Ein Verifizierter GDGL-Löser und Smales 14. Problem[1]

Fabian Immler[2]

Abstract: Diese Dissertation stellt eine Formalisierung von gewöhnlichen Differentialgleichungen (GDGL) und die Verifikation von rigorosen (mit garantierten Fehlerschranken) numerischen Algorithmen im interaktiven Theorembeweiser Isabelle/HOL vor. Die Formalisierung umfasst Fluss und Poincaré-Abbildung dynamischer Systeme. Die verifizierten Algorithmen basieren auf Runge-Kutta-Verfahren und affiner Arithmetik. Sie zertifizieren numerische Schranken für den Lorenz Attraktor und heben dadurch den numerischen Teil von Tuckers Beweis von Smales 14. Problem auf eine formale Grundlage.

1 Einleitung

Gewöhnliche Differentialgleichungen (GDGLen) modellieren eine Vielzahl an Systemen unserer alltäglichen Umgebung. Sei es die Bewegung von Teilchen, Autos, Zügen, Flugzeugen oder Planeten – oft ist eine zuverlässige Analyse sicherheitskritisch. Diese Dissertation zeigt dass derartige Analysen mit höchsten Korrektheitsgarantien ausgeführt werden können – nämlich mit formaler (und maschinenüberprüfbarer) formaler Logik.

Dies ist eine anspruchsvolle Thematik, da sie sich im Spannungsfeld von Mathematik und Informatik bewegt, zwischen der Modellierung kontinuierlicher Phänomene und Ihrer Implementierung mit diskreten Datenstrukturen, zwischen logischer Deduktion und numerischen Berechnungen.

Wir verwenden formale Logik, um zwischen diesen gegensätzlichen Konzepten zu vermitteln und stellen so eine mächtige Verbindung zwischen numerischen Berechnungen und logischer Deduktion her. Zudem schlagen wir die Brücke zwischen Theorie und Praxis: wir stellen realistische Implementierungen mit vernünftiger Performanz vor, welche zeigen, dass alle theoretisch etablierten Korrektheitsgarantien können auch praktisch ausgenutzt werden.

Wir konzentrieren uns auf sogenannte rigorose GDGL-Löser. Neben der Verifikation von bekannten und etablierten Methoden erforscht (und verifiziert formal) diese Arbeit auch neue Algorithmen zur Erreichbarkeitsanalyse kontinuierlicher Systeme. Die praktische Anwendbarkeit wird anhand einschlägiger Probleme aus dem Bereich cyber-physikalischer Systeme (im Rahmen der ARCH-Software-Competition) demonstriert.

Die finale große Anwendung ist Tucker's Beweis von Smale's 14. Problem, welches das volle Potential formal verifizierter Berechnungen aufzeigt: Bisher musste man für Tucker's Beweis den Berechnungen eines komplexen Computerprogramms vertrauen, nun

[1] Englischer Titel der Dissertation: "A Verified ODE Solver and Smale's 14th Problem"

[2] Institut für Informatik, Technische Universität München, immler@in.tum.de

bekommt man formale Garantien, die die Korrektheit des verwendeten Algorithmus auf die sicheren Grundlagen formaler Logik hebt.

Rigorose GDGL-Löser. Meist haben GDGLen keine symbolische Lösung, weshalb man auf numerische Verfahren zurückgreift, um Simulationen ihrer Evolution zu studieren. Für sicherheitskritische Anwendungen sind sogenannte *rigorose GDGL-Löser* (engl. rigorous ODE solver) von besonderer Bedeutung: Sie berechnen garantierte Schranken auf alle auftretenden Näherungsfehler.

Allerdings hängt die Korrektheit der berechneten Schranken davon ab, dass die zugrundeliegenden mathematischen Ideen wahrheitsgemäß implementiert werden. Dies ist – auch im Licht der zahllosen Beispiele von fehlerhaften Implementierungen, besonders in sicherheitskritischen Anwendungen – eine wirkliche Sorge. In der Tat existieren Beispiele von Fehlern in vermeintlich rigorosen GDGL-Lösern auf. Es gibt sogar Beispiele von Mängeln in zugrundegelegten mathematischen Ideen.

Formale Verifikation. Ein umfassender Ansatz um Korrektheit einer Implementierung sicherzustellen ist *formale Verifikation*. Das bedeutet, über Computerprogramme in einem strengen, logischen Kalkül zu argumentieren. Für Programme mit aufwändiger Spezifikation oder schwierigen Korrektheitsargumenten reichen vollautomatische Verifikationswerkzeuge nicht aus, hier ist formale Verifikation in *interaktiven Theorembeweisern* das Mittel der Wahl. Interaktive formale Verifikation verursacht enormen Aufwand. Dieser Aufwand lohnt, wenn starke Korrektheitsgarantien benötigt werden, insbesondere für grundlegende, etwaig sicherheitskritische Computerprogramme. Bemerkenswerte Beispiele sind der verifizierte Compiler CompCert [Le09] oder der verifizierte Betriebssystemkern seL4 [Kl09].

GDGLen sind allgegenwärtig in vielen wissenschaftlichen und technischen Disziplinen und demnach von ähnlicher grundlegender Bedeutung. Trotzdem wurde bisher kein rigoroser GDGL-Löser formal verifiziert. Es gibt reichlich Arbeit an (rigoroser) Numerik, von IEEE-Gleitkommazahlen bis hin zu formal verifizierten Bibliotheken für rigorose numerische (engl. rigorous numerics oder self-validated numerics) Berechnungen und nichtlinearer Optimierung. Bibliotheken für Mathematik, insbesondere Analysis, sind in vielen Beweisassistenten gut ausgebaut (wie in der Übersicht von Boldo *et al.* [BLM16] dargestellt), jedoch gibt es keine umfassende Formalisierung der Theorie von GDGLen.

- Ein Ziel dieser Arbeit ist die formale Verifikation eines rigorosen GDGL-Lösers – ein *verifizierter GDGL-Löser*.

Dies ist ein herausforderndes Unternehmen, da es sowohl die Formalisierung der zugrundeliegenden Mathematik als auch schwieriger Algorithmen verlangt. Diese Herausforderungen sind es Wert anzunehmen, da uns ein formal verifizierter GDGL-Löser mit bisher nicht gekannten Garantien für die berechneten Schranken belohnt.

Zudem streben wir eine Implementierung an, die nicht nur formal verifiziert ist, sondern auch angemessen effizient um realistische Beispielprobleme zu bearbeiten.

Smale's 14. Problem. Ein – neben sicherheitskritischen Systemen – weiteres Gebiet, in dem starke Korrektheitsgarantien wünschenswert sind ist die Mathematik, insbesondere im Kontext von Computerbeweisen (engl. computer-assisted proofs). Computerbeweise sind Beweise, die von der Ausgabe eines Computerprogramms abhängen und demnach entscheidend von einer korrekten Implementierung. Computerbeweise waren schon Ziel formaler Verifikation, wie sich in herausragenden Beispielen wie dem Flyspeck Projekt für einen formalen Beweis der Keplerschen Vermutung [Ha15] sowie der formalen Verifikation des Vier-Farben-Satzes [Go08] zeigt.

- Mit dem verifizierten GDGL-Löser zielen wir auf eine spezielle Anwendung ab: Tucker's Computerbeweis von Smale's 14. Problem (der Lorenz Attraktor).

Dies war ein wichtiges ungelöstes Problem, seine Lösung brachte Tucker z.B. den Preis der European Mathematical Society ein. Das Problem war Teil einer von Fields-Medaillenträger Stephen Smale zusammengestellten Liste mathematischer Probleme für das 21. Jahrhundert, neben der berühmten Riemannschen Vermutung oder der Frage ob $\mathsf{P} = \mathsf{NP}$. Tucker's Beweis [Tu02] basiert auf numerischen Schranken die von einem von ihm speziell für dieses Problem geschriebenen rigorosen GDGL-Löser berechnet wurden. In den ersten Versionen von Tucker's Programm wurden Fehler gefunden (und korrigiert). Aber dies verdeutlicht den Mehrwert formaler Verifikation: Mit einem verifizierten GDGL-Löser können wir sicher sein dass die berechneten Schranken nicht von etwaigen verbleibenden Fehlern kompromittiert sind.

2 Beiträge

Der zentrale Beitrag dieser Dissertation ist die formale Verifikation eines rigorosen GDGL-Lösers. Grundlegend für die Verifikation ist die Formalisierung von Mathematik GDGLen, insbesondere die Begriffe von Fluss und Poincaré-Abbildung. Die wesentliche Anwendung ist der Lorenz Attraktor, die Verifikation der Berechnungen von Tuckers Computerbeweis.

Der verifizierte GDGL-Löser ist modular aufgebaut, wesentlich um seine Verifikation handhaben zu können. Die verschiedenen Module sind derart unterschiedlich, dass jedes für sich genommen als unabhängiger Beitrag gesehen werden kann. Nichtsdestotrotz sind alle Module entlang Tuckers Beweis motiviert: Die Lorenz-Gleichungen erzeugen ein dynamisches System und demnach einen Begriff von Fluss. In seinem Beweis verwendet Tucker eine zur Analyse dynamischer Systeme übliche Technik, die sogenannte Poincaré-Abbildung. Tucker beweist, dass der Lorenz-Attraktor chaotisch ist, also insbesondere sensitiv von Anfangsbedingungen abhängt. Diese Abhängigkeit wird mit Ableitungen von Fluss und Poincaré-Abbildung quantifiziert. Neben der Formalisierung der abstrakten

mathematischen Konzepte implementieren und verifizieren wir rigorose numerische Algorithmen, die garantierte Schranken auf diese Quantitäten berechnen. Abschließend wenden wir diese Algorithmen auf Tuckers Computerbeweis an.

Alle in dieser Dissertation genannten Entwicklungen sind formalisiert, das heißt, in einer formalen Sprache geschrieben und mit einem maschinenüberprüfbaren Kalkül bewiesen. Die Formalisierung umfasst etwa 50000 Zeilen an Beweistext und ist im "Archive of Formal Proof" [IH17] verfügbar.

3 Formalisierung von Mathematik und GDGLen

Zur Formalisierung verwenden wir den Interaktiven Theorembeweiser Isabelle [Pa89] und seine populärste Instantiierung mit Logik höherer Stufe, Isabelle/HOL [NPW02]. Isabelle folgt der Tradition von Edinburgh LCF und besitzt einen kleinen Kern. Dieser Kern stellt eine abstrakte Schnittstelle zu sogenannten *formalen Theoremen* zur Verfügung und implementiert primitive logische Schlussregeln. Das System stellt sicher, dass formale Theoreme nur durch Anwendung primitiver Schlussregeln auf existierende formale Theoreme konstruiert werden können. Ein formales Theorem liefert demnach höchsten Standard mathematischer Strenge: seine Gültigkeit kann auf die Axiome der zugrundeliegenden Logik zurückgeführt werden.

Formalisierung von Mathematik in Isabelle/HOL. Wir verwenden Isabelle/HOL wird als Logik zur Formalisierung von Mathematik. Ein Alleinstellungsmerkmal von Isabelle/HOL unter anderen HOL-basierten Theorembeweisern sind *(axiomatische) Typklassen*. Typklassen sind sehr gut geeignet, um hierarchische Strukturen von Räumen in der Mathematik darzustellen und um polymorphe Spezifikationen zu organisieren. Hier geht es etwa um die Formalisierung von topologischen, metrischen, und Vektorräumen. Die Dissertation arbeitet aus, wie die (existierende) Analysis-Bibliothek entlang einer Hierarchie von Typklassen organisiert ist. Typklassen erlauben es beispielsweise, abstrakt Theoreme über metrische Räume zu beweisen. Konkrete Typen (etwa der Typ der reellen Zahlen, oder endlichdimensionale reellwertige Vektoren) können dann als Instanz einer Typklasse deklariert werden, was es erlaubt, die abstrakten Theoreme für den konkreten Typen zu verwenden.

GDGLen in Isabelle/HOL. Dieser Abschnitt liefert einen Überblick über die Formalisierung von GDGLen in Isabelle/HOL und stellt einige der formalisierten Hauptresultate heraus. Wir nehmen eine GDGL als gegeben durch Ihre rechte Seite f an: $\dot{x}(t) = f(t, x(t))$. Für einen Anfangswert $x(t_0) = x_0$ zu Anfangszeit t_0 existiert (unter Annahmen an f) eine eindeutige Lösung. Insbesondere dann wenn f nicht von t abhängt, wird die Lösung für $t_0 = 0$ als *Fluss* $\phi(x_0, t)$ bezeichnet (um die Abhängigkeit vom Anfangswert x_0 zu betonen). Die Poincaré-Abbildung ist ein wichtiges Werkzeug zur Analyse dynamischer Systeme. Ein Poincaré-Schnitt Σ ist eine Teilmenge des Zustandsraums. Für einen Punkt

$x \in \Sigma$, ist die Poincaré-Abbildung definiert als der Punkt $P(x)$, an dem der Fluss von x, zuerst zum Poincaré-Schnitt rückkehrt.

$$
\begin{aligned}
(\phi \textit{ solves-ode } f)\ T\ X :=& (\phi \textit{ has-vderiv-on } (\lambda t.\ f(t, \phi\ t)))\ T \wedge (\forall t \in T.\ \phi\ t \in X) \\
\textit{lipschitz } S\ g\ L :=& (\forall x, y \in S.\ \textit{dist } (g\ x)\ (g\ y) \leq L \cdot \textit{dist } x\ y) \\
\textit{local-lipschitz } f :=& \forall (t,x) \in T \times X.\ \exists \varepsilon > 0.\ \exists L. \\
& \forall u \in \mathscr{B}_\varepsilon(t) \cap T.\ \textit{lipschitz } (\lambda x.\ f(u,x))\ (\mathscr{B}_\varepsilon(x) \cap X)\ L \\
\textit{ll-open } T\ X\ f :=& \textit{open } T \wedge \textit{open } X \wedge \textit{local-lipschitz } T\ X\ f \wedge \\
& (\forall x \in X.\ \textit{continuous-on } T\ (\lambda t.\ f(t,x))) \\
(\phi \textit{ uniquely-solves-ode } f \textit{ from } t_0)\ T\ X :=& (\phi \textit{ solves-ode } f)\ T\ X \wedge t_0 \in T \wedge \textit{is-interval } T \wedge \\
& (\forall \psi.\ \forall T' \subseteq T.\ (t_0 \in T' \wedge \textit{is-interval } T' \wedge (\psi \textit{ solves-ode } f)\ T'\ X \wedge \psi\ t_0 = \phi\ t_0) \longrightarrow \\
& (\forall t \in T'.\ \psi\ t = \phi\ t)) \\
\phi(x_0, t) :=& \textit{sol } 0\ x_0 \\
\tau(x) :=& (\mathsf{LEAST}\ t > 0.\ \phi(x,t) \in \Sigma) \\
P(x) :=& \phi(x, \tau(x))
\end{aligned}
$$

Abb. 1: Auswahl an formalen Definitionen

Die wesentlichen Theoreme, die formalisiert wurden, betreffen lokale und globale Existenz und Eindeutigkeit, und Abhängigkeiten von Anfangsbedingungen von Fluss und Poincaré-Abbildung. Eine Auswahl an relevanten formalen Definitionen ist in Abbildung 1. Ein erstes grundlegendes Theoreme ist die Existenz einer (auf dem maximalen Existenzintervall *ex-ivl*, dessen Definition wir hier auslassen) eindeutigen Lösung *sol*. Technische Schwierigkeiten treten bei der Formalisierung auf, da Banachräume als Typklasse formalisiert sind. Weitere wesentliche Theoreme betreffen die Stetigkeit und Differenzierbarkeit (in x_0 und t) des Flusses ϕ und der Poincaré-Abbildung P.

4 Rigorose Numerik

Rigorose Numerik bedeutet, mit Mengen (anstatt mit näherungsweisen Werten) zu rechnen, welche garantieren, die anzunähernde Werte einzuschließen. Der klassische Ansatz zu rigoroser Numerik ist Intervallarithmetik. Aber prinzipiell können verschiedenste Datenstrukturen zur Darstellung der Einschließenden Mengen verwendet werden. Beliebte alternative Datenstrukturen sind „centered forms", „affine forms" oder Taylormodelle.

Eine zentrale Idee unseres Ansatzes ist das tiefe Einbetten arithmetische Ausdrücke, was es erlaubt viele Theoreme unabhängig von der später gewählten Datenstruktur zur Mengendarstellung zu beweisen. Für konkrete Berechnungen verwenden wir (beliebig genaue) Gleitkommazahlen mit expliziten Rundungsoperationen zur Effizienzsteigerung. Als Mengendarstellung konzentrieren wir uns auf „affine forms".

Affine Arithmetik. Ein Problem von klassischer Intervallarithmetik [MKC09] ist die Tatsache, dass Abhängigkeiten zwischen Variablen nicht berücksichtigt werden. Beispielsweise für $x \in [0;2]$, der Ausdruck $x - x$ wertet zu $[-2;2]$ in Intervallarithmetik aus, wohingegen das exakte Resultat mit dem Intervall $[0;0]$ darstellbar wäre.

Affine Arithmetik [dFS04] ist eine Erweiterung von Intervallarithmetik, die lineare Abhängigkeiten verfolgen kann. Die zugrundeliegende Datenstruktur ist eine formale Summe (eine „affine form") $A_0 + \sum_i \varepsilon_i A_i$ über formalen Parametern ε_i, welche über dem Intervall $[-1;1]$ interpretiert werden. Die Idee ist, dass die ε_i symbolisch behandelt werden und dadurch lineare Abhängigkeiten vermitteln.

Spezifikation and Verifikation des GDGL-Lösers. Eine wichtige Einsicht bei der Verifikation rigoroser numerischer Algorithmen ist die Tatsache, dass *jede* Menge, die den echten Wert enthält zum Beweis von Korrektheit ausreicht.

Das gibt die Möglichkeit, über Implementierungsdetails zu abstrahieren: Einerseits über die konkrete Darstellung der einhüllenden Menge und andererseits der eigentliche Algorithmus der die Einhüllung berechnet.

Wir verwenden Lammichs [La13] *Autoref*-Framework zum automatischen Verfeinern nichtdeterministischer Spezifikationen. Dieses Framework stellt die Infrastruktur zu Verfügung, welche wir benutzen um abstrakte Spezifikationen (wie etwa „eine Einhüllung der Lösung einer GDGL") zu konkreten, ausführbaren Implementierungen (wie etwa ein rigoroses Runge-Kutte-Verfahren) zu Verfeinern.

Der offensichtliche Vorteil eines Frameworks wie Autoref ist, dass man die Korrektheit eines Algorithmus sehr bequem auf einer abstrakten Ebene verifizieren kann und sich nicht um Implementierungsdetails scheren muss.

5 Verifikation Algorithmischer Geometrie

Die von einer „affine form" dargestellte Menge ist ein Zonotop. Schnitt von Zonotopen mit Hyperebenen ist eine wichtige Operations, zum Beispiel um Poincaré-Abbildungen zu berechnen. Dieser Abschnitt behandelt die Verifikation eines Algorithmus von le Guernic und Girard [GLG08] um den Schnitt von Zonotopen und Hyperebenen zu überapproximieren.

Formale Verifikation geometrischer Algorithmen ist ein herausforderndes und interessantes Thema. Solche Algorithmen sind leicht mit einer geometrischen Intuition präsentiert, aber diese Intuition muss formal präzise gefasst werden.

5.1 Knuths Counterclockwise Prädikate

Der Kern des Schnittalgorithmus den wir verifizieren ist ähnlich zu Berechnungen der konvexen Hülle einer Punktmenge in der Ebene. Für letztere Algorithmen entwickelte

Knuth [Kn92] eine Theorie, welche den Begriff von Orientierung von Punkten in der Ebene axiomatisiert. Die Idee ist, dass wenn drei Punkte p,q,r in der Ebene nacheinander besucht werden, ist eine Drehung entweder im oder gegen den Uhrzeigersinn nötig. Eine Drehung gegen den Uhrzeigersinn wird als ternäres Prädikat pqr geschrieben, im Uhrzeigersinn die Negation $\neg pqr$.

Knuth beobachtete, dass einige wenige Eigenschaften der ternären Prädikate ausreichen um viele Konzepte in algorithmischer Geometrie formal zu fassen. Drei der fünf Eigenschaften sind in Abbildung 2 illustriert.

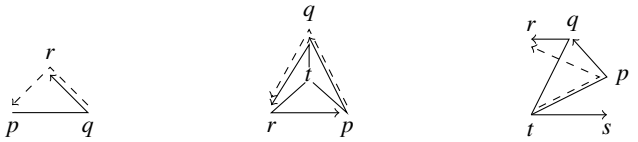
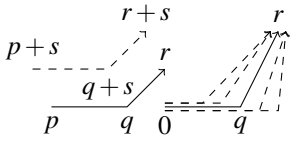


Abb. 2: Zyklische Symmetrie $pqr \longrightarrow qrp$ (links), Enthaltensein $tpq \wedge tqr \wedge trp \longrightarrow pqr$ (mitte), Transitivität $tsp \wedge tsq \wedge tsr \wedge tpq \wedge tqr \longrightarrow tpr$ (rechts); gestrichelte Prädikate werden von durchgezogenen impliziert.



Im Gegensatz zu Knuths Theorie, die auf endliche, diskrete Punktmengen abzielt, benötigt unsere Anwendung Mengen in kontinuierlichen Vektorräumen. Hierfür erweitern wir Knuths Theorie auf kontinuierliche Vektorräume. Zwei der zusätzlichen vier Eigenschaften sind beispielhaft links illustriert.

5.2 Verifikation von le Guernic and Girard's Algorithmus

Le Guernic and Girard's Algorithmus arbeitet für beliebig-dimensionale Zonotope. Der erste Schritt des Alorithmus reduziert das Problem auf mehrere zweidimensionale Probleme. Dieser Schritt ist leicht zu verifizieren. Der zweidimensionale Schnittalgorithmus ist aufwändiger: zunächst entledigen wir uns in einer Vorverarbeitung von kollinearen Punkten. Die eigentliche Verifikation ist wesentlich einfacher (da weniger Spezialfälle) wenn man zunächst nur das innere des Zonotops betrachtet und die Kanten vorerst ausschließt. Sie werden dann in einem letzten Stetigkeitsargument wieder mit in die Korrektheitsaussage aufgenommen.

6 Ein Verifizierter GDGL-Löser

Im Zentrum des verifizierten GDGL-Lösers steht eine Schleife zur Erreichbarkeitsanalyse. Gestartet auf einer Menge X_0 ist das Ziel, einen Poincaré-Schnitt Σ zu erreichen. Vergleiche auch Abbildung 3a. Die Schleife verwaltet drei Arten von Mengen: X ist die Menge deren zukünftige erreichbare Mengen noch weiter exploriert werden müssen. C ist die Menge aller bisher explorierten Mengen und I ist die Menge der Punkte für die die Erreichbarkeitsanalyse gestoppt hat, da sie den Poincaré-Schnitt Σ erreicht haben.

Die Schleife operiert immer auf einem Teil von X und entweder unterteilt diese Menge (dies erhält die Präzision aufrecht, sollte die Dynamik nicht-konvexe erreichbare Mengen erzeugen) oder wendet einen Schritt eines Runge-Kutta Verfahrens an, um die in einem Zeitschritt erreichbare nächste Menge herauszufinden. Die Runge-Kutta Schritte sind in affiner Arithmetik implementiert. Sollte ein solcher Runge-Kutta-Schritt den Poincaré-Schnitt Σ erreichen, muss die zugehörige Poincaré-Abbildung berechnet werden. Dies geschieht mit einer geometrisch berechneten Überapproximation (wie in Abschnitt 5 beschrieben) der exakten Schnittmenge.

Runge-Kutta-Verfahren werden verifizert indem (dies ist Standard für die Herleitung der Konvergenz von Runge-Kutta-Verfahren) die Taylorreihenentwicklungen von Lösung mit der der numerischen Approximation verglichen werden. Für rigorose Methoden wird noch eine explizite Schranke *rk2-remainder* für die Restglieder benötigt.

Lemma 6.1 (Runge-Kutta-Verfahren mit Fehlerschranke)**.** *Für* $0 < p \leq 1$ *und eine konvexe a-priori Schranke X für den Fluss* $\phi(x_0, [0;h]) \subseteq X$*:*

$$\phi(x_0, h) \in \mathit{rk2}_h(x_0) + \mathit{convex\text{-}hull}(\mathit{rk2\text{-}remainder}_h(x_0, p, [0;1], X))$$

7 Smale's 14th Problem

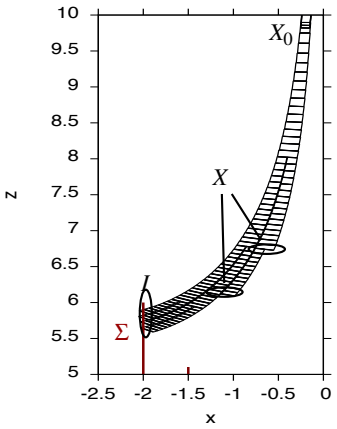
Im Jahr 1963, führte der Meteorologe Edward Lorenz [Lo63] das folgende System von GDGLen als vereinfachtes Modell für atmosphärische Dynamiken ein: $\dot{x} = -\sigma x + \sigma y, \dot{y} = -xz + \rho x - y, \dot{z} = xy - \beta z$ Lorenz stellte fest, dass selbst die kleinste Störung in Anfangsbedingungen zu komplett unterschiedlichem Langzeitverhalten des Systems führen würden. In Bezug auf seine ursprüngliche Motivation machte er den Begriff des Schmetterlingseffekts populär. Lorenz' System entwickelt sich zu einer komplizierten Struktur, dem sogenannten Lorenz-Attraktor (Abbildung 3b), welcher einen Kultstatus als Beispiel deterministischen Chaos genießt.

Trotz seiner Popularität und großem Aufwand der in seine Erforschung gesteckt wurde, konnte lange nicht bewiesen werden, dass der Lorenz-Attraktor in einem streng mathematischen Sinn chaotisch ist. Dies motivierte Fields-Medaillenträger Stephen Smale, den Lorenz-Attraktor auf seine Liste von achtzehn ungelösten mathematischen Problemen für das 21. Jahrhundert zu setzen [Sm98].

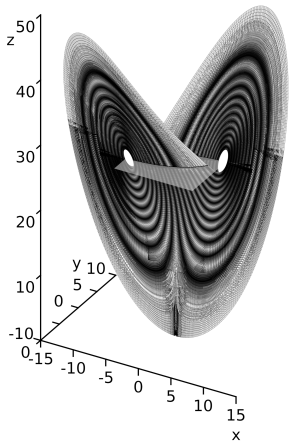
7.1 Verifikation von Tucker's Beweis

Tucker löste dieses Problem mit einem Computerbeweis. Er berechnete numerische Überapproximationen für einen Poincaré-Schnitt $\Sigma = [-6;6] \times [6;6] \times \{27\}$ im Lorenz-Attraktor (Abbildung 3b). Tucker identifiziert eine vorwärts invariante Region $N \subseteq \Sigma$, d.h., Lösungen die in N starten, kehren wieder nach N zurück. Zusätzlich berechnet Tucker die Ableitungen der zugehörigen Poincaré-Abbildung und zeigt, dass es ein (unter dem Bild der Ableitung) vorwärts invariantes Kegelfeld gibt, welches ausreichend stark expandiert. Dies

ist eine hinreichende Bedingung für Chaos. Der nicht-computergestützte Teil von Tuckers Beweis behandelt auch die stabile Mannigfaltigkeit Γ, deren relevanten Eigenschaften wir in unserem formalisierten Beweis annehmen. Zusammenfassend berechnet Tuckers Pro-



(a) Zustand während der Schleife zur Erreichbarkeitsanalyse.



(b) Numerische Schranken für den Lorenz Attraktor, formal verifiziert.

Abb. 3

gramm numerische Schranken für N, P, $\mathscr{E}$ und $\mathscr{E}^{-1}$, sodass folgende Eigenschaften gelten:

Theorem 7.1 (Vorwärts Invariente Menge, Ableitungen, Kegel und Expansionen)**.**
(1) $\forall x \in N - \Gamma.\ P(x) \in N$
(2) $\forall x \in N - \Gamma.\ \forall v \in \mathfrak{C}(x).\ \mathsf{D}P|_x \cdot v \in \mathfrak{C}(P(x))$
(3) $\forall x \in N - \Gamma.\ \forall v \in \mathfrak{C}(x).\ \|\mathsf{D}P|_x \cdot v\| \geq \mathscr{E}(x)\,\|v\|$
(4) $\forall x \in N - \Gamma.\ \forall v \in \mathfrak{C}(x).\ \|\mathsf{D}P|_x \cdot v\| \geq \mathscr{E}^{-1}(P(x))\,\|v\|$

Die Originaldaten von Tucker sind online verfügbar[3] als eine Unterteilung von N, assoziiert mit Informationen über $P, \mathscr{E}, \mathscr{E}^{-1}$. Wir beweisen Theorem 7.1 formal, indem jeder Teil von N mit dem verifizierten GDGL-Löser berechnet wird und überprüft wird, ob die assoziierten Schranken gültig sind.

Literaturverzeichnis

[BLM16] Boldo, Sylvie; Lelay, Catherine; Melquiond, Guillaume: Formalization of real analysis: a survey of proof assistants and libraries. Mathematical Structures in Computer Science, 26(7):1196–1233, 2016.

[dFS04] de Figueiredo, Luiz Henrique; Stolfi, Jorge: Affine Arithmetic: Concepts and Applications. Numerical Algorithms, 37(1-4):147–158, 2004.

[GLG08] Girard, Antoine; Le Guernic, Colas: Zonotope/Hyperplane Intersection for Hybrid Systems Reachability Analysis. In (Egerstedt, Magnus; Mishra, Bud, Hrsg.): Hybrid Systems: Computation and Control, Jgg. 4981 in LNCS, S. 215–228. Springer, 2008.

[3] `http://www2.math.uu.se/~warwick/main/rodes/ResultFile`

[Go08] Gonthier, Georges: Formal proof–the four-color theorem. Notices of the AMS, 55(11):1382–1393, 2008.

[Ha15] Hales, Thomas; Adams, Mark; Bauer, Gertrud; Dang, Dat Tat; Harrison, John; Hoang, Truong Le; Kaliszyk, Cezary; Magron, Victor; McLaughlin, Sean; Nguyen, Thang Tat et al.: A formal proof of the Kepler conjecture. arXiv preprint arXiv:1501.02155, 2015.

[IH17] Immler, Fabian; Hölzl, Johannes: Ordinary Differential Equations. Archive of Formal Proofs, September 2017. `http://isa-afp.org/entries/Ordinary_Differential_Equations.shtml`, Formal proof development.

[Im18] Immler, Fabian: A Verified ODE Solver and Smale's 14th Problem. Dissertation, Technische Universität München, München, 2018.

[Kl09] Klein, Gerwin; Elphinstone, Kevin; Heiser, Gernot; Andronick, June; Cock, David; Derrin, Philip; Elkaduwe, Dhammika; Engelhardt, Kai; Kolanski, Rafal; Norrish, Michael et al.: seL4: Formal verification of an OS kernel. In: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. ACM, S. 207–220, 2009.

[Kn92] Knuth, Donald: Axioms and Hulls. Springer, Berlin New York, 1992. Number 606 in Lecture Notes in Computer Science.

[La13] Lammich, Peter: Automatic data refinement. In: International Conference on Interactive Theorem Proving. Springer, S. 84–99, 2013.

[Le09] Leroy, Xavier: Formal verification of a realistic compiler. Communications of the ACM, 52(7):107–115, 2009.

[Lo63] Lorenz, Edward N.: Deterministic Nonperiodic Flow. Journal of the Atmospheric Sciences, 20(2):130–141, 1963.

[MKC09] Moore, Ramon E; Kearfott, R Baker; Cloud, Michael J: Introduction to interval analysis. SIAM, 2009.

[NPW02] Nipkow, Tobias; Paulson, Lawrence C.; Wenzel, Markus: Isabelle/HOL: A proof assistant for higher-order logic. LNCS. Springer, 2002.

[Pa89] Paulson, Lawrence C.: The foundation of a generic theorem prover. Journal of Automated Reasoning, 5(3):363–397, 1989.

[Sm98] Smale, Steve: Mathematical problems for the next century. The Mathematical Intelligencer, 20(2):7–15, 1998.

[Tu02] Tucker, Warwick: A Rigorous ODE Solver and Smale's 14th Problem. Foundations of Computational Mathematics, 2(1):53–117, 2002.

Fabian Immler, Jahrgang 1987, studierte Informatik (B.Sc. und M.Sc) von 2007 bis 2012 and der Technischen Universität München. Dort promovierte er [Im18] am Lehrstuhl für Logik und Verifikation, mit Forschungsaufenthalten bei Warwick Tucker's Gruppe "Computer Aided Proofs in Analysis" an der Uppsala University in Schweden. Seine Dissertation verteidigte er 2018 und erhielt dafür den „Heinz-Schwärtzel-Dissertationspreis für Grundlagen der Informatik". Die von ihm entwickelten und formal verifizierten Algorithmen erhielten den von Bosch gesponsorten „ARCH 2019 Best Result Award".