $q.p$ steht für den Wert p des Objektes q.

A.1. Parameter der Situation 1

	Parameter	p
Storage-System	$<< net\_storage, a, q, p >>$	1000
	$<< net\_storage\_in\_use, a, q, p >>$	800
	$<< gross\_storage, a, p >>$	1400
	$<< read\_rate\_max, a, p >>$	10
	$<< read\_rate, a, q, p >>$	4
	$<< write\_rate\_max, a, p >>$	6
	$<< write\_rate, a, q, p >>$	2
	$<< buffer, a, p >>$	30
	$<< buffer\_usage, a, q, p >>$	10
	$<< connections, a, p >>$	50
	$<< connections\_in\_use, a, X, q, p >>$	30
	$<< nodes, a, p >>$	6
	$<< node\_connections, a, q, p >>$	18
Netzanbindung	$<< upstream, b, p >>$	8
	$<< upstream\_usage, b, q, p >>$	4
	$<< downstream, b, p >>$	8
	$<< downstream\_usage, b, q, p >>$	3
	$<< ping, b, c, p >>$	3
	$<< connections, b, p >>$	50
	$<< connections\_in\_use, b, Y, q, p >>$	20

Tabelle A.1.: Parameter der Situation 1

A.2. Parameter der Situation 2

	Parameter	p
Storage-System	$<< net\_storage, a, q, p >>$	1500
	$<< net\_storage\_in\_use, a, q, p >>$	900
	$<< gross\_storage, a, p >>$	2000
	$<< read\_rate\_max, a, p >>$	10
	$<< read\_rate, a, q, p >>$	9.5
	$<< write\_rate\_max, a, p >>$	6
	$<< write\_rate, a, q, p >>$	5
	$<< buffer, a, p >>$	30
	$<< buffer\_usage, a, q, p >>$	25
	$<< connections, a, p >>$	50
	$<< connections\_in\_use, a, X, q, p >>$	50
	$<< nodes, a, p >>$	6
	$<< node\_connections, a, q, p >>$	18
Netzanbindung	$<< upstream, b, p >>$	8
	$<< upstream\_usage, b, q, p >>$	8
	$<< downstream, b, p >>$	8
	$<< downstream\_usage, b, q, p >>$	7
	$<< ping, b, c, p >>$	15
	$<< connections, b, p >>$	50
	$<< connections\_in\_use, b, Y, q, p >>$	30

Tabelle A.2.: Parameter der Situation 2

A.3. Parameter der Situation 3

	Parameter	p
Storage-System	$<< net\_storage, a, q, p >>$	1500
	$<< net\_storage\_in\_use, a, q, p >>$	900
	$<< gross\_storage, a, p >>$	2000
	$<< read\_rate\_max, a, p >>$	10
	$<< read\_rate, a, q, p >>$	9.5
	$<< write\_rate\_max, a, p >>$	6
	$<< write\_rate, a, q, p >>$	3
	$<< buffer, a, p >>$	30
	$<< buffer\_usage, a, q, p >>$	10
	$<< connections, a, p >>$	50
	$<< connections\_in\_use, a, X, q, p >>$	46
	$<< nodes, a, p >>$	6
	$<< node\_connections, a, q, p >>$	18
Netzanbindung	$<< upstream, b, p >>$	8
	$<< upstream\_usage, b, q, p >>$	4
	$<< downstream, b, p >>$	8
	$<< downstream\_usage, b, q, p >>$	8
	$<< ping, b, c, p >>$	5
	$<< connections, b, p >>$	30
	$<< connections\_in\_use, b, Y, q, p >>$	30

Tabelle A.3.: Parameter der Situation 3

A.4. Skalen und Regeln von Storage-Systemen

Skala	Parameter	Bild
read	$<< read\_rate, a, q, p >>$	$\frac{p}{q.p}$
write	$<< write\_rate, a, q, p >>$	$\frac{p}{q.p}$
buffer_free	$<< buffer\_usage, a, q, p >>$	$1 - \frac{p}{q.p}$
data	$<< net\_storage\_belegt, a, q, p >>$	p
netto	$<< net\_storage, a, q, p >>$	p
gross	$<< gross\_storage, a, p >>$	p
connections	$<< connections, a, p >>$	p
used_connections	$<< connections\_in\_use, a, X, q, p >>$	p
edges	$<< node\_connections, a, q, p >>$	p
nodes	$<< nodes, a, p >>$	p
applications	$<< connections\_in\_use, a, X, q, p >>$	$\|X\|$

Tabelle A.4.: Skalen von Storage-Systemen

Nr.	Bedingung	Schlussfolgerung
1	$\text{read} > 0.9 \vee \text{write} > 0.9$	$\text{Verfügbarkeit}(0.5, w, s)$
		$\text{Zukunft}(0.2, w, s)$
	$\text{read} \leq 0.9 \wedge \text{write} \leq 0.9$	$\text{Verfügbarkeit}(0.9, w, s)$
		$\text{Zukunft}(0.9, w, s)$
2	$\text{buffer\_free} \leq 0.3$	$\text{Verfügbarkeit}(0.5, w, s)$
		$\text{Stabilität}(0.2, w, s)$
	$\text{buffer\_free} > 0.3$	$\text{Verfügbarkeit}(0.9, w, s)$
		$\text{Stabilität}(0.9, w, s)$
3	$\frac{\text{data}}{\text{netto}} > 0.2$	$\text{Verfügbarkeit}(0.5, w, s)$
		$\text{Zukunft}(0.3, w, s)$
	$\frac{\text{data}}{\text{netto}} \leq 0.2$	$\text{Verfügbarkeit}(0.9, w, s)$
		$\text{Zukunft}(0.9, w, s)$
4	$\frac{\text{netto}}{\text{brutto}} < 0.7 \vee \frac{\text{netto}}{\text{brutto}} > 0,8$	$(\text{Verfügbarkeit}(0.4, w, s)$
		$\text{Sicherheit}(0.4, w, s)$
	$0.7 \geq \frac{\text{netto}}{\text{brutto}} \geq 0.8$	$\text{Verfügbarkeit}(0.9, w, s)$
		$\text{Sicherheit}(0.9, w, s)$
5	$\frac{\text{used\_connections}}{\text{connections}} > 0.9$	$\text{Verfügbarkeit}(0.3, w, s)$
		$\text{Zukunft}(0.4, w, s)$
	$\frac{\text{used\_connections}}{\text{connections}} \leq 0.9$	$\text{Verfügbarkeit}(0.9, w, s)$
		$\text{Zukunft}(0.9, w, s)$
6	$\text{edges} < \text{nodes} * 2$	$\text{Verfügbarkeit}(0.2, w, s)$
		$\text{Zukunft}(0.4, w, s)$
	$\text{edges} \geq \text{nodes} * 2$	$\text{Verfügbarkeit}(0.9, w, s)$
		$\text{Zukunft}(0.9, w, s)$
7	$\text{applications} * 3 < \text{used\_connections}$	$\text{Sicherheit}(0.2, w, s)$
	$\text{applications} * 3 \geq \text{used\_connections}$	$\text{Sicherheit}(0.9, w, s)$

Tabelle A.5.: Regeln von Storage-Systemen

A.5. Skalen und Regeln von Netzwerkkommunikation

Skala	Parameter	Bild
uplink	$<< upstream\_usage, b, q, p >>$	$\frac{p}{q.p}$
downlink	$<< downstream\_usage, b, q, p >>$	$\frac{p}{q.p}$
ping	$<< ping, b, c, p >>$	p
used_connections	$<< connections\_in\_use, b, Y, q, p >>$	p
connections	$<< connections, b, p >>$	p
clients	$<< connections\_in\_use, b, Y, q, p >>$	$\|Y\|$

Tabelle A.6.: Skalen von Netzwerkkommunikation

Nr.	Bedingung	Schlussfolgerung
1	uplink $>$ 0.9 $\vee$ downlink $>$ 0.9	Zukunft$(0.4, w, s)$
	uplink $\leq$ 0.9 $\wedge$ downlink $\leq$ 0.9	Zukunft$(0.9, w, s)$
2	ping $>$ 10	Stabilität$(0.3, w, s)$
	ping $\leq$ 10	Stabilität$(0.9, w, s)$
3	$\frac{\text{used\_connections}}{\text{connections}} > 0.9$	Zukunft$(0.4, w, s)$
	$\frac{\text{used\_connections}}{\text{connections}} \leq 0.9$	Zukunft$(0.9, w, s)$
4	clients $*$ 3 $<$ used_connections	Sicherheit$(0.1, w, s)$
	clients $*$ 3 $\geq$ used_connections	Sicherheit$(0.9, w, s)$

Tabelle A.7.: Regeln von Netzwerkkommunikation

A.6. Skalen und Regeln von Datenschutz und Datensicherheit

Skala	Parameter	Bild
netto	$<< net\_storage, a, q, p >>$	p
brutto	$<< gross\_storage, a, p >>$	p
buffer_free	$<< buffer\_usage, a, q, p >>$	$1 - \frac{p}{q.p}$
edges	$<< node\_connections, a, q, p >>$	p
nodes	$<< nodes, a, p >>$	p
applications	$<< connections\_in\_use, a, X, q, p >>$	$\|X\|$
ping	$<< ping, b, c, p >>$	p
clients	$<< connections\_in\_use, b, Y, q, p >>$	$\|Y\|$

Tabelle A.8.: Skalen von Datenschutz und Datensicherheit

Nr.	Bedingung	Schlussfolgerung
1	$\frac{\text{netto}}{\text{brutto}} > 0.7$	Sicherheit$(0.4, w, s)$
	$\frac{\text{netto}}{\text{brutto}} \leq 0.7$	Sicherheit$(0.9, w, s)$
2	buffer_free $\leq 0.05 \vee$ edges $<$ nodes $\vee$ ping > 50	Verfügbarkeit$(0.3, w, s)$
	buffer_free $> 0.05 \wedge$ edges $\geq$ nodes $\wedge$ ping ≤ 50	Verfügbarkeit$(0.9, w, s)$
3	clients $>$ applications	Angriffsgefahr$(0.3, w, s)$
	clients $\leq$ applications	Angriffsgefahr$(0.9, w, s)$

Tabelle A.9.: Regeln von Datenschutz und Datensicherheit

A.7. Persona

Domäne	Indikator	unterer Grenzwert	oberer Grenzwert
Naturgesetze	Stabilität	0.7	1
	Zukunft	0.7	1
	Sicherheit	0.7	1
Datenschutz & Datensicherheit	Verfügbarkeit	0.8	1
	Angriffsgefahr	0.5	1

Tabelle A.10.: Grenzwerte der Persona

Nr.	Bedingung	Schlussfolgerung
1	$\frac{\text{data}}{\text{netto}} > 0.9$	Verfügbarkeit$(0.4, w, s)$
		Zukunft$(0.3, w, s)$
	$\frac{\text{data}}{\text{netto}} \leq 0.9$	Verfügbarkeit$(0.9, w, s)$
		Zukunft$(0.9, w, s)$

Tabelle A.11.: Regeln der Persona

A.8. Berechnungen der Situationsbewertung

A.8.1. Situation 1

Auswertung der Domäne Storage-System

1. Verfügbarkeit: $(0.9, w, s)$ und Zukunft: $(0.9, w, s)$
2. Verfügbarkeit: $(0.9, w, s)$ und Stabilität: $(0.9, w, s)$
3. Verfügbarkeit: $(0.9, w, s)$ und Zukunft: $(0.9, w, s)$
4. Verfügbarkeit: $(0.9, w, s)$ und Sicherheit: $(0.9, w, s)$
5. Verfügbarkeit: $(0.9, w, s)$ und Zukunft: $(0.9, w, s)$
6. Verfügbarkeit: $(0.9, w, s)$ und Zukunft: $(0.9, w, s)$
7. Sicherheit: $(0.9, w, s)$

Auswertung der Domäne Netzanbindung

1. Zukunft: $(0.9, w, s)$
2. Stabilität: $(0.9, w, s)$
3. Zukunft: $(0.9, w, s)$
4. Sicherheit: $(0.9, w, s)$

Auswertung der Domäne Datenschutz und Datensicherheit

1. Sicherheit: $(0.4, w, s)$
2. Verfügbarkeit: $(0.9, w, s)$
3. Angriffsgefahr: $(0.9, w, s)$

Auswertung der Persona

1. Verfügbarkeit: $(0.9, w, s)$ und Zukunft: $(0.9, w, s)$

Zusammenfassung der Indikatoren

Zukunft:	$[7 \cdot (0.9, w, s)] \div 7 =$	**0.9**
Stabilität:	$[2 \cdot (0.9, w, s)] \div 2 =$	**0.9**
Sicherheit:	$[3 \cdot (0.9, w, s) + (0.4, w, s)] \div 4 =$	**0.775**
Angriffsgefahr:	$[(0.9, w, s)] \div 1 =$	**0.9**
Verfügbarkeit:	$[8 \cdot (0.9, w, s)] \div 8 =$	**0,9**

Tabelle A.12.: Indikatoren der Situation 1

A.8.2. Situation 2

Auswertung der Domäne Storage-System

1. Verfügbarkeit: $(0.5, w, s)$ und Zukunft: $(0.2, w, s)$
2. Verfügbarkeit: $(0.5, w, s)$ und Stabilität: $(0.2, w, s)$
3. Verfügbarkeit: $(0.9, w, s)$ und Zukunft: $(0.9, w, s)$
4. Verfügbarkeit: $(0.9, w, s)$ und Sicherheit: $(0.9, w, s)$
5. Verfügbarkeit: $(0.3, w, s)$ und Zukunft: $(0.4, w, s)$
6. Verfügbarkeit: $(0.9, w, s)$ und Zukunft: $(0.9, w, s)$
7. Sicherheit: $(0.9, w, s)$

Auswertung der Domäne Netzanbindung

1. Zukunft: $(0.4, w, s)$
2. Stabilität: $(0.3, w, s)$
3. Zukunft: $(0.9, w, s)$
4. Sicherheit: $(0.9, w, s)$

Auswertung der Domäne Datenschutz und Datensicherheit

1. Sicherheit: $(0.4, w, s)$
2. Verfügbarkeit: $(0.9, w, s)$
3. Angriffsgefahr: $(0.9, w, s)$

Auswertung der Persona

1. Verfügbarkeit: $(0.9, w, s)$ und Zukunft: $(0.9, w, s)$

Zusammenfassung der Indikatoren

Zukunft:	$[(0.2, w, s) + 2 \cdot (0.4, w, s) + 4 \cdot (0.9, w, s)] \div 7 \approx$	**0.657**
Stabilität:	$[(0.2, w, s) + (0.3, w, s)] \div 2 =$	**0.25**
Sicherheit:	$[3 \cdot (0.9, w, s) + (0.4, w, s)] \div 4 =$	**0.775**
Angriffsgefahr:	$[(0.9, w, s)] \div 1 =$	**0.9**
Verfügbarkeit:	$[(0.3, w, s) + 5 \cdot (0.9, w, s) + 2 \cdot (0.5, w, s)] \div 8 =$	**0.725**

Tabelle A.13.: Indikatoren der Situation 2

A.8.3. Situation 3

Auswertung der Domäne Storage-System

1. Verfügbarkeit: $(0.5, w, s)$ und Zukunft: $(0.2, w, s)$
2. Verfügbarkeit: $(0.9, w, s)$ und Stabilität: $(0.9, w, s)$
3. Verfügbarkeit: $(0.9, w, s)$ und Zukunft: $(0.9, w, s)$
4. Verfügbarkeit: $(0.9, w, s)$ und Sicherheit: $(0.9, w, s)$
5. Verfügbarkeit: $(0.9, w, s)$ und Zukunft: $(0.9, w, s)$
6. Verfügbarkeit: $(0.9, w, s)$ und Zukunft: $(0.9, w, s)$
7. Sicherheit: $(0.2, w, s)$

Auswertung der Domäne Netzanbindung

1. Zukunft: $(0.4, w, s)$
2. Stabilität: $(0.9, w, s)$
3. Zukunft: $(0.4, w, s)$
4. Sicherheit: $(0.9, w, s)$

Auswertung der Domäne Datenschutz und Datensicherheit

1. Sicherheit: $(0.4, w, s)$
2. Verfügbarkeit: $(0.9, w, s)$
3. Angriffsgefahr: $(0.3, w, s)$

Auswertung der Persona

1. Verfügbarkeit: $(0.9, w, s)$ und Zukunft: $(0.9, w, s)$

Zusammenfassung der Indikatoren

Zukunft:	$[(0.2, w, s) + 2 \cdot (0.4, w, s) + 4 \cdot (0.9, w, s)] \div 7 \approx$	**0.657**
Stabilität:	$[2 \cdot (0.9, w, s)] \div 2 =$	**0.9**
Sicherheit:	$[2 \cdot (0.9, w, s) + (0.4, w, s) + (0.2, w, s)] \div 4 =$	**0.6**
Angriffsgefahr:	$[(0.3, w, s)] \div 1 =$	**0.3**
Verfügbarkeit:	$[7 \cdot (0.9, w, s) + (0.5, w, s)] \div 8 \approx$	**0.85**

Tabelle A.14.: Indikatoren der Situation 3

Literaturverzeichnis

[ABK+07] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A Nucleus for a Web of Open Data. In Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, 2007.

[Agg15] Charu C. Aggarwal. *Data Mining - The Textbook.* Springer, 2015.

[All84] James F. Allen. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23(2):123–154, 1984.

[Bak09] Joanne Baker. *50 Schlüsselideen Physik.* Springer-Verlag, 2009.

[BP81] Jon Barwise and John Perry. Situations and Attitudes. *The Journal of Philosophy*, 78(11):668–691, 1981.

[CPSK07] Krzysztof J. Cios, Witold Pedrycz, Roman W. Swiniarski, and Lukasz Kurgan. *Data Mining: A Knowledge Discovery Approach.* Springer Science & Business Media, 2007.

[Dev91] Keith Devlin. *Logic and information.* Cambridge University Press, 1991.

[DLL00] Giuseppe De Giacomo, Yves Lespérance, and Hector J. Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1-2):109–169, 2000.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 226–231. AAAI Press, 1996.

[Fer12] David A. Ferrucci. Introduction to "This is Watson". *IBM Journal of Research and Development*, 56(3):1, 2012.

[FFL05] Alexander Ferrein, Christian Fritz, and Gerhard Lakemeyer. Using Golog for Deliberation and Team Coordination in Robotic Soccer. *Künstliche Intelligenz*, 19(1):24, 2005.

[FG93] Alice E. Fischer and Frances S. Grodzinsky. *The Anatomy of Programming Languages*. Prentice Hall, 1993.

[GBMP13] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013.

[Hag14] Silvia Hagen. *IPv6 Essentials*. O'Reilly Media, 2014.

[Her16] Cormac Herley. The Unfalsifiability of Security Claims. *Proceedings of the National Academy of Sciences*, 113(23):6415–6420, 2016.

[Hin15] Ralph Hintemann. Kurzstudie zur Entwicklung von Rechenzentren im Jahr 2014. `http://www.borderstep.de/publikation/hintemann-r-2015-kurzstudie-zur-entwicklung\-von-rechenzentren-im-jahr-2014/`, 2015. [erreicht am 21.06.2016].

[Hof93] Norbert Hoffmann. *Kleines Handbuch neuronale Netze - anwendungsorientiertes Wissen zum Lernen und Nachschlagen*. Vieweg, 1993.

[IFR] IFRC. International federation of red cross and red crescent societies. `http://www.ifrc.org/en/what-we-do/disaster-management/about-disaster-management/`. [erreicht am 23.06.2016].

[JAW+00] Thomas Jennewein, Ulrich Achleitner, Gregor Weihs, Harald Weinfurter, and Anton Zeilinger. A fast and compact quantum random number generator. *Review of Scientific Instruments*, 71(4):1675–1680, 2000.

[JM15] Volker Johanning and Roman Mildner. *Car IT kompakt*. Springer, 2015.

[JT14] Hannu Jaakkola and Bernhard Thalheim. Multicultural Adaptive Systems. In Bernhard Thalheim, Hannu Jaakkola, Yasushi Kiyoki, and Naofumi Yoshida, editors, *Information Modelling and Knowledge Bases XXVI, 24th International Conference on Information Modelling and Knowledge Bases (EJC 2014), Kiel, Germany, June 3-6, 2014*, volume 272 of *Frontiers in Artificial Intelligence and Applications*, pages 172–191. IOS Press, 2014.

[Kla12] Herbert Klaeren. Konzepte höherer Programmiersprachen (Entwurf), 2012. unveröffentlicht.

[Kle01] Hans-Joachim Klein. Null Values in Relational Databases and Sure Information Answers. In Leopoldo E. Bertossi, Gyula O. H. Katona, Klaus-Dieter Schewe, and Bernhard Thalheim, editors, *Semantics in Databases, Second International Workshop, Dagstuhl Castle, Germany, January 7-12, 2001, Revised Papers*, volume 2582 of *Lecture Notes in Computer Science*, pages 119–138. Springer, 2001.

[Koh82] Teuvo Kohonen. Self-organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[Kra93] Klaus Peter Kratzer. *Neuronale Netze - Grundlagen und Anwendungen (2. Aufl.)*. Hanser, 1993.

[Lin08] Fangzhen Lin. Situation Calculus. In Frank van Harmelen, Vladimir Lifschitz, and Bruce W. Porter, editors, *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 649–669. Elsevier, 2008.

[LRL+97] Hector J. Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *The Journal of Logic Programming*, 31(1-3):59–83, 1997.

[LT12] David G. Loomis and Lester D. Taylor. *Forecasting the Internet: Understanding the Explosive Growth of Data Communications*. Topics in Regulatory Economics and Policy. Springer US, 2012.

[LTL] Lennart S. Lutz, Tito Tang, and Markus Lienkamp. Analyse der rechtlichen Situation von teleoperierten (und autonomen) Fahrzeugen. [erreicht am 07.09.2016].

[Mad12] Ananth Madhavan. Exchange-Traded Funds, Market Structure, and the Flash Crash. *Financial Analysts Journal*, 68(4):20–35, 2012.

[McC63] John McCarthy. Situations, Actions, and Causal Laws. Technical report, DTIC Document, 1963.

[McC02] John McCarthy. Actions and Other Events in Situation Calculus. In Dieter Fensel, Fausto Giunchiglia, Deborah L. McGuinness, and Mary-Anne Williams, editors, *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002*, pages 615–628. Morgan Kaufmann, 2002.

[MD15] Eckard Minx and Rainer Dietrich. *Autonomes Fahren - Wo wir heute stehen und was noch zu tun ist*. Springer, 2015.

[Men13] Albert J. Menkveld. High frequency trading and the new market makers. *Journal of Financial Markets*, 16(4):712 – 740, 2013.

[MH69] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Readings in Artificial Intelligence*, pages 431–450, 1969.

[NYC15] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 427–436. IEEE Computer Society, 2015.

[Pip10] Nikolaus Piper. Das Wendejahr 2008. `http://www.sueddeutsche.de/geld/folgen-der-finanzkrise-das-wendejahr-1.385118`, 2010. [erreicht am 07.09.2016].

[PR95] Javier Pinto and Raymond Reiter. Reasoning about time in the situation calculus. *Annals of Mathematics and Artificial Intelligence*, 14(2-4):251–268, 1995.

[PSA14] Tassilo Pellegrini, Harald Sack, and Sören Auer. *Linked Enterprise Data - Management und Bewirtschaftung vernetzter Unternehmensdaten mit Semantic Web Technologien*. X.media.press. Springer, 2014.

[Rei77] Raymond Reiter. On Closed World Data Bases. In *Logic and Data Bases*, pages 55–76, 1977.

[Rei91] Raymond Reiter. The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression. *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, 27:359–380, 1991.

[Rei01] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT press, 2001.

[Ros58] Frank Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65(6):386, 1958.

[Soa15] Nate Soares. The Value Learning Problem. Technical report, Citeseer, 2015. [erreicht am 07.09.2016].

[Sor09] Didier Sornette. *Why Stock Markets Crash: Critical Events in Complex Financial Systems*. Princeton University Press, 2009.

[WMR10] Eggert Winter, Riccardo Mosena, and Laura Roberts. *Gabler Wirtschaftslexikon, 17 Auflage*. Gabler Verlag, 2010.

[ZGL16] David Zimbra, M. Ghiassi, and Sean Lee. Brand-Related Twitter Sentiment Analysis Using Feature Engineering and the Dynamic Architecture for Artificial Neural Networks. In Tung X. Bui and Ralph H. Sprague Jr., editors, *49th Hawaii International Conference on System Sciences, HICSS 2016, Koloa, HI, USA, January 5-8, 2016*, pages 1930–1938. IEEE Computer Society, 2016.