Grundlagen des aktiven Automatenlernens: Eine algorithmische Sichtweise[1]

Malte Isberner[2]

Abstract: Der immer schnellere, verstärkt auf Agilität fokussierte Prozess der industriellen Softwareentwicklung stellt ein großes Hindernis für die Anwendung formaler, modell-basierter Techniken dar. *Aktives Automatenlernen* verspricht mit der Möglichkeit, jederzeit auf dem neuesten Stand gehaltene Modelle automatisch generieren zu können, einen Ausweg. Der praktische Einsatz dieser Technologie brachte jedoch lange Zeit große Fragen mit sich, da existierende Algorithmen insbesondere für den vielversprechenden Anwendungsfall der *kontinuierlichen Validierung* inhärent ungeeignet waren. Durch eine komplett frische, erstmals formal grundierte Auseinandersetzung mit diesem nahezu dreißig Jahre alten Forschungsthema löst die vorliegende Dissertation das zentrale Problem der effizienten Behandlung von langen Gegenbeispielen, und bahnt so den Weg für eine Vielzahl von neuen Anwendungs- und Forschungsperspektiven.

1 Einleitung

Vor nahezu dreißig Jahren veröffentlichte Dana Angluin ihr vielbeachtetes Werk *Learning Regular Sets from Queries and Counterexamples* [An87], in welchem sie ein positives Lernbarkeitsresultat für reguläre Sprachen lieferte: mittels einer endlichen Anzahl sogenannter *Membership* und *Equivalence Queries*, die an einen *Lehrer* (*Teacher*) gestellt werden, kann ein *Lerner* ein Modell einer regulären (Ziel-)Sprache in Form eines deterministischen endlichen Automaten (*deterministic finite automaton, DFA*) erlernen. Eine *Membership Query* entspricht hierbei der Frage „Ist das Wort $w \in \Sigma^*$ ein Element der Zielsprache?“, und eine *Equivalence Query* steht für eine Anfrage der Form „Ist meine Hypothese der Sprache, repräsentiert durch den DFA $\mathcal{H}$, korrekt?“. Falls eine solche Anfrage positiv beantwortet wird, so ist der Lernvorgang offensichtlich abgeschlossen. Anderenfalls erwartet der Lerner ein *Gegenbeispiel*, welches die Fehlerhaftigkeit seiner Hypothese beweist und in der Folge zu einer *Verfeinerung* der Hypothese führt. Ein Lehrer, welcher für eine gegebene Zielsprache diese beiden Anfragetypen beantworten kann, heißt auch *minimal adäquat* (*minimally adequate teacher, MAT*), und der gesamte Prozess wird *aktives Automatenlernen* (*active automata learning*) genannt. Angluin zeigte weiterhin, dass eine – in der Größe des kanonischen DFA für die Zielsprache – polynomielle Anzahl solcher Anfragen ausreicht, und stellte auch den ersten polynomiellen Algorithmus zur Lösung des Problems, genannt L*, vor.

[1] Englischer Titel der Dissertation: „*Foundations of Active Automata Learning: An Algorithmic Perspective*“ [Is15].

[2] TU Dortmund, malte.isberner@cs.tu-dortmund.de

Obwohl dieses Resultat in der theoretischen Forschungsgemeinschaft auf viel Beachtung stieß, erschien es lange Zeit für die Praxis ungeeignet. So listet etwa eine Übersicht von de la Higuera [dlH05] aus dem Jahre 2005 als einzige potentielle Anwendung das Erlernen von Karten in unbekanntem Terrain auf – ein möglicher Anwendungsfall, dessen sich schon Rivest und Schapire [RS93] im Jahre 1993 bedient hatten, jedoch ohne von einer tatsächlichen Realisierung zu berichten. Als ein großes Hindernis für die praktische Anwendbarkeit wurde die Annahme der Existenz eines Lehrers, welcher definitiv richtige Antworten gibt, gesehen. In vielen potentiellen Anwendungskontexten wurde daher passiven Lernverfahren, bei welchen der Lehrer durch eine Menge von Beispielen (*Samples*) ersetzt wird, der Vorzug gegeben. Bei diesen Verfahren liegt der Fokus nicht darauf, *die richtigen Fragen zu stellen*, sondern ein Modell zu finden, welches möglichst gut im Einklang mit der Beispielmenge steht.

Gegen Ende des letzten Jahrtausends stieg das Interesse an aktivem Automatenlernen jedoch wieder sprunghaft an, was den Arbeiten zweier Forschergruppen zu verdanken ist: 1999 stellten Peled *et al.* ihre Idee des *Black-Box-Checking* [PVY99] vor, welche die Anwendung von Model Checking [CGP99] auf Black-Box-Systeme erlaubte. Unabhängig davon präsentierten Steffen *et al.* kurze Zeit später ihren Ansatz der *Test-basierten Modellinferenz* [Ha02]. Beiden Ansätzen war gemein, dass sie eine Brücke zwischen aktivem Automatenlernen und dem Feld der formalen Methoden schlugen. Dies war nicht zuletzt dadurch motiviert, dass trotz immer leistungsfähigeren Techniken die Anwendung formaler modellbasierter Methoden in der Praxis oft an einem vergleichsweise simplen Problem scheiterte: Modelle existieren oft nicht, sind nur partiell vorhanden (bspw. aufgrund der Verwendung externer Komponenten wie etwa Web-Services), oder repräsentieren eine veraltete Spezifikation, welche nicht mehr mit der Funktionalität der Implementierung übereinstimmt. Hier verspricht aktives Automatenlernen eine Abhilfe, da es zur Generierung von Modellen, welche das tatsächliche Systemverhalten reflektieren, benutzt werden kann.

Das Interesse der Formale-Methoden-Forschungsgemeinschaft an aktivem Automatenlernen führte zu großen Fortschritten in der praktischen Anwendbarkeit dieser Technologie. Motiviert durch eine Fülle von Anwendungsfällen wurden zahlreiche Optimierungen und Methoden zur Performanzverbesserung entwickelt. In einer vielbeachteten Fallstudie von Cho *et al.* [Ch10] wurde aktives Automatenlernen zur Inferenz des Modells eines Botnet-Steuerungsprotokolls eingesetzt, um so Wissen über die Funktionsweise zu erlangen und Gegenmaßnahmen entwickeln zu können.

Ein wenig überraschend erscheint in diesem Kontext die Tatsache, dass seit Angluins initialer Beschreibung ihres L*-Algorithmus nur wenige Verbesserungen auf der rein algorithmischen Ebene erzielt wurden. Zu nennen sind hier die Arbeiten von Rivest und Schapire [RS93], die eine neue effizientere Art der *Gegenbeispielbehandlung* einführten, von Kearns und Vazirani [KV94], welche die ursprüngliche tabellenartige durch eine baumbasierte Datenstruktur ersetzten, sowie von Howar [Ho12], der beide Ansätze zu einem neuen Algorithmus kombinierte. Noch überraschender erscheint es aber, dass viele Fallstudien – darunter auch die oben erwähnte von Cho *et al.* [Ch10] –

diese teilweise über zwanzig Jahre alten Entwicklungen ignorierten, trotz einer nachweislich höheren Effizienz.

Es darf vermutet werden, dass dies auf die Kompliziertheit der Lernalgorithem, deren Verhalten oftmals auch als überraschend und kontraintuitiv empfunden wird, zurückzuführen ist. So behauptete etwa Irfan [Ir12] noch 2012 in seiner Dissertation fälschlicherweise, dass der Algorithmus von Rivest und Schapire [RS93] nicht funktioniere. Kritisch muss ebenfalls angemerkt werden, dass nahezu sämtliche zuvor existierende Literatur zum aktiven Automatenlernen sich darauf beschränkt, das Funktionieren bestimmter Algorithmen zu beweisen, Fragen nach dem *Warum?* jedoch oft aus dem Weg geht.

1.1 Beitrag der Dissertation

Das Anliegen der Dissertation *Foundations of Active Automata Learning: An Algorithmic Perspective* [Is15] ist nicht weniger als die Behebung dieses Missstandes. Dies wird angegangen durch einen radikalen Paradigmenwechsel, weg von konkreten, notwendigerweise durch bestimmte Designentscheidungen eingeschränkten Algorithmen hin zu einer allgemein-mathematischen Beschreibung und Analyse des Problems. Aufbauend auf dieser ergeben sich ganz natürlich eine Reihe sowohl fundamental wichtiger als auch wünschenswerter Eigenschaften. Es wird ein theoretisches Framework entworfen, dessen allgemeine Natur es erlaubt, sämtliche existierenden Algorithmen in Hinblick auf diese Eigenschaften zu untersuchen und zu klassifizieren, und erlaubt so eine Beantwortung der Frage, warum manche Algorithmen anderen überlegen sind, und warum gewisse Phänomene nur bei bestimmten Algorithmen zu beobachten sind.

Dies führt auch zu der Erkenntnis, dass kein bisher beschriebener Algorithmus alle wünschenswerten Eigenschaften in sich vereint. Hierbei handelt es sich nicht um einen Zufall, da die beiden möglichen Gegenbeispielbehandlungsparadigmen – präfix-basierte und suffix-basierte Behandlung – inhärent mit einer Verletzung jeweils einer dieser Eigenschaften einhergeht. Das Beibehalten bzw. Wiederherstellen der wünschenswerten Eigenschaften ist daher ein komplexer, involvierter Prozess. Realisiert wird er im sogenannten TTT-Algorithmus, welcher im Rahmen dieser Dissertation neu und auf Basis des oben erwähnten mathematischen Frameworks entwickelt wurde. Anhand einer Reihe von Experimenten zeigt sich sehr deutlich, dass der Effekt hiervon weit über theoretische Gesichtspunkte hinausgeht: TTT ist der einzige existierende Algorithmus, der mit Gegenbeispielsequenzen großer Länge umgehen kann. Die oftmals vorgeschlagene Kombination von Lernen mit Monitoring zur Modellvalidierung wird daher durch den TTT-Algorithmus überhaupt erst möglich (mehr dazu in Abschnitt 2).

Im letzten Teil der Dissertation werden die Implikationen der gewonnenen Erkenntnisse auf das Lernen von Sprachklassen jenseits der regulären Sprachen untersucht. Als exemplarisches Beispiel wurden hier die sogenannten *Visibly-Pushdown*-Sprachen gewählt. Diese wurden von Alur und Madhusudan [AM04] als geeignetes

Modell für Programme mit Rekursion vorgeschlagen, und eröffnen dadurch ein weites Feld an interessanten Anwendungsfällen in Bezug auf Systeme, deren Verhalten mit endlichen Automaten nur inadäquat modelliert werden kann. Es zeigt sich, dass sich sämtliche im Rahmen der Dissertation entwickelten Konzepte und identifizierten Phänomene nahezu eins-zu-eins auf das Feld der *Visibly-Pushdown*-Sprachen bzw. -Automaten übertragen lässt. Dies umfasst auch eine Version des TTT-Algorithmus für diese Sprachklasse, welcher ähnlich herausragende Eigenschaften wie die Version für reguläre Sprachen aufweist.

2 Der TTT-Algorithmus

Auch wenn die Arbeit – wie oben aufgeführt – viele fundamentale Erkenntnisse und Einsichten über aktives Automatenlernen an sich enthält, so ist das Herzstück der Dissertation zweifellos der TTT-Algorithmus. Er ist nicht nur aufgrund seiner hohen Effizienz von großer praktischer Relevanz, sondern weist auch herausragende theoretische Eigenschaften auf. Damit stellt er einen direkten Beweis für den Wert der im ersten Teil der Arbeit durch eine rein mathematische Betrachtung gewonnen Erkenntnisse auf. Zudem liefert er eine Blaupause für die Entwicklung von effizienten Lernalgorithmen für weitere Sprachklassen, wie dies im dritten Teil der Arbeit anhand von *Visibly-Pushdown*-Sprachen ausgeführt wird. Der Rest dieser Zusammenfassung widmet sich daher detailliert diesem Algorithmus und seinen Eigenschaften.

2.1 Motivation

Die ursprüngliche Motivation für die Entwicklung des TTT-Algorithmus entsprang einer praktischen Herausforderung im Rahmen des EU-Projekts CONNECT [Is09]: hier wurde aktives Lernen eingesetzt, um formale (Verhaltens-)Modelle für vernetzte Systeme zu erhalten. Diese Modelle dienten als Eingabe für eine automatische Konnektorsynthese, mit dem Ziel der Herstellung von Interoperabilität in einem heterogenen Umfeld. Da *Equivalence Queries* in der Realität nur approximiert werden können, liefert aktives Lernen immer nur (stetig verbesserte) *Hypothesen*. Aufgrund dieser Tatsache, sowie um der Möglichkeit eines evolvierenden Systemverhaltens Rechnung zu tragen, sollten diese Modelle kontinuierlich mithilfe von Monitoring-Techniken validiert werden [Be12]. Die Aufgabe der Monitoring-Komponente war es also, Abweichungen zwischen dem tatsächlichen Systemverhalten sowie dem vom Modell vorausgesagten Verhalten zu erkennen, und in Form von Gegenbeispielen dem Lerner zugänglich zu machen. Abbildung 1 stellt dieses Szenario schematisch dar.

Es stellte sich jedoch heraus, dass sämtliche zu diesem Zeitpunkt existierenden aktiven Automatenlernalgorithmen für einen solchen Einsatzzweck ungeeignet sind. Dies hängt damit zusammen, dass die von der Monitoring-Komponente gelieferten Gegenbeispielen üblicherweise sehr lang waren: das gelernte Modell bildet in frühen Iterationen nur das „übliche“ Systemverhalten ab. Gegenbeispiele welche ein Verhalten

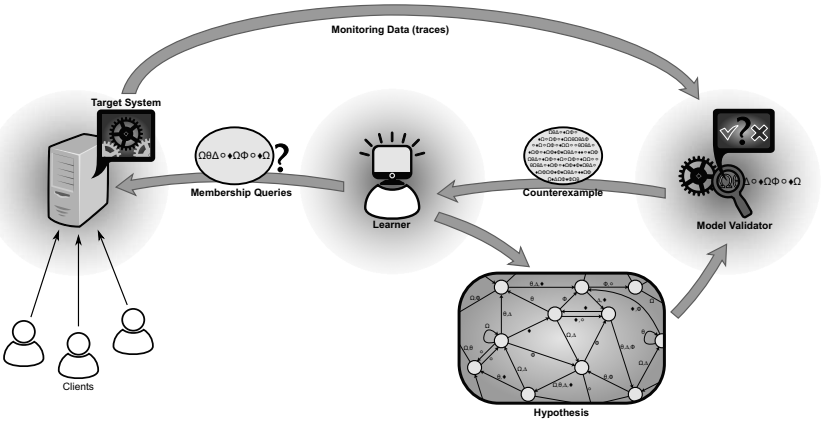


Abbildung 1: Visualisierung eines möglichen Szenarios welches Lernen und Monitoring kombiniert [IHS14b].

abseits des Üblichen aufzeigen haben daher ebenfalls oft einen sehr langen „Präfix üblichen Verhaltens“, in welchem keine Abweichung erkennbar ist.

Zwar wächst bei den meisten aktiven Lernalgorithmen die *Anzahl* an benötigten Membership Queries nur logarithmisch mit der Länge von Gegenbeispielen, die *Länge* dieser Queries jedoch linear. Dies ist aus zwei Gründen problematisch: zum einen steht die zur Ausführung einer Membership Query benötigte Zeit in einem direkten Zusammenhang mit der Länge dieser. Ist das Zielsystem beispielsweise ein entfernter (Web-)Service, so liegt die Zeit für die Ausführung eines einzelnen Symbols leicht im Bereich von mehr als 50 Millisekunden (verursacht durch Latenz), was Millionen von Rechenzyklen entspricht. Zum anderen ist der durch lange Gegenbeispiele hervorgerufene Effekt persistent: selbst nach der vollständigen Abarbeitung eines langen Gegenbeispiels sind die in zukünftigen Iterationen generierten Membership Queries signifikant länger. Dies liegt darin begründet, dass sämtliche Lernalgorithmen Teilstücke des Gegenbeispiels zu einer Verfeinerung ihrer Wissensbasis verwenden.

2.2 Idee

Der TTT-Algorithmus ersetzt diese *ad-hoc*-Verfeinerung auf Basis eines Gegenbeispiels durch eine involvierte Analyse, in welcher das Gegenbeispiel lediglich als Richtschnur zu einer natürlichen, lückenlos auf vorhandenem Wissen inkrementell aufbauenden Verfeinerung verwendet wird. Der Name TTT leitet sich aus dem wichtigsten Merkmal des Algorithmus ab: einem elaborierten Zusammenspiel dreier Baum-basierter Datenstrukturen. Dies ist exemplarisch in Abbildung 2 dargestellt: die *Spannbaum-Hypothese* (*spanning Tree*, links) stellt die vom Lerner vermutete Zustandsstruktur dar, wobei die Spannbaumkanten erreichende Pfade definieren. Die

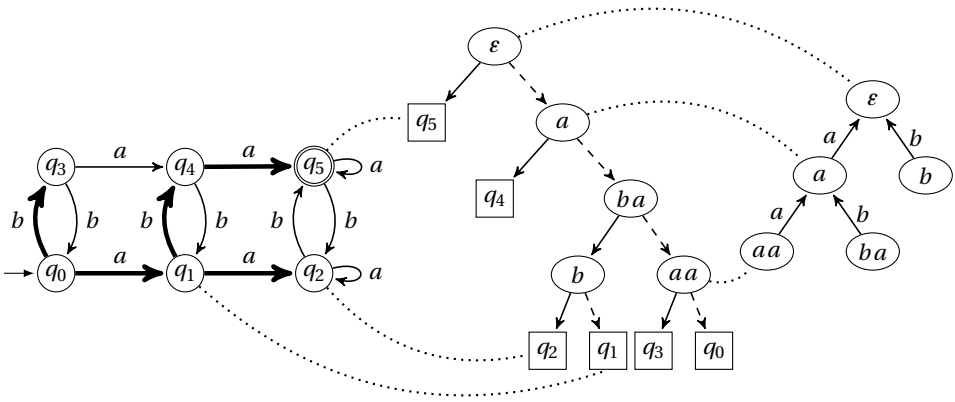


Abbildung 2: Zusammenspiel der Datenstrukturen im TTT-Algorithmus (v.l.n.r.): Spannbaum-Hypothese, *Discrimination Tree*, *Suffix Trie*

Trennung von Zuständen ist im *Discrimination Tree* (mitte) verankert; dieser Baum definiert damit eine Approximation der bekannten Nerode-Relation [Ne58]. Der *Suffix Trie* (rechts) schließlich speichert die sog. trennenden Suffixe, welche die inneren Knoten des Discrimination Tree beschriften.

Auch ohne die genaue Rolle dieser Phänomene im Detail zu erkennen erkennt man aus dem oben beschriebenen Aufbau sowie der Skizze in Abbildung 2 das Eingangs erwähnte zentrale Unterscheidungsmerkmal gegenüber anderen Algorithmen: der Aufbau des Wissens des Lerners, welches in den Datenstrukturen repräsentiert ist, geschieht *inkrementell* und *fundiert*. Die Spannbaumstruktur in der Hypothese stellt sicher dass jeder Zustand in einem Schritt von einem existierenden Zustand aus erreicht wird, und die Speicherung der Suffixe in einem Trie bedingt, dass sich die Trennung jedes Zustandspaares in einem Schritt auf die Trennung eines anderen Zustandspaares reduzieren lässt. Aus dieser *Redundanzfreiheit* folgen mehrere Alleinstellungsmerkmale, sowohl von der theoretischen als auch der praktischen Perspektive her.

2.3 Speicherplatzoptimalität

Eine besondere Eigenschaft des TTT-Algorithmus ist die Tatsache, dass er mit linearem Speicherplatz auskommt: sämtliche internen Datenstrukturen benötigen Speicherplatz in $\Theta(kn)$, wobei k die Alphabetgröße und n die Zustandsanzahl der Hypothese ist. Man erkennt leicht, dass dies dem asymptotischen Speicherplatz der Hypothese alleine (in einer üblichen Repräsentation, bspw. als Zustandsübergangstabelle) entspricht. Es lässt sich also festhalten, dass die vom TTT-Algorithmus für den Lern*prozess* benötigte Menge an Information asymptotisch gleich groß wie die im Lern*ergebnis* enthaltene Menge an Information ist.

Dass der TTT-Algorithmus eine lineare Speicherplatzkomplexität hat, wird bereits bei Betrachtung von Abbildung 2 deutlich: sämtliche internen Datenstrukturen sind baumbasiert, und haben so einen jeweils linearen Platzbedarf. Wie in der Arbeit detailliert nachgewiesen wird, ist dieser Speicherplatzbedarf optimal, d.h. kein anderer aktiver Lernalgorithmus kann mit asymptotisch weniger Speicher auskommen. Da ein Lernalgorithmus in seinen Datenstrukturen Informationen ablegt, die er vorher durch Membership Queries gewonnen hat, ist die Speicherplatzkomplexität Zeuge eines hocheffizienten Umgangs mit Informationen.

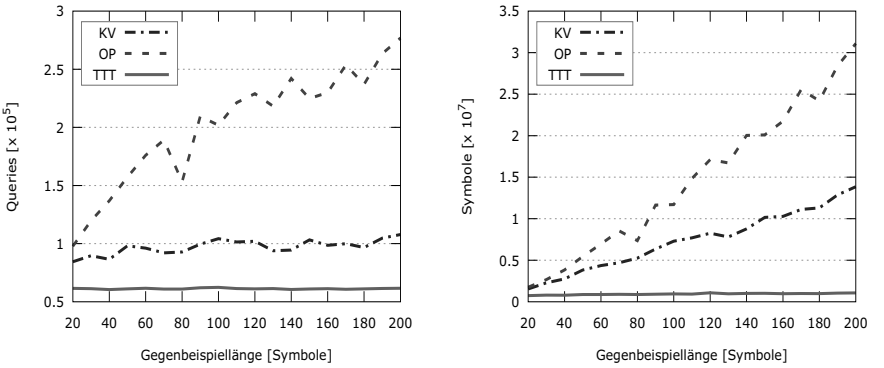
2.4 Praktische Evaluation

Um die praktische Bedeutung des TTT-Algorithmus besser analysieren zu können, wurden zahlreiche Experimentserien auf realistischen und synthetischen Beispielen durchgeführt. Hierfür wurde der TTT-Algorithmus auf Basis der *LearnLib*-Bibliothek [IHS15] implementiert, die in der Hauptsache vom Autor entwickelt wurde und der wissenschaftlichen Gemeinschaft unter einer Open-Source-Lizenz zur Verfügung steht.[3]

Über sämtliche Experimentserien hinweg zeigt sich das konsistente Bild, dass TTT allen anderen Algorithmen mindestens ebenbürtig ist. Dies ist insofern besonders, als dass die Performanz eines Algorithmus oft stark vom Profil des Zielsystems variiert – ein Algorithmus, der für einen Fall gut funktioniert, kann für ein anderes System sehr viel schlechter abschneiden als andere Algorithmen. TTT hingegen liefert konsistent eine Spitzenperformanz, und empfiehlt sich daher klar als *der* Algorithmus der Wahl für sämtliche Anwendungsfälle.

Bezogen auf das spezielle Szenario langer Gegenbeispiele, welches ja die ursprüngliche Motivation für die Entwicklung des TTT-Algorithmus darstellte (vgl. Abschnitt 2.1), ergibt sich ein völlig eindeutiges Bild: TTT schneidet hier nicht nur besser als sämtliche anderen Algorithmen ab, sondern wird von langen Gegenbeispielen in seiner Performanz nahezu nicht beeinflusst. Dies ist sehr schön in Abbildung 3 zu erkennen: der linke Plot zeigt die Anzahl Membership Queries verschiedener Algorithmen beim Lernen eines ausgewählten Systems (`pots2`) in Abhängigkeit von der Gegenbeispiellänge. Während eine Erhöhung dieser bei dem Algorithmus **OP** bereits zu einer deutlichen Verschlechterung der Performanz führt, hat dies beim Algorithmus **KV** keinen unmittelbar sichtbaren Einfluss. Dies ändert sich, wenn man statt der Anzahl Membership Queries deren kumulierte Länge betrachtet (rechts): während der Plot für TTT hier nach wie vor eine „Flatline“ zeigt, ist bei beiden Vergleichsalgorithmen ein sehr deutlicher Anstieg mit wachsender Gegenbeispiellänge zu erkennen. Vergleichbare oder sogar noch dramatischere Resultate zeigen sich für sämtliche Kombinationen von Zielsystemen und Vergleichsalgorithmen.

[3] `http://learnlib.de/`



(a) Anzahl Membership Queries

(b) Gesamtzahl Symbole in Membership Queries

Abbildung 3: Query- und Symbolkomplexität des TTT-Algorithmus auf dem Modell `pots2` im Vergleich mit anderen Algorithmen

2.5 Übertragung auf Visibly-Pushdown-Systeme

Einer der häufigsten Kritikpunkte an aktiven Lernverfahren betrifft die Tatsache, dass diese in der Regel auf die Klasse der regulären Sprachen beschränkt sind, was wiederum das mögliche erfassbare (bzw. erlernbare) Verhalten eines Systems einschränkt. Gleichwohl sind deutlich mächtigere Sprachklassen wie etwa kontextfreie Sprachen für diese Zwecke ungeeignet, da selbst mit Kenntnis der Grammatik bzw. eines Kellerautomaten viele Probleme unentscheidbar sind. Als Ausweg haben Alur und Madhusudan [AM04] *Visibly-Pushdown*-Automaten vorgeschlagen, also Kellerautomaten, bei denen die Aktion auf dem Keller – *push*, *pop*, oder keine Veränderung – durch das Eingabesymbol bestimmt sind (nicht jedoch das Kellersymbol). Verwendet werden diese üblicherweise zur Modellierung von Systemen mit (rekursiven) Funktionsaufrufen.

Im Rahmen der Dissertation wurde daher ebenfalls eine Variante des TTT-Algorithmus für *Visibly Pushdown*-Sprachen entwickelt. Dem voraus geht eine detaillierte Analyse der sich in diesem Problemfeld ergebenden Phänomene und Konzepte. Diese lassen sich zu großen Teilen auf entsprechende Konzepte aus dem Bereich des aktiven Lernens regulärer Sprachen übertragen, mit dem Resultat, dass sich die TTT-Variante für *Visibly Pushdown*-Sprachen nahezu von selbst, bzw. durch reinen Analogieschluss ergibt. In Experimentserien wird weiterhin gezeigt, dass dieser anderen Algorithmen auf eine ähnliche Art und Weise überlegen ist wie in der Domäne der regulären Sprachen, beispielsweise im Hinblick auf sein Laufzeitverhalten für lange Gegenbeispiele.

3 Ausblick

Es ist die persönliche Hoffnung des Autors, dass seine Dissertation eine gravierende Änderung der Forschungsperspektive in Bezug auf aktives Automatenlernen anstößt: weg von einem *ad-hoc*-Vorgehen mit nachträglicher Rechtfertigung, hin zu einer sorgfältigen, mathematischen Analyse der Konzepte und Phänomene, welche es erlaubt eine der zentralen Fragen in diesem Feld zu beantworten: „Welche sind die *richtigen* Fragen, die gestellt werden *müssen*?“

Auch ohne solcherart philosophische Betrachtungen steht der Wert dieses Vorgehens außer Frage. Der TTT-Algorithmus positioniert sich klar als der beste existierende Algorithmus für das allgemeine Problem des Automatenlernens. Ein Gewinn für die zukünftige Forschung ist aber nicht zuletzt auch von der Übertragung auf *Visibly Pushdown*-Systeme zu erhoffen. Abgesehen von den direkt hierdurch eröffneten Anwendungsfällen liefert diese eine Blaupause, wie allein durch Identifikation der richtigen Konzepte und Analogieschlüsse sich ein hocheffizienter Algorithmus für eine neue Sprachklasse geradezu von selbst ergeben kann. Es ist zu erwarten, dass eine Übertragung auf Lernverfahren für andere Sprachklassen – wie etwa Registerautomaten [IHS14a] – zu signifikanten Effizienzsteigerungen und einem drastisch gesteigerten Potential dieser Technologie führen wird.

Literatur

[AM04] Alur, R.; Madhusudan, P.: Visibly Pushdown Languages. In: Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing. STOC ’04, ACM, New York, NY, USA, S. 202–211, 2004.

[An87] Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation, 75(2):87–106, 1987.

[Be12] Bertolino, A.; Calabrò, A.; Merten, M.; Steffen, B.: Never-stop Learning: Continuous Validation of Learned Models for Evolving Systems through Monitoring. ERCIM News, 2012(88):28–29, 2012.

[CGP99] Clarke, E. M.; Grumberg, O.; Peled, D. A.: Model Checking. The MIT Press, Cambridge, MA, USA, 1999.

[Ch10] Cho, C. Y.; Babić, D.; Shin, E. C. R.; Song, D.: Inference and Analysis of Formal Models of Botnet Command and Control Protocols. In: Proceedings of the 17th ACM Conference on Computer and Communications Security. CCS ’10, ACM, New York, NY, USA, S. 426–439, 2010.

[dlH05] de la Higuera, C.: A Bibliographical Study of Grammatical Inference. Pattern Recogn., 38(9):1332–1348, September 2005.

[Ha02] Hagerer, A.; Hungar, H.; Niese, O.; Steffen, B.: Model Generation by Moderated Regular Extrapolation. In (Kutsche, R.-D.; Weber, H., Hrsg.): Fundamental Approaches to Software Engineering, Jgg. 2306, S. 80–95. Springer Berlin / Heidelberg, 2002.

[Ho12] Howar, F.: Active Learning of Interface Programs. Dissertation, TU Dortmund, 2012.

[IHS14a] Isberner, M.; Howar, F.; Steffen, B.: Learning register automata: from languages to program structures. Machine Learning, 96(1-2):65–98, 2014.

[IHS14b] Isberner, M.; Howar, F.; Steffen, B.: The TTT Algorithm: A Redundancy-Free Approach to Active Automata Learning. In (Bonakdarpour, B.; Smolka, S. A., Hrsg.): Runtime Verification, Jgg. 8734 in Lecture Notes in Computer Science, S. 307–322. Springer International Publishing, 2014.

[IHS15] Isberner, M.; Howar, F.; Steffen, B.: The Open-Source LearnLib. In (Kroening, Daniel; Păsăreanu, Corina S., Hrsg.): Computer Aided Verification, Jgg. 9206 in Lecture Notes in Computer Science, S. 487–495. Springer International Publishing, 2015.

[Ir12] Irfan, M. N.: Analysis and optimization of software model inference algorithms. Dissertation, Université de Grenoble, Grenoble, Frankreich, September 2012.

[Is09] Issarny, V.; Steffen, B.; Jonsson, B.; Blair, G. S.; Grace, P.; Kwiatkowska, M. Z.; Calinescu, R.; Inverardi, P.; Tivoli, M.; Bertolino, A.; Sabetta, A.: CONNECT Challenges: Towards Emergent Connectors for Eternal Networked Systems. In: ICECCS. IEEE Computer Society, S. 154–161, Juni 2009.

[Is15] Isberner, M.: Foundations of Active Automata Learning: An Algorithmic Perspective. Dissertation, TU Dortmund, September 2015.

[KV94] Kearns, M. J.; Vazirani, U. V.: An Introduction to Computational Learning Theory. MIT Press, Cambridge, MA, USA, 1994.

[Ne58] Nerode, A.: Linear Automaton Transformations. Proceedings of the American Mathematical Society, 9(4):541–544, 1958.

[PVY99] Peled, D.; Vardi, M. Y.; Yannakakis, M.: Black Box Checking. In (Wu, J.; Chanson, S. T.; Gao, Q., Hrsg.): Formal Methods for Protocol Engineering and Distributed Systems, Jgg. 28 in IFIP, S. 225–240. Springer US, 1999.

[RS93] Rivest, R. L.; Schapire, R. E.: Inference of finite automata using homing sequences. Inf. Comput., 103(2):299–347, 1993.

Malte Isberner wurde am 3. Mai 1988 in Duisburg geboren. Im Jahr 2006 begann er das Studium der Informatik an der TU Dortmund, welches er im August 2011 *mit Auszeichnung* abschloss. Während seiner Promotion am *Lehrstuhl für Programmiersysteme* von Prof. Dr. Bernhard Steffen beschäftigte er sich eingehend mit theoretischen und praktischen Aspekten aktiven Automatenlernens. Im Rahmen dieser Arbeit wurde er zum Hauptentwickler der Automatenlernbibliothek *LearnLib*, die auf der renommierten *CAV*-Konferenz 2015 mit dem *Best Artifact Award* ausgezeichnet wurde. Nach Auslandsaufenthalten an der Universität Uppsala, der Carnegie Mellon University sowie dem Cybersecurity-Startup *FireEye* schloss er seine Promotion im September 2015 mit der Note *ausgezeichnet (summa cum laude)* ab. Seitdem arbeitet er als Software Engineer bei Google im kalifornischen Mountain View.