Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Parameterkurven haben Eigenschaften, die sich nur schlecht auf Papier bzw. zweidimensional darstellen lassen. In der Unity Umgebung besteht die Möglichkeit mittels Pfadanimation die Parameterkurven darzustellen. Diese Projektarbeit zeigt exemplarisch drei verschiedene Parameterkurven, die in der Virtuellen Realität dargestellt werden, um Eigenschaften der Kurven erlebbar zu machen.

Kapitel 2

Verwendete Technologie

2.1 Unity

Unity ist eine Grafik-Engine, mit deren Hilfe 3D-Anwendungen realisiert werden. Es exisitert ein integrierter Editor, der ähnlich einem 3D-Grafik Programm, dazu verwendet wird die entsprechende Umgebung zu gestalten. Die fertige Anwendung bzw. das Spiel können für unterschiedlichste Plattformen von Android bis hin zu Linux erzeugt werden. Diese Anwedungen sind unabhängig von der Unity Entwicklungsumgebung lauffähig. Es wird ein Kompilat mit Abhänigkeiten und Ressourcen erzeugt.

2.1.1 SciptEngine

Unity stellt eine ScripEngine zur Verfügung, mit deren Hilfe derer die Umgebung und die darin vorhandenen Objekte beinflusst werden kann. Als Sprache für die Skripte kommt C# zum Einsatz. Die Logik und der Ablauf der Ereignisse in der Anwendung werden durch Skripte festgelegt. Als Framework kommt das *Mono Framework* zum Einsatz, eine Open Source Implementierung von Microsofts *.NET Framework*.

2.1.2 SteamVR Plugin für Unity

Es existiert ein Plugin für die HTC Vive, eine VR-Brille. Dieses Plugin ist über den in Unity integrierten Asset Store zu beziehen. Es beinhaltet unter anderem sogenannte Prefabs, vorgefertigte Objekte, die in eine Szene eingesetzt werden können. Damit die Vive inklusive ihrer Controller funktioniert müssen die Prefabs [Steam VR] und [Camera Rig] in die Szene eingesetzt werden.

2.2 JetBrains Rider

Die Standart Entwicklungsumgebung für Unity ist *Microsoft Visual Studio 2017 in der Community Edition (kostenlos)*. Grundsätzlich ist jede IDE die C# unterstüzt geeignet. Visual Studio 2017 und JetBrains Rider bieten zudem die Möglichkeit, sich an den Unity Prozess anzuhängen und einen Debugger einzusetzen.
In diesem Projekt wurde Rider verwendet, da im Schwerpunkt auf Computern mit den Betriebsystem MacOS X entwickelt wurde und die Rider IDE in dieser Umgebung performanter und agiler ist.

2.3 HTC Vive

Das finale Anzeigemedium ist die VR-Brille des Herstellers HTC namens Vive. Auf ihr w

2.4 Beweisverfahren

2.5 Resolution

2.6 Hornklauseln

2.7 Berechenbarkeit und Komplexität

Der intuitive Weg ein Modell einer Formel zu bestimmen ist, einfach stur die Wahrheitstabellen aufzuschreiben und auszuwerten führt in jedem Fall in endlicher Zeit zum Ziel, dem bestimmen aller Modelle einer Formel. Es können Formeln jeden Typs entschieden werden. Dieses Verfahren lässt die benötigte Rechenzeit exponentiell wachsen, da jede Variable zwei Zustände haben kann ist die Komplexität 2^n.

Ein Lösungsansatz ist die Verwendung von semantischen Bäumen, diese betrachten nur Variablen die in den jeweiligen Klauseln vorkommen und sparen so Rechenzeit. Im worst case jedoch sind alle Variablen in den Klauseln vorhanden, was wiederum zur exponentiellen Komplexität führt.
Die in 2.5 besprochene Resolution hat ebenfalls einen „schlechten “ worst case. Die Zahl der abgeleiteten Klauseln wächst expotentiell mit den anfänglich vorhandenen Klauseln.

Da es offensichtlich keine gut skalierende Methode für alle Fälle gibt, muss der Algorithmus passend zum Problem ausgewählt werden. Wolfgang Ertel formuliert eine Faustregel: Bei vielen Klauseln mit wenigen Variablen ist die Wahrheitstafelmethode vorzuziehen. Wenige Klauseln mit vielen Variablen lassen sich mit der Resolution wahrscheinlich schneller beweisen.
Zur Zeit wurde noch kein schnellerer Algorithmus zum Beweisen von Aussagen gefunden. Wolfgang Ertel beruft sich auf einen Beweis des Begründers der Komplexitätstheorie. Dieser zeigte, das das sog. 3-Sat-Problem (Alle KNF-Formeln, deren Klauseln genau drei Literale haben) NP-vollständig ist. Vermutlich trifft dies auf auch auf eine allgemeine Lösung zu.
Die Eigenschaften der Hornklauseln jedoch erlauben es einen Algorithmus zu verwenden der die Rechenzeit „nur “ linear mit der Anzahl der Literale in der Formel wächst.

2.8 Anwendungen und Grenzen

Das automatisierte Beweisen von Aussagen ist aus der Entwicklung in der Digitaltechnik nicht mehr wegzudenken. Sei es beim Schaltungsentwurf oder beim Testen von Mikroelektronik bzw. Prozessoren. Hierfür existieren dedizierte Beweisverfahren. Im Bereich der KI sind einfache Systeme mit Aussagenlogischen Beweisern im Einsatz. Jedoch sind Vorraussetzungen zu beachten. Variablen müssen diskret sein, ohne Beziehung und die diskreten Werte sollten wenige sein.
Ein geeigneterer Weg ist die Prädikatenlogik, die im Folgenden besprochen wird.

Kapitel 3

Prädikatenlogik

In der Aussagenlogik sind Beziehungen zwischen bestimmten Variablen nicht ausdrückbar. Genauso können Eigenschaften für alle oder existierende Variablen in der Aussagenlogik nicht beschrieben werden. In der Prädikatenlogik sind jedoch diese Beziehungen und Eigenschaften ausdrückbar.
Die Prädikatenlogik erweitert die Aussagenlogik um Quantifizierungen, Konstanten, Variablen, Funktionssymbole und Prädikatensymbole. In Kombination mit den vorherigen vorgestellten Junktoren aus der Aussagenlogik handelt es sich um die Prädikatenlogik der Stufe 1.

3.1 Syntax

Definition Terme:
Zu den Termen gehören die Menge aller Variablen und Konstanten, diese werden atomare Terme genannt. Ein Funktionssymbol, das aus n-Termen besteht ist auch ein Term. Hierbei sind die Menge aller Variablen, Konstanten und Funktionssymbole paarweise disjunkt.[**?**, S.38 Def. 3.1]

Beispiel für Terme:

- Variable: x
- Konstanten: 3
- Funktionssymbol: $exp(x)$

Definition Quantoren:
Quantoren legen fest, für welche Variablen x einer Grundmenge eine Aussageform

$A(x)$ gilt, so gibt es den Allquantor und den Existenzquantor.
Der Allquantor beschreibt das für alle Variablen der Grundmenge das eine Aussageform $A(x)$, welche von der Variable x abhängig ist, diese Aussage gilt.
Der Existenzquantor beschreibt hingegen, dass es mindestens eine Variable aus der Grundmenge gibt, bei der Aussage $A(x)$ gilt. Des Weiteren gilt die Regel:

$$\forall x \varphi \equiv \neg \exists \neg \varphi$$

Der All- und Existenzquantor sind über diese Regel gegenseitig ersetzbar.

Beispiel:

$$\forall x \epsilon \mathbb{N}_0 : x \leq 0$$

„Für alle x aus dem Bereich der natürlichen Zahlen inklusive 0 gilt, dass $x \leq 0$ ist."
Diese Aussage ist logischer Weise falsch, da z.B. 1 eine natürliche Zahl ist und größer gleich 0 ist. Die korrekte Formulierung wäre:

$$\exists x \epsilon \mathbb{N}_0 : x \leq 0$$

„Es gibt mindestens ein x aus dem Bereich der natürlichen Zahlen inklusive 0, wo gilt das $x \leq 0$ ist." Diese Aussage ist korrekt, da z.B. 0 kleiner gleich 0 ist.

Definition Prädikatenlogische Formeln:

- Falls P ein Prädikatensymbol ist, mit k Anzahl an Argumenten und $t_1, ..., t_k$ Terme sind, so ist $P(t_1, ..., t_k)$ eine atomare Formel.
- Falls F eine Formel oder eine atomare Formel ist, so ist die Negation von F auch eine Formel oder atomare Formel
- $P(t_1, ..., t_n)$ und $\neg P(t_1, ..., t_n)$ heißen Literale.
- Falls F und G Formeln sind, so sind Kombinationen der Formeln mit Junktoren, bekannt aus der Aussagenlogik, z.B. $(F \wedge G)$ und $(F \vee G)$ auch Formeln
- Falls x eine Variable ist und F eine Formel, so sind auch $\forall x : F$ und $\exists x : F$ Formeln.
- Aussagen oder auch geschlossene Formeln werden so genannt, bei denen jede Variable im Wirkungsbereich eines Quantors liegt, das Gegenteil nennt man freie Variablen.

- Die Definition der Konjunktiven Normalform und der Hornklausel gelten für Formeln analog.

[**?**, vgl. S.38 Def. 3.2]

3.2 Semantik

„In der Prädikatenlogik wird die Bedeutung von Formeln rekursiv über den Formelaufbau definiert, indem wir zuerst den Konstanten, Variablen und Funktionssymbolen Objekte in der realen Welt zuordnen.“ [**?**, S.38] Anders ausgedrückt repräsentieren Formeln erst Aussagen, wenn den Funktionssymbolen, Konstanten und Variablen richtige Objekte zugewiesen werden, die wahr oder falsch sein können.

Beispiel:
$(mann(x) \Rightarrow \neg frau(x))$
Wenn x ein Mann ist, dann ist x keine Frau

Hans ist z.B. ein Objekt
$(mann(Hans) \Rightarrow \neg frau(Hans))$
Wenn Hans ein Mann ist, dann ist er keine Frau -> Hans ist Mann, ergo keine Frau.

Definition: Belegung bzw. Interpretation
In der Prädikatenlogik ist eine Belegung bzw. eine Interpretation eine Abbildung, die jeder freien Variable ein Element aus einer Grundmenge, oder auch Universum genannt, zugeordnet wird. Jedem n-stelligen Funktionssymbol wird eine n-stellige Funktion aus der Grundmenge zugeordnet und jedem n-stelligen Prädikatssymbol wird eine n-stellige Relation aus der Grundmenge zugeordnet.[**?**, vgl. S.40 Def. 3.3]

Definition: Wahrheit
Ist eine Formel Quantorenfrei, ergibt sie dich Wahrheit aus den atomaren Formeln, wie in der Aussagenlogik. Enthält eine Formel einen Allquantor, z.B. $\forall x : Formel$, dann ist die Formel gültig, wenn sie für alle Belegungen für die Variable x wahr ist. Bei einem Existenzquantor, z.B. $\exists x : Formel$ muss mindestens eine Belegung für die Variable x geben, welche die Formel wahr macht.[**?**, vgl. S.40 Def. 3.4]

Die in der Aussagenlogik vorgestellten Sätze des Deduktionstheorems und des Widerspruchsbeweises gelten auch für die Prädikatenlogik erster Ordnung.

Wolfgang Ertel[**?**] veranschaulicht an einem Beispiel, dass eine Formel von der Belegung abhängig ist, ob eine Formel der Wahrheit entspricht oder nicht.

$$F \equiv gr(plus(c_1, c_3), c_2)$$
$$Belegung_1 : c_1 \mapsto 1, c_2 \mapsto 2, c_3 \mapsto 3, plus \mapsto +, gr \mapsto >$$
$$1 + 2 > 2 \text{ nach Auswertung } 4 > 2$$
$$Belegung_2 : c_1 \mapsto 2, c_2 \mapsto 3, c_3 \mapsto 1, plus \mapsto -, gr \mapsto >$$
$$2 - 1 > 3 \text{ nach Auswertung } 1 > 3$$

[**?**, vgl. S.41]

„Die Wahrheit einer Formel in PL1 hängt also von der Belegung ab.“[**?**, S.40]

In der Wissensbasis (WB) können Relationen formalisiert werden, um z.B. die semantische Eindeutigkeit von Funktionssymbolen zu garantieren, so wird in Kapitel 3.2.1 von Wolfgang Ertel die Äquivalenzrelation die Prädikatenlogik definiert, sodass diese auch auf Funktionssymbolen und Prädikatensymbole funktionieren. Die Gleichheit ist in der Mathematik so definiert, dass die Relation reflexiv, symmetrisch und transitiv ist.[**?**, S.43]

3.3 Quantoren und Normalform

In der KI werden Formeln die Quantoren besitzen in einer äquivalenten standardisierten Normalform umgewandelt, damit möglichst wenig Quantoren in einer Formel auftauchen. Der Grund für die Umwandlung ist das Quantoren die Struktur der Formeln komplexer und dadurch das Beweisen mit anwendbarer Inferenzregeln umständlicher machen. [**?**, vgl. S.44]
So ist eine prädikatenlogische Formel P in pränexer Normalform, wenn alle Quantoren am Anfang der Formel sind und nach allen Quantoren eine quantorenfreie Formel folgt. [**?**, vgl. S.44]

Definition Pränexer Normalform
Eine prädikatenlogische Formel
$varphi$ ist in pränexer Normalform, wenn gilt

- $\varphi = Q_1 x_1 ... Q_n x_n \psi$

- ψ ist eine quantorenfreie Formel
- $Q_i \epsilon \forall, \exists$ für $i = 1, ..., n$.

[**?**, vgl. S.44 Def. 3.6]
Eine prädikatenlogische Formel lässt sich äquivalent in eine pränexer Normalform umwandeln. [**?**, vgl. S.46 Satz 3.2]
Um die Komplexität der Formel weiter zu verringern, können mit Hilfe der Skolemisierung alle Existenzquantoren eliminiert werden. Die resultierende Formel ist zwar nicht gleich der Ausgangsformel, jedoch bleibt die Erfüllbarkeit erhalten. Dies ist jedoch in vielen Fällen ausreichend, wenn man wie bei der Resolution die Unerfüllbarkeit von $WB \wedge \neg Q$ zeigen will.[**?**, vgl. S.46]
Nach der Skolemisierung erhält man eine Formel in konjunktiver Normalform. Dieses Verfahren kann in ein Programmschema dargestellt werden, wie in der Abbildung **??** veranschaulicht.
Ertel beschreibt außerdem, dass bei Transformation in die Normalform die Anzahl der Literale im worst-case exponentiell anwachsen, was zu exponentieller Rechenzeit und Speicherbedarf führen kann. Die Folge daraus ist, dass der Suchraum für einen anschließenden Resolutionsbeweis explosionsartig anwächst. [**?**, vgl. S.47]

Beispiel: Normalformtransformation

$$\forall x \exists y \forall z K(g(a,z),y) \implies K(g(f(a),f(z)),x)$$
Skolemform
$$\neg K(g(a,z),s(x)) \vee K(g(f(a),f(z)),x)$$

3.4 Resolution

Wolfgang Ertel beschreibt das Resolutionskalkül als eines der wichtigsten in der Praxis verwendeten automatisierbaren Kalkül für Formeln in konjunktiver Normalform. Es gibt noch weitere Kalküle wie z.B. das Gentzenkalkül oder das Sequenzenkalkül, diese sind jedoch nicht für Automatisierung gedacht. [**?**, vgl. S.47]
Ein Kalkül ist eine Sammlung von syntaktischen Umformungsregeln, die unter gegebenen Voraussetzungen aus bereits vorhandenen Formeln neue Formeln erzeugen.
Die Resolution ist ein eine Art Erfüllbarkeitstest um eine Formel auf ihre Gültigkeit zu testen. Das im Jahr 1965 vorgestellte Resolutionskalkül ist ein Widerlegungskalkül, welches die Formel auf Unerfüllbarkeit testet. Die Idee ist es die einen logischen

Widerspruch durch Verneinung der Formel abzuleiten, anstatt direkt die Allgemeingültigkeit der Formel zu zeigen. Anders ausgedrückt, um zu zeigen, das eine Formel allgemeingültig ist, muss gezeigt werden, dass die Verneinung der Formel unerfüllbar ist.
Das Resolutionskalkül ist aus der Aussagenlogik bekannt und besteht aus einer einzigen Umformungsregel. Entsteht durch die Resolution eine leere Klausel so ist die negierte Formel unerfüllbar und somit die nicht negierte Formel allgemeingültig.
Das gesamte Verfahren dient dazu, die Unerfüllbarkeit einer Formel in der Konjunktiven Normalform zu testen und gegebenenfalls nachzuweisen. Beim Verfahren muss man jedoch beachten, dass das Verfahren für einige Eingaben exponentielle Laufzeit besitzt.
Um die Resolution von der Aussagenlogik in die Prädikatenlogik auszuweiten, müssen die prädikatenlogischen Formeln umgeformt werden.
Im ersten Schritt wird die zu beweisende Formel negiert und durch die vorher beschriebenen Verfahren in eine der Aussagenlogik ähnliche konjunktive Normalform normalisiert.
Die Formel wird also in die Pränexeform gebracht, sprich alle Quantoren stehen am Anfang der Formel und hat die Gestalt einer konjunktiven Normalform. Anschließend werden alle Existenzquantoren durch die Skolemfunktion entfernt. Die Resultierenden Allquantoren können entfernt werden und das Ergebnis ist die Formel in Klauselform.
In der vorhin gezeigten Abbildung **??** ist die strukturierte Beschreibung der Normalformtransformation zu finden.
Im nächsten Schritt wird die Unifikation eingesetzt. Die Unifikation ist ein Ersetzungsschritt, um prädikatenlogische Ausdrücke zu vereinheitlichen.

Definition Unifikation:
Zwei Literale heißen unifizierbar, wenn es eine Ersetzung σ für alle Variablen gibt, welche die Literale gleich macht. Solch ein σ wird Unifikator genannt. Ein Unifikator heißt allgemeinster Unifikator (engl. most general unifier (MGU)), wenn sich aus ihm alle anderen Unifikatoren durch Ersetzung von Variablen ergeben. [**?**, vgl. S.51]

Beispiel:
Die Literale $P(f(g(x)), y, z)$ und $P(u, u, f(u))$ scheinen nicht resolvierbar zu sein, da sich die Terme unterscheiden. Jedoch sieht man durch die unifizierung, das beide vereinheitlicht werden können.

$$\sigma : \qquad y/f(g(x)), \qquad z/f(f(g(x))), \qquad u/f(g(x))$$

So ergeben sich durch die Unifizierung folgende Literale.

$$P(f(g(x)), f(g(x), f(f(g(x)) \text{ und } P(f(g(x)), f(g(x)), f(f(g(x))))$$

Diese kann nun resolviert werden, entsteht nun durch die aus der Aussagenlogik bekannten Resolvierung eine leere Klausel, so ist die negierte Formel unerfüllbar.[**?**, vgl. S.52]

Definition Prädikatenlogische Resolutionsregel:

$$\frac{(A_1 \vee ... \vee A_m \vee B), \quad (\neg B \vee C_1 \vee ... \vee C_n) \qquad \sigma(b) = \sigma(B')}{\sigma(A_1) \vee ... \vee \sigma(A_m) \vee \sigma(C_1) \vee ... \vee \sigma(C_n)}$$

wobei σ der "Most General Unifier" von B und B' ist.[**?**, S.52]

Definition Faktorisierung einer Klausel

$$\frac{(A_1 \vee A_2 \vee ... \vee A_n) \quad \sigma(A_1) = \sigma(A_2)}{\sigma(A_1) \vee ... \vee \sigma(A_n)}$$

wobei σ der "Most General Unifier" von A_1 und A_2 ist.[**?**, vgl. S.52]

Beim Resolutionsverfahren von prädikatenlogischen Klauseln kann es vorkommen, das gleichartige Terme mehr als einmal in der Resolvente stehen. Diese Klauseln kann man durch Faktorisierung vereinfachen. Ein anschauliches Beispiel ist das Barbier-Paradoxon.

Beispiel:
„Es gibt einen Barbier der alle Menschen rasiert, die sich nicht selbst rasieren.“In Prädikatenlogik der Stufe 1 formuliert:

$$\forall x \, rasiert(barbier, x) \Leftrightarrow \neg rasiert(x, x)$$

In diesem Beispielist die Aussage von Beginn an widersprüchlich ist und wir möchten durch Resolution die Unerfüllbarkeit ableiten. Durch die Normalformtransformation ergibt sich folgende Klauselform:

$$(\neg rasiert(barbier, x) \vee \neg rasiert(x, x))_1 \wedge (rasiert(barbier, x) \vee rasiert(x, x))_2$$

Mit dieser Klausel lässt sich noch kein Widerspruch ableiten, deswegen wäre es hier sinnvoll zu faktorisieren und auf die die faktorisierten Klauseln das Resoultionsverfahren anzuwenden.
Faktorisierung der ersten Klausel:

$$\frac{\neg rasiert(barbier, x) \vee \neg rasiert(x, x) \quad \sigma' = [x/barbier]}{\neg rasiert(barbier, barbier)_F 1}$$

Faktorisierung der zweiten Klausel:

$$\frac{rasiert(barbier, x) \vee rasiert(x, x) \quad \sigma'' = [y/barbier]}{rasiert(barbier, barbier)_F 2}$$

Aus den beiden faktorisierten Klauseln kann nun die leere Klausel abgeleitet werden.

$$Res(F1, F2) : leere\, Klausel$$

[**?**, vgl. S.52]
Die Resolutionsregel ist nicht vollständig, jedoch in Kombination mit der Faktorisierungsregel ist sie widerlegungsvollständig. Das heißt durch Anwendung beider Regeln lässt sich aus jeder unerfüllbaren Formel die leere Klausel ableiten.[**?**, vgl. S.53 Satz 3.6]

In der Praxis ist aufgrund des großen kombinatorischen Suchraums beim Resolutionsverfahren, die Suche nach einem Beweis sehr aufwändig. Der Beweiser erzeugt selbst nach Unifizierung mir jedem Schritt eine neue Klausel, welches die Anzahl an Klauseln im nächsten Resolutionssschritt erhöht. Aufgrund dessen wurden verschiedene Strategien entwickelt, um den Suchraum einzuschränken.
Zu den wichtigsten Strategien gehören die Unit-Resolution, Set of Support-Strategie, Input-Resolution, Pure Literal-Regel und die Subsumption. Die Unit-Resolution führt nicht immer zu einer Reduzierung des Suchraumes, daher gehört er zu den heuristischen Verfahren. Die Set of Support-Strategie (SOS) hingegen reduziert den Suchraum. Hier wird eine Teilmenge von $WB \wedge \neg Q$ als (SOS) definiert und bei jedem Resolutionsschritt wird eine Klausel aus dem SOS genommen. Die Strategie ist nicht vollständig, deshalb muss sichergestellt werden, dass die Menge der Klauseln auch ohne das SOS erfüllbar ist. Die Input-Resolution reduziert den Suchraum, in dem an jedem Resolutionsschritt eine Klausel der Eingabemenge $WB \wedge \neg Q$ beteiligt ist, sie ist auch nicht vollständig. Die Pure Literal-Regel ist vollständig und wird auch aktiv in der Praxis

verwendet, hier werden alle Klauseln entfernt, die Literale enthalten, zu denen es kein komplementäres Literal in den anderen Klauseln gibt. Bei der Subsumption werden auch Klauseln entfernt z.B. K_2, wenn Literale einer Klausel K_1 eine Teilmenge der Literale der anderen Klausel K_2 darstellen. [VGgl. S.53f] Ein weitere Quelle für eine Vergrößerung des Suchraumes ist die Gleichheit. Enthält eine Wissensbasis Gleichheitsaxiome und kann dies dazu führen das bei deren Ableitung neue Klauseln mit Gleichungen entstehen, auf die wieder Gleichheitsaxiome anwendbar sind. So wurden Inferenzregeln für Gleichheit entwickelt, wie z.B. die Demodulation, Paramodulation und Termersetzungssysteme für gerichtete Gleichungen.[**?**, vgl. S.54]

3.5 Automatische Theorembeweiser

Automatische Theorembeweiser sind Beweiskalküle, welche auf Rechnern implementiert wurden. Neben Spezialbeweisern für Teilmengen von PL1 existieren auch Beweiser für die volle Prädikatenlogik oder Logiken höherer Stufe.
So wurde bereits 1984 ein Resolutionsbeweiser namens Otter entwickelt, welcher vor allem in Spezialgebieten der Mathematik eingesetzt wurde. An der Universität München wurde der Beweiser SETHEO und später der auf Parallelrechnern implementierte Beweiser PARTHEO entwickelt. Später wurde dort auch noch der Gleichheitsbeweiser Namens E entwickelt.
Es zeigt sich allerdings im Zuge dieser Entwicklungen auch, das Hardware-basierte Beweiser nicht lohnenswert sind, da die Hardwareentwicklung so schnell voranschreitet, dass die neuen Beweiser schnell wieder veraltet sind.
Nimmt man in Betracht, dass sowohl Menschen als auch Maschinen bei vielen Beweisen an ihre Grenzen stoßen, so liegt es nahe, Eine kombinierte Version zu Entwickeln. So kann der Mensch mit seinem Vorwissen die Suche nach Beweisführungen einschränken und komplexe Einzelschritte von der Maschine übernehmen lassen.

3.6 Anwendungen

Automatische Theorembeweiser spielen in der Mathematik lediglich eine untergeordnete Rolle und die Prädikatenlogik war hauptsächlich in der Anfangszeit der KI von größerer Bedeutung.
Eine wichtige Rolle hingegen spielt die Logik heute bei Verifikationsaufgaben. Bei heutigen hoch komplexen Softwaresystemen ist es beinahe unmöglich alle Szenarien zu testen. Dies bietet ein gutes Einsatzgebiet für Inferenzsysteme. Auch können Sicher-

heitseigenschaften von kryptographischen Protokollen automatisch verifiziert werden. Des Weiteren wird auch die Wiederverwendung von Softwaremodulen durch Logik unterstützt. So kann ein Programmierer auf der Suche nach einem Modul durch Prädikatenlogik Vorbedingungen PRE_Q und Nachbedingungen $POST_Q$ des gesuchten Moduls formulieren werden. Bei der Suche nach einem passenden Modul M muss nun also $PRE_Q \Rightarrow PRE_M$ gelten.
Es müssen also bei M alle Voraussetzungen von PRE_Q gegeben sein. Zusatzbedingungen sind hier kein Problem.
Außerdem muss nun logischerweise auch für die Nachbedingungen $POST_M \Rightarrow POST_Q$ gelten. Es muss also nach der Anwendung von M alle von der Anfrage geforderten Eigenschaften erfüllt sein.

Eine weiteres Anwendungsgebiet der PL1 stellt das „Semantic Web“ dar, wodurch das World Wide Web für Maschinen besser interpretierbar wird. Durch die Erweiterung von Web-Seiten durch Beschreibungen ihrer Semantik, in einer formalen Beschreibungssprache, kann das Web effektiver nach syntaktischen Textbausteinen durchsucht werden. Diese Beschreibungssprache besteht aus entscheidbaren Teilmengen der Prädikatenlogik, wobei die Entwicklung von effizienten Kalkülen für das Schließen einen zentralen Punkt darstellt.
Das World Wide Web Consortium entwickelte die Sprache RDF (Resource Description Framework). OWL (Web Ontology Language), eine auf RDF aufbauende und deutlich mächtigere Sprache, erlaubt außerdem die Beschreibung von Relationen zwischen Objekten und Klassen von Objekten.

Kapitel 4

Grenzen der Logik

4.1 Das Suchraumproblem

4.1.1 Problemstellung

Bei der Suche nach Beweisen existieren, bei fast jedem Schritt, bis zu unendlich viele Möglichkeiten Inferenzregeln anzuwenden. Nimmt man den Worst-Case an, so müssen alle Möglichkeiten ausprobiert werden. Dies führt zu einem Zeitaufwand, welcher diesen Prozess nicht sinnvoll umsetzbar macht.
Es gibt jedoch Möglichkeiten mit diesem extrem anwachsenden Suchraum umzugehen. So zeigt sich, dass Menschen, obwohl viel langsamer in der Ausführung von Inferenzen, durchaus schneller beim Lösen schwieriger Probleme sein können. Dies resultiert aus Faktoren wie Intuition, Lemmas (Hilfssätzen) und Erfahrung, durch welche der Suchraum extrem verkleinert werden kann und somit eine deutliche Zeitersparnis mit sich bringen.
Das Ziel muss es also sein diese Faktoren in irgendeiner Form auf die Maschine übertragen werden.

4.1.2 Lösungsansätze

Die Idee ist nun, die zuvor genannten Eigenschaften auf Maschinen zu übertragen. Dazu kommen verschiedene Ansätze in Frage. So können zum Beispiel Heuristiken integriert werden. Durch diese wird versucht die Intuition nachzubilden.
Um dies zu erreichen, wird maschinelles Lernen eingesetzt. Es werden aus vorangegangenen erfolgreichen Beweisen Klauselpaare als positiv oder negativ gespeichert und es wird versucht, aus diesen Trainingsdaten ein Programm zu erzeugen, welches Klauselpaare heuristisch bewerten kann.

Ein weiterer Ansatz ist, durch interaktive Systeme den Menschen bei der Beweisführung zu unterstützen. So bleibt die Kontrolle über die Beweisführung zwar beim Menschen, aber es können zum Beispiel Algebraprogramme eingesetzt werden. Diese können für den Menschen schwierige mathematische Vorgänge übernehmen ohne die intuitive Lenkung der Beweisführung durch den Menschen aufzugeben.

4.2 Entscheidbarkeit und Unvollständigkeit

Durch die Prädikatenlogik erster Stufe gibt es korrekte und vollständige Kalküle und Theorembeweiser. Man kann demnach bei einer wahren Aussage in endlicher Zeit Beweisen, dass sie tatsächlich wahr ist. Dies gilt jedoch nicht für unwahre Aussagen, da die Menge der allgemeingültigen Formeln der Prädikatenlogik erster Stufe halbentscheidbar ist.
Ist eine Aussage nicht allgemeingültig, so kann es sein, dass der Beweiser nicht hält. Dies ist unpraktisch, da für eine bereits als wahr bekannte Aussage kein Beweis mehr notwendig ist. Für eine möglicherweise unwahre jedoch schon. Die Prädikatenlogik erweist sich als zu mächtige Sprache, um noch entscheidbar zu sein.
Möchte man jedoch eine Logik höherer Stufe quantifizieren, so ergibt sich schnell das Problem, dass die Vollständigkeit einer Logik sofort verloren geht, wenn man sie auch nur minimal erweitert.
Kurt Gödel hat in diesem Kontext den Gödelschen Unvollständigkeitssatz bewiesen. Dieser besagt, dass jedes Axiomensystem für die Natürlichen Zahlen mit Addition und Multiplikation (die Arithmetik) unvollständig ist. Das heißt, es gibt in der Arithmetik wahre Aussagen, die nicht beweisbar sind [vgl. S.68].

4.3 Ein Beispiel

Ein fundamentales Problem der Logik und passende Lösungsansätze, zeigt das folgende Beispiel [vgl. S. 71,72]

Es sei gegeben:
1. Tweety ist ein Pinguin
2. Pinguine sind Vögel
3. Vögel können fliegen

In Prädikatenlogik 1 ergibt sich als Wissensbasis:
$pinguin(tweety)$
$pinguin(x) \Rightarrow vogel(x)$
$vogel(x) \Rightarrow fliegen(x)$

Es lässt sich nun $fliegen(tweety)$ ableiten, jedoch können Pinguine nicht fliegen. Demnach gilt:
$pinguin(x) \Rightarrow \neg fliegen(x)$

daraus kann $\neg fliegen(tweety)$abgeleitet werden, was jedoch im Widerspruch zu fliegen(tweety) steht. Hier erkennen wir die Eigenschaft der Monotonie. Die Definition der Monotonie besagt, dass eine Logik monoton ist, wenn für eine beliebige Wissensbasis WB und eine beliebige Formel φ die Menge der aus WB ableitbaren Formeln eine Teilmenge der aus WB $\bigcup \varphi$ ableitbaren Formeln ist [vgl. S. 70].
Wird die Formelmenge also erweitert, wächst die Menge der beweisbaren Aussagen monoton. Demnach führt eine Erweiterung der Wissensbasis nie zum Ziel. Es muss also die falsche Aussage 3, Vögel können fliegen, durch eine präzisere Aussage ersetzt werden. Die neue Aussage heißt nun "Vögel außer Pinguine können fliegen". Aus dieser neuen Wissensbasis ergeben sich neue Klauseln:
$pinguin(tweety)$
$pinguin(x) \Rightarrow vogel(x)$
$vogel(x) \bigwedge \neg pinguin(x) \Rightarrow fliegen(x)$
$pinguin(x) \Rightarrow \neg fliegen(x)$

Der vorher entstandene Widerspruch ist nun behoben. Jedoch bleibt das Problem der erweiterbaren Wissensbasis weiter bestehen. Erweitert man nun diese um einen weiteren Typ Vogel so lässt sich mit der neuen Wissensbasis keinerlei Aussage über die Flugeigenschaften der neuen Vogelart machen. Es muss zuerst definiert werden, dass die neue Vogelart kein Pinguin ist. Folgt man dieser Logik weiter, so muss für jede Vogelart welche fliegen kann definieren, dass sie keine Pinguine sind. Jedoch muss man dann noch für alle anderen nicht fliegenden Vögel ebenfalls die Ausnahmen vergeben. Eine Lösung dieses Problems ist die Einführung einer Default-Logik, welche Default-Regeln etabliert. In diesem Beispiel wäre "Vögel können fliegenßolch eine Regel. Nun müssten nur noch alle Ausnahmen definiert werden und alle anderen würden automatisch mit dem Default-Wert ausgestattet.

4.4 Modellierung von Unsicherheit

Wie in dem vorher angeführten Beispiel gezeigt, gibt es Logiken, welche nicht sinnvoll mit wahr/falsch modellierbar sind. Es bietet sich an mit Wahrscheinlichkeiten zu arbeiten. so können Aussagen mit einer Wahrscheinlichkeit versehen werden um Ausnahmen darzustellen. Um komplexe Anwendungen mit vielen Variablen zu modellieren können Bayes-Netze verwendet werden. Zuletzt wurde außerdem noch die Fuzzy-Logik entwickelt um unscharfe Variablen modellieren zu können.
Vergleicht man die verschiedenen Logikformalismen ergibt sich folgendes Bild:

Formalismus	Anzahl der Wahrheitswerte	Wahrscheinlichkeiten ausdrückbar
Aussagenlogik	2	nein
Fuzzy-Logik	∞	nein
diskrete Wahrscheinlichkeitslogik	n	ja
stetige Wahrscheinlichkeitslogik	∞	ja

Literaturverzeichnis