Symmetrische und effiziente Synthese*

Rüdiger Ehlers
ruediger.ehlers@uni-bremen.de
Universität Bremen

Abstract: Bei der reaktiven Synthese wird der Prozess der Konstruktion eines reaktiven Systems vereinfacht, indem dieses anhand einer formalen Spezifikation automatisch erzeugt wird. Trotz der offensichtlichen Vorteile dieses Konzeptes gegenüber der manuellen Konstruktion eines reaktiven Systems erzeugen aktuelle Syntheseansätze noch zu unstrukturierte Lösungen nach zu langer Rechenzeit, so dass reaktive Synthese noch nicht als Teil aktueller Systementwurfsvorgehensmodelle etabliert ist.

Die Dissertation „Symmetrische und effiziente Synthese“ behandelt sowohl das Problem der mangelnden Struktur als auch die zu verbessernde Effizienz der reaktiven Synthese. Zuerst wird die Synthese symmetrisch strukturierter Systeme betrachtet, welche dadurch, dass sie aus mehreren Instanzen desselben Prozesses bestehen, bessere Eigenschaften haben. Es werden Systemarchitekturen mit entscheidbaren und unentscheidbaren symmetrischen Syntheseproblemen charakterisiert. Für eine große Klasse von Architekturen wird ein komplexitätstheoretisch optimaler Synthesealgorithmus vorgestellt.

Zur Erhöhung der Effizienz der reaktiven Synthese wird ein aktueller Synthesealgorithmus, der seine praktische Anwendbarkeit bereits unter Beweis gestellt hat, um eine Klasse von unterstützten Systemeigenschaften erweitert, ohne dass er seine Effizienz verliert. Diese Erweiterung erlaubt auch die Synthese robuster Systeme, die sogar in unerwarteten Umgebungen geeignetes Verhalten zeigen. Außerdem wird der ACTL $\cap$ LTL Syntheseansatz vorgestellt, der die Effizienz des bereits erprobten Synthesealgorithmus in eine weit expressivere Spezifikationslogik überträgt.

1 Einleitung

Da Fehler in sicherheitskritischen reaktiven Systemen gravierende Auswirkungen haben können, wird formalen Methoden, die die Korrektheit eingebetteter Systeme sicherstellen können, eine große Bedeutung beigemessen. Ein reaktives System ist eine Datenverarbeitungseinheit, die in jedem Schritt ihrer Berechnung ein Eingabedatum einliest und ein passendes Ausgabedatum erzeugt. Die Berechnung terminiert nicht, d. h. das System hat keinen Zeitpunkt, zu dem die Berechnung abgeschlossen ist. Beispiele für solche Systeme sind Ampelsteuerungen, Steuerungseinheiten in Luft- und Landfahrzeugen sowie Maschinensteuerungen. Da das System nicht-terminierend ist, befasst sich die Spezifikation desselben mit dessen Verhalten während der Laufzeit, anstatt Bedingungen über die Ausgabe bei Berechnungsabschluss anzugeben.

Der meistverbreitete Ansatz zum Sicherstellen der Korrektheit reaktiver Systeme ist *Verifikation*, bei dem der Korrektheitsbeweis eines Systems erbracht wird, nachdem dieses

*Englischer Titel der Dissertation: “Symmetric and Efficient Synthesis”

konstruiert wurde. Verifikation beseitigt jedoch nicht die inhärente Schwierigkeit, korrekte Systeme zu konstruieren und bietet deshalb nur eine unbefriedigende Lösung. Zur Lösung dieses Problems wurde der Ansatz der *reaktiven Synthese* vorgeschlagen, welcher es dem Ingenieur/der Ingenieurin eines reaktiven Systems erlaubt, sich auf die Beschreibung der Spezifikation zu konzentrieren, von welcher die Systemimplementierung dann automatisch erzeugt wird. Ausgehend von der ersten Formulierung des Syntheseproblems für reaktive Systeme durch Church [Chu62] wurde in den letzten 40 Jahren eine Vielzahl von Spezifikationsformalismen und passenden Synthesemethoden entwickelt.

Trotz des Fortschritts der Forschung auf dem Gebiet der Synthese und der Attraktivität des Konzeptes ist dessen Einfluss auf die Praxis des Systementwurfs momentan eher als gering einzustufen. Für diese Diskrepanz gibt es eine Reihe von Gründen. Einer davon ist, dass die Qualität automatisch erzeugter Implementierungen im Allgemeinen vergleichbar mit der manuell erzeugter sein soll, und somit eine ähnliche Verständlichkeit, Wartbarkeit und technische Umsetzbarkeit haben soll. Aktuelle Synthesealgorithmen garantieren jedoch nicht, Implementierungen mit diesen Eigenschaften zu berechnen, und es zeigte sich, dass die Algorithmen dies auch in der Praxis nicht tun. Weil jedoch diese Qualitätseigenschaften schwer zu formalisieren sind, ist es am vielversprechendsten anzusehen, stattdessen Anforderungen an die Struktur der zu synthetisierenden Systeme in den Syntheseprozess aufzunehmen, die diese Eigenschaften implizieren.

Orthogonal zum Aspekt der Struktur synthetisierter Systeme ist die Frage der Skalierbarkeit des Syntheseprozesses. Aufgrund der hohen Komplexität des Syntheseproblems wurde von vielen Wissenschaftlern die Idee der Synthese als inpraktikabel verworfen. Beispielsweise wurde für Linearzeitlogik (LTL, [Pnu77]) als Spezifikationssprache gezeigt, dass das zugehörige Syntheseverfahren eine doppelt-exponentielle Zeitkomplexität hat. Dennoch zeigen die vergangenen Jahre ein Wiederaufleben des Interesses an reaktiver Synthese. Der Hauptgrund hierfür ist die Beobachtung, dass viele Spezifikationen in der Praxis entweder eine simple Struktur oder eine simple Lösung haben. Dies widerspricht der Komplexitätsanalyse nicht: Die doppelt-exponentielle Zeitkomplexität ist tatsächlich nur ein Zeichen dafür, dass der Spezifikationsformalismus es erlaubt, kompakte Spezifikationen zu schreiben, dessen kleinste Implementierungen doppelt-exponentiell in der Länge der Spezifikation groß sind. Wenn wir annehmen, dass solche Spezifikationen nicht als Eingabe für ein Syntheseverfahren verwendet werden, können Lösungstechniken angewandt werden, die auf die praktisch relevanten Spezifikationsfälle optimiert sind.

Die Syntheseansätze, die dieser Idee folgen, können grob in zwei Forschungsrichtungen unterteilt werden. Die erste dieser Richtungen befasst sich mit der Effizienzverbesserung der Synthese für vollständiges LTL. Dabei betrachtet sie eine Vielzahl von Techniken, wie z. B. solchen, die auf dem Lösen von Paritätsspielen basieren [SSR08], bishin zu Ansätzen wie *Bounded Synthesis* [FS07]. In der zweiten Forschungsrichtung wird die volle Expressivität von LTL gegen die Möglichkeit der Optimierung auf ein wohlgewähltes Subfragment von LTL eingetauscht, um die Skalierbarkeit der Synthese substanziell zu verbessern. Dieser Ansatz schränkt die Menge der Spezifikationen, die betrachtet werden können, ein. Trotzdem gibt es eine große Zahl praktischer Anwendungen, in denen Implementierungen mit ihm erfolgreich synthetisiert wurden (z.B. [BGJ$^+$07, OTMW11]). Dies zeigt, dass eine gute Auswahl der Spezifikationslogik es erlaubt, ein passendes Syntheseverfahren zu

entwickeln, welches die Effizienz und Expressivität in der Spezifikation, die in praktischen Anwendungen benötigt wird, kombiniert.

Die Doktorarbeit „Symmetrische und effiziente Synthese“ beschreibt neue Resultate und Techniken, die sowohl das Problem der ungenügenden Struktur synthetisierter reaktiver Systeme angehen als auch die Skalierbarkeit aktueller Syntheseverfahren für praktische Spezifikationen verbessern. Darüber hinaus enthält der Einführungsteil der Arbeit eine problemorientierte Einführung in das Gebiet der reaktiven Synthese und eine Beschreibung eines modernen Syntheseansatzes, einschließlich der vollständigen Herleitung und Motivation für dessen Komponenten. Dieser Einführungsteil ist in sich abgeschlossen und damit auch für reine Lehrzwecke geeignet.

2 Symmetische Synthese

Struktur in reaktiven Systemen kann viele Formen haben. So kann ein System kompakt durch ein kurzes Programm oder durch einen Schaltkreis geringer Tiefe dargestellt werden. In der bestehenden Literatur finden sich Syntheseverfahren zum Erzeugen explizit repräsentierter Systeme mit wenigen Zuständen [FS07], kleiner Schaltkreise [EKH12] und auch verteilter Systeme [PR90], bei denen ein reaktives System aus mehreren Subsystemen besteht, die durch eine vordefinierte Kommunikationsstruktur die Spezifikationen in Kooperation erfüllen.

In all diesen Arbeiten wurden jedoch keine Eigenschaften über die interne Struktur der Lösungsteile betrachtet. So präferieren Ingenieure z. B. *symmetrische* Systeme, d. h. Implementierungen, die aus multiplen Instanzen derselben Subimplementierung bestehen.

Die symmetrische Implementierbarkeit reaktiver Systeme ist eine klassische Forschungsfrage der Informatik. Das Philosophenproblem, bei dem eine Menge von n Philosophen sich bei der Benutzung ebensovieler Besteckteile koordinieren müssen, ist das wohl bekannteste Szenario in welchem man sich für die symmetrische Implementierbarkeit einer Spezifikation interessiert. Für ein reaktives System, welches einen kontinuierlichen Eingabestrom liest und gleichzeitig einen Ausgabestrom erzeugt, bedeutet eine symmetrische Implementierung, dass alle Subkomponenten parallel und zur selben Zeit laufen. Symmetrische Systeme sind oft einfacher zu handhaben als monolithische Systeme, da die Symmetrieanforderung garantiert, dass das System ohne zentrale Koordinationskomponente auskommt. Außerdem sind symmetrische Systeme typischerweise einfacher zu warten als andere verteilte Systeme, da sie weniger Komponenten haben und oft auch einfacher zu verstehen, da die Komponenten in einer Art und Weise agieren müssen, die eine zentrale Koordination unnötig machen. Zuguterletzt sei angemerkt, dass es spezielle Verifikationsalgorithmen gibt, die Symmetrie in zu verifizierenden Systemen ausnutzen können, so dass Symmetrie hilfreich für die Zertifizierung von Systemen ist.

Bei dem bereits genannten Philosophenproblem ist das Interesse an symmetrischer Implementierbarkeit von eher theoretischer Natur. Es ist jedoch allgemein bekannt, dass es keine synchrone symmetrische Implementierung für die Philosophen gibt, bei der diese nicht verhungern, was die inhärenten Beschränkungen symmetrischer Implementierungen

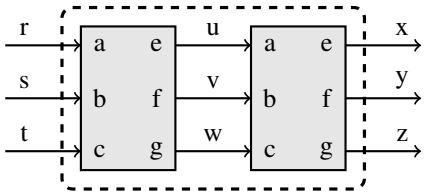

Abbildung 1: Grafische Darstellung der symmetrischen S2 Architektur

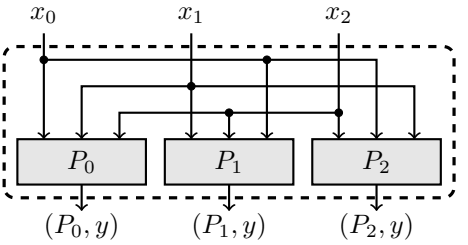

Abbildung 2: Eine einfache rotationssymmetrische Architektur.

zeigt. Um diese zu vermeiden, wurde eine breite Palette von Techniken zum *Brechen der Symmetrie* entwickelt. Typischerweise wird dies durch die Ausführung eines verteilten Algorithmus erreicht, nach dessen Terminierung ein Prozess als Koordinationsprozess bestimmt worden ist (auch „*leader election*“ genannt).

Die Forschung auf dem Gebiet der Symmetriebrechung ist gewissermaßen sehr pessimistisch, da sie dadurch motiviert ist, dass einige Anwendungen Symmetriebrechung benötigen. Dies trifft jedoch nicht auf alle Anwendungen zu. Das Einsparen des Symmetriebrechungsschrittes hat viele Vorteile. Beispielsweise benötigen Algorithmen zur Symmetriebrechung Kommunikationsverbindungen zwischen den Prozessen, die bei symmetrischen Implementierungen oft nicht nötig sind. Darüber hinaus verzögert die Wahl eines Koordinationsprozesses die Verfügbarkeit der Prozesse für deren Hauptaufgabe und vergrößert deren Implementierung. Daher sind Systemimplementierungen, die ohne Symmetriebrechnung auskommen, im Allgemeinen vorzuziehen. Dies wirft die Frage auf, mit welchen Methoden man für eine Anwendung eines verteilten Systems erkennen kann, ob Symmetriebrechnung überhaupt nötig ist.

2.1 Ergebnisse

Die Doktorarbeit „Symmetrische und effiziente Synthese“ beinhaltet die erste gründliche Analyse des Problems der Synthese symmetrischer Systeme. Die erste Kernfrage in diesem Kontext ist, welche *symmetrische Architekturen* entscheidbare Syntheseprobleme aufweisen. Eine Architektur definiert, mit welchen Signalen die Teilkomponenten, welche jeweils z. B. als Mealy- oder Moore-Maschinen dargestellt werden können und auch *Prozesse* genannt werden, interagieren können. Außerdem legt sie fest, wie die Ein- und Ausgabe des Gesamtsystems auf die Prozesse verteilt ist.

Abbildung 1 zeigt eine einfache Beispielarchitektur mit zwei Prozessen. Das symmetrische Syntheseproblem stellt sich in diesem Fall wie folgt dar: Gegeben ist eine Spezifikation über $r, s, \ldots, y, z$, die das gewünschte Verhalten des Gesamtsystems repräsentiert. Gesucht ist eine Prozessimplementierung, die die Eingabesignale a, b und c liest, auf die Ausgabesignale e, f und g schreibt und sicherstellt, dass das Gesamtsystem die Spezifikation erfüllt, wenn die Implementierung für beide Prozesse der Architektur verwendet wird.

Die Architektur wird als fest angenommen und ist nicht Teil der Eingabe des Syntheseproblems. Dies erlaubt es uns, zwischen Architekturen mit entscheidbaren und unentscheidbaren symmetrischen Syntheseproblemen zu unterscheiden.

Leider ist schon die sehr einfache *S2 Architektur* in Abbildung 1 für alle Spezifikationsformalismen von praktischer Relevanz unentscheidbar. Dies kann gezeigt werden, indem die Synthese für eine andere Architektur ohne Symmetriebedingung, in der die Prozesse somit unterschiedliche Implementierungen haben können, und für die von Pnueli und Rosner die Unentscheidbarkeit des Syntheseproblems gezeigt wurde [PR90], auf die symmetrische Synthese für die S2 Architektur reduziert wird. Die Archiektur von Pnueli und Rosner bezieht ihre Unentscheidbarkeit von der Tatsache, dass die beiden Prozesse die Eingabe für den jeweiligen anderen Prozess nicht lesen können. Durch geschickte Kodierung in der Spezifikation kann dies im symmetrischen Fall dadurch simuliert werden, dass die Prozesse nicht wissen, ob sie der erste oder der zweite Prozess der Kette sind.

Das Unentscheidbarkeitsergebnis kann dann für den Fall, dass die Prozesse jeweils nur ein Eingabe- und Ausgabesignal haben, generalisiert werden, indem die Werte der Signale serialisiert werden. Die Dissertation zeigt auch, wie dies mit *nicht-zählenden Spezifikationslogiken* wie z. B. LTL durchgeführt werden kann. Dadurch haben im Endeffekt alle Architekturen, die interne Kommunikation zwischen den Prozessen haben, ein unentscheidbares symmetrisches Syntheseproblem. Da jedoch eine der Motivationen für symmetrische Synthese das Einsparen ebensolcher Prozesskommunikation ist, stellt sich die Frage nach den verbleibenden Fällen.

Tatsächlich hat eine große Klasse von symmetrischen Architekturen ohne interne Prozesskommunikation ein entscheidbares Syntheseproblem für alle praktisch relevanten Spezifikationslogiken. In diesen so genannten *rotationssymmetrischen* Architekturen erhalten alle Prozesse die komplette Eingabe an das Gesamtsystem, jedoch in individuell rotierter Form. Abbildung 2 zeigt ein einfaches Beispiel. Architekturen von dieser Form erlauben die präzise Charakterisierung der Attribute derjenigen *Berechnungsbäume*, die das Verhalten eines rotationssymmetrischen Gesamtsystems darstellen, als sogenannte *Symmetrieeigenschaft*. Ein Berechnungsbaum beschreibt die Menge der möglichen Berechnungen eines reaktiven Systems. Der Baum verzweigt bezüglich der Eingabe an das System, von der wiederum die Ausgabe des Systems abhängen kann, mit dem die Knoten des Baumes markiert sind. Das Kernresultat, das die Synthese für rotationssymmetrische Gesamtsysteme erlaubt, ist die Dekomposition der Symmetrieeigenschaft in zwei Teile:

1. Die Bedingung, dass das zu synthetisierende System nicht spontan die Symmetrie durch seine Ausgabe bricht
2. Die Evaluation der Systemspezifikation und ihrer rotierten Versionen nur entlang *normalisierter* Eingabeworte an das System

Beide Bedingungen sind notwendig und für reaktive Systeme mit endlichem Zustandsraum, die wir bei der Synthese anstreben, auch hinreichend. Die erste Bedingung kann relativ leicht in einen reaktiven Syntheseprozess eingebettet werden, da sie z. B. in LTL beschrieben werden kann. Die zweite Bedingung wird durch Modifikation der *Baumautomaten*, die in Syntheseprozessen typischerweise verwendet werden, implementiert. Im Fall einer positiven Realisierbarkeitsantwort durch den Synthesealgorithmus ist dann nur

noch ein Nachbearbeitungsschritt der Implementierung erforderlich. In diesem wird das Systemverhalten für nicht-normalisierte Eingabeströme durch Rotationen des Systemverhaltens für normalisierte Eingabeströme ersetzt. Die Zeitkomplexität des Syntheseverfahrens für rotationssymmetrische Systeme ist exponentiell in der Anzahl der Prozesse für alle verbreiteten Spezifikationsformalismen. Eine Komplexitätsanalyse, die zeigt, dass der Algorithmus optimal ist, komplettiert die Betrachtung der symmetrischen Synthese.

3 Effiziente Synthese

Schon die ersten Lösungen zum Problem der reaktiven Synthese erlauben es, sehr expressive Spezifikationsformalismen zu verwenden [BL69]. Die hohe Komplexität des Syntheseproblems für eben diese Formalismen motiviert jedoch die Betrachtung schwächerer Spezifikationslogiken, die sich besser für effiziente Synthesealgorithmen eignen.

Unter denjenigen Ansätzen, die in diese Klasse fallen, ist *Generalised Reactivity(1) Synthese* [PPS06], welche typischerweise als *GR(1) Synthese* abgekürzt wird, die wohl meistverbreitete. In diesem Kontext besteht die Spezifikation aus *Annahmen* und *Garantien*. Erstere beschreiben die Eigenschaften der Umgebung des zu synthetisierenden reaktiven Systems, welche als gegeben angenommen werden können. Letztere beschreiben die gewünschten Systemeigenschaften. Um eine niedrige Komplexität des Syntheseprozesses zu erreichen, sind die erlaubten Formen der Annahmen und Garantien, verglichen mit der Temporallogik LTL, in welcher diese klassischerweise formuliert werden, syntaktisch eingeschränkt. Eine Vielzahl von Arbeiten zeigt jedoch, dass es möglich ist, unter diesen Einschränkungen erfolgreich Systeme mit praktischer Relevanz zu synthetisieren (siehe z.B. [BGJ+07, WTM10, OTMW11]). Leider hat der GR(1) Syntheseansatz dennoch zwei substanzielle Einschränkungen: *ungenügende Expressivität* und die Notwendigkeit zur *Präsynthese*.

3.1 Ergebnisse

Obwohl viele Eigenschaften in Spezifikationen praktischer reaktiver Systeme als Kombination von Eigenschaftstypen, die bei der GR(1) Synthese verwendet werden können, darstellbar sind, trifft dies auf eine wichtige Klasse von Spezifikationsteilen nicht zu: *Stabilitätseigenschaften*. Diese beschreiben, dass ab einem beliebigen Zeitpunkt der Ausführung eines reaktiven Systems eine gegebene einfache Subeigenschaft immer erfüllt sein muss. Ein Beispiel hierfür ist die Bedingung, dass ein reaktives System nur am Anfang seiner Ausführung eine endliche Anzahl von Reaktionsfristen auf bestimmte Stimuli verpassen darf, d. h., ab einem beliebigen Zeitpunkt ein Normalbetriebszustand erreicht ist.

Im Rahmen der Doktorarbeit „Symmetric and Efficient Synthesis“ wurde eine Erweiterung des GR(1) Syntheseverfahrens entwickelt, welche die Menge der erlaubten Eigenschaften in den Annahmen und Garantien um Stabilitätseigenschaften ergänzt. Die Erweiterung

behält diejenigen Attribute der GR(1) Synthese bei, die dessen effiziente Implementierung erlauben. Dies ist zum einen die Reduktion des Syntheseproblems auf das Lösen von Paritätsspielen mit einer konstanten Anzahl an Farben und einer Spielgröße, die nur exponentiell in der Anzahl der Propositionen in der Spezifikation ist. Zum anderen ist dies die Möglichkeit einer *symbolischen* Lösung dieses Spiels, z. B. mit binären Entscheidungsdiagrammen (BDDs). Darüber hinaus wird gezeigt, dass jede weitere substanzielle Erweiterung der unterstützten Eigenschaftsklassen für die Annahmen und Garantien dazu führen würde, dass diese Vorteile der GR(1) Synthese verloren gingen (sofern P$\neq$NP gilt).

Stabilitätseigenschaften komplettieren den GR(1) Syntheseansatz, dessen Erweiterung den Namen *generalisierte Rabin(1) Synthese* erhielt, auf zwei Arten. So charakterisieren Wongpiromsarn et al. [WTM10] die Eigenschaftstypen, die in Robotikanwendungen für die Annahmen und Garantien benötigt werden. Stabilitätseigenschaften waren die letzte Klasse, die bei der GR(1) Synthese noch fehlte. Darüber hinaus ist die Spezifikationsklasse mit der Erweiterung um Stabilitätseigenschaften nun unter *Härtung* abgeschlossen. Hierbei wird eine Systemspezifikation in eine neue Spezifikation übersetzt, die dafür sorgt, dass sich das zu synthetisierende System im Falle der Verletzung der gemachten Annahmen in einer robusten Art und Weise verhalten muss, d. h., nach kurzer Zeit wieder in den Normalbetriebszustand zurückkehren muss. Dies ist in unvorhergesehenen Betriebsbedingungen von Relevanz, da normalerweise keine Bedingungen an das Verhalten des reaktiven Systems gestellt werden, die nach Verletzung der Annahmen gelten müssen.

Neben der Erweiterung des GR(1) Syntheseansatzes um Stabilitätseigenschaften wurde im Rahmen der Doktorarbeit auch die Frage behandelt, wie dem Problem der *Präsynthese* begegnet werden kann. Dieser Begriff beschreibt die Aktivität, Spezifikationsteile in die von der GR(1) Synthese benötigte spezielle syntaktische Form zu überführen. Es hat sich gezeigt, dass es für die Effizienz des Syntheseverfahrens von großer Relevanz ist, wie genau eine Spezifikation in diese Form überführt wird. Als Konsequenz hieraus wird dieser Schritt typischerweise per Hand ausgeführt.

Dieser manuelle Schritt steht im starken Gegensatz zum Grundanspruch der Synthese, die Konstruktion von reaktiven Systemen vollständig zu automatisieren. Zur Behebung dieses Defizits wurde im Rahmen der Doktorarbeit der *ACTL $\cap$ LTL Syntheseansatz* entwickelt. Kernidee ist es, als Annahmen und Garantien alle Eigenschaften zuzulassen, deren Erfüllung in einem Berechnungsbaum sowohl in der Temporallogik ACTL (*Computation Tree Logic* ohne existenzielle Pfadquantoren) als auch in LTL darstellbar sind. Wie von Maidl [Mai00] gezeigt wurde, sind dies genau diejenigen Eigenschaften, die als *universelle sehr schwache Büchiautomaten* repräsentiert werden können.

Der Vorteil der Darstellung von Spezifikationen als universelle sehr schwache Automaten ist, dass diese effizient in symbolisch repräsentierte Synthesespiele übersetzt werden können. Dadurch sind sie gut als Zwischenebene für Spezifikationen in einer geeigneten Logik und Synthesespielen geeignet. Um die Vorteile dieses Automatentyps in praktischen Syntheseansätzen zu nutzen, bedarf es jedoch eines Verfahrens, um aus einer geeigneten Spezifikationssprache in Automaten des Typs zu übersetzen. Maidl [Mai00] zeigte für ACTL, wie dies funktioniert. Für praktische Anwendungen ist eine Linearzeitlogik jedoch wesentlich besser geeignet. Ein entsprechender Algorithmus hierzu war bisher noch unbekannt und wurde im Rahmen der Dissertation entwickelt.

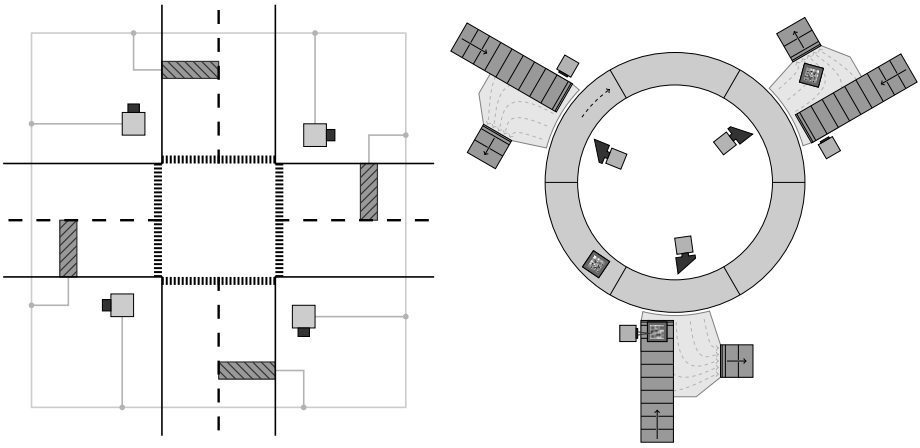

Abbildung 3: Grafische Darstellung zweier Anwendungen, für die reaktive Steuerungseinheiten symmetrisch implementiert werden können.

Die resultierende Konstruktion stellt auch den ersten Algorithmus dar, um eine Formel in LTL nach ACTL zu übersetzen, wann immer dies möglich ist. Eine Konstruktion für die Rückrichtung [CD88] wird mittlerweile in vielen Vorlesungen über formale Methoden gelehrt – nur die Hinrichtung war noch ungelöst. Mit den Resultaten zur ACTL $\cap$ LTL Synthese wurde diese Lücke nun geschlossen.

4 Zusammenführung und experimentelle Evaluation

Die Resultate zur symmetrischen Synthese, zur generalisierten Rabin(1) Synthese und zur ACTL $\cap$ LTL Synthese, die im Rahmen der Doktorarbeit erzielt wurden, können einzeln verwendet werden, aber insbesondere auch in Kombination. Dies erlaubt zum Beispiel die effiziente Synthese robuster Systeme mit Annahmen und Garantien in ACTL $\cap$ LTL.

Zur Evaluation der praktischen Nutzbarkeit der in der Dissertation vorgestellten Verfahren wurden diese in Kombination anhand von zwei Fallstudien bewertet. Abbildung 3 stellt diese grafisch dar. Es handelt sich im ersten Fall um ein Ampelsystem, das Sensoren für wartende Autos besitzt, sowie um einen Rotationssortierer, der Pakete mit Hilfe eines permanent rotierenden Verteiltisches in die jeweils richtigen Ausgangsrutschen schieben muss. Beide Fallstudien sind natürliche Anwendungsgebiete für symmetrische Systeme.

Die Experimente zeigen, dass die Anwendung symmetrischer Synthese auch Einblick in die inhärenten Eigenschaften eines verteilten Systems bietet. Beispielsweise zeigt sich beim Ampelsystem, dass es ausreichend ist, dass Symmetrie *partiell* gebrochen wird, damit das System dessen Funktion erfüllen kann. Alle durchgeführten Experimente benötigen höchstens ein paar Minuten auf aktuellen Rechnern und zeigen somit die praktische Anwendbarkeit der vorgestellten Techniken.

5 Zusammenfassung und Ausblick

Die Dissertation „Symmetrische und effiziente Synthese“ stellt Lösungen für die schnelle und automatische Synthese strukturierter reaktiver Systeme vor. Zur Verbesserung der Struktur synthetisierter Systeme wird das Problem der symmetrischen Synthese betrachtet und für eine große Klasse von Architekturen gelöst. Der Synthesealgorithmus besitzt eine optimale Zeitkomplexität. Für Architekturen, die Kommunikation zwischen den Prozessen enthalten, wird hingegen bewiesen, dass diese ein unentscheidbares Syntheseproblem aufweisen.

Zur Verbesserung der Effizienz der reaktiven Synthese wird ein effizienter Syntheseansatz, der seine Anwendbarkeit schon mehrfach unter Beweis gestellt hat, der aber nicht alle in praktischen Anwendungen nützlichen Spezifikationsteilarten unterstützt, um Stabilitätseigenschaften erweitert. Dies erlaubt insbesondere die Synthese *robuster* Systeme, die sich auch dann noch sinnvoll verhalten, wenn die in der Spezifikation gegebenen Annahmen über die Umgebung des Systems verletzt werden. Die Expressivitätserweiterung um Stabilitätseigenschaften ist die nachweisbar substanziell letztmögliche, ohne dass die guten Eigenschaften des Algorithmus verloren gehen (sofern P$\neq$NP gilt).

Darüber hinaus wird im Rahmen der Dissertation gezeigt, wie der Kodierungsprozess einer Spezifikation in eine Form, die von effizienten und vergleichsweise einfachen Synthesealgorithmen verwendbar ist, automatisiert werden kann. Dabei werden alle Eigenschaften in Spezifikationen unterstützt, die sowohl in der Temporallogik LTL, als auch in ACTL geschrieben werden können. Die Anwendung der resultierenden universellen sehr schwachen Automaten in Synthesespielen ist einfach und erlaubt die Benutzung symbolischer Berechnungstechniken.

Die Resultate über die symmetrische und effiziente Synthese reaktiver Systeme werden durch zwei Fallstudien komplementiert, in welchen die neu entwickelten Techniken in Kombination angewandt werden. Darüber hinaus enthält die Dissertation eine abgeschlossene leicht zugängliche Einführung in das Gebiet der reaktiven Synthese, welche auch ohne Betrachtung der erzielten Forschungsergebnisse von Interesse ist.

Neben der Erweiterung des Wissens über Methoden und Konzepte der reaktiven Synthese gibt die Dissertation auch einen Ausblick auf zukünftige Forschung auf dem Gebiet: Sie zeigt, dass nichtreguläre Eigenschaften zu synthetisierender Systeme, wie hier die Symmetrieeigenschaft, durch *Dekomposition* dieser Eigenschaften in den Syntheseprozess integriert werden können. Weiterhin zeigt sie, dass durch geschickte Wahl eines Automatenmodells als Zwischenrepräsentation für die Spezifikation in der Synthese Effizienz und Expressivität kombiniert werden können. Das in der Dissertation verwendete Automatenmodell ist z.B. sehr gut geeignet, um neuartige Repräsentationen für Spielpositionsmengen während der Synthesespiellösung zu untersuchen und somit die Syntheseeffizienz in der Zukunft weiter zu erhöhen. Schließlich gibt die Dissertation einen Ausblick auf zukünftige Lösungen zur Synthese von Systemen *hoher Qualität*, die sowohl auf algorithmischer Modifikation der Spezifikation basieren, als auch auf der Anwendung von Techniken, die eine fundamentalere Änderung des Synthesealgorithmus erfordern.

Literatur

[BGJ+07] Roderick Bloem, Stefan Galler, Barbara Jobstmann, Nir Piterman, Amir Pnueli und Martin Weiglhofer. Specify, Compile, Run: Hardware from PSL. *Electr. Notes Theor. Comput. Sci.*, 190(4):3–16, 2007.

[BL69] J. Richard Büchi und Lawrence H. Landweber. Definability in the Monadic Second-Order Theory of Successor. *J. Symb. Log.*, 34(2):166–170, 1969.

[CD88] Edmund M. Clarke und I. A. Draghicescu. Expressibility results for linear-time and branching-time logics. In *REX Workshop*, Seiten 428–437, 1988.

[Chu62] Alonzo Church. Logic, Arithmetic and Automata. In *Intl. Congr. Math.*, Seiten 23–25, 1962.

[Ehl13] Rüdiger Ehlers. *Symmetric and Efficient Synthesis*. Dissertation, Universität des Saarlandes, 2013.

[EKH12] Rüdiger Ehlers, Robert Könighofer und Georg Hofferek. Symbolically Synthesizing Small Circuits. In *FMCAD*, Seiten 91 – 100, 2012.

[FS07] Bernd Finkbeiner und Sven Schewe. SMT-Based Synthesis of Distributed Systems. In *Automated Formal Methods (AFM)*, 2007.

[Mai00] Monika Maidl. The Common Fragment of CTL and LTL. In *FOCS*, 2000.

[OTMW11] Necmiye Ozay, Ufuk Topcu, Richard M. Murray und Tichakorn Wongpiromsarn. Distributed Synthesis of Control Protocols for Smart Camera Networks. In *ICCPS*, 2011.

[Pnu77] Amir Pnueli. The Temporal Logic of Programs. In *FOCS*, Seiten 46–57. IEEE, 1977.

[PPS06] Nir Piterman, Amir Pnueli und Yaniv Sa'ar. Synthesis of Reactive(1) Designs. In *VMCAI*, Seiten 364–380, 2006.

[PR90] Amir Pnueli und Roni Rosner. Distributed Reactive Systems Are Hard to Synthesize. In *FOCS*, Seiten 746–757. IEEE Computer Society, 1990.

[SSR08] Saqib Sohail, Fabio Somenzi und Kavita Ravi. A Hybrid Algorithm for LTL Games. In *VMCAI*, Seiten 309–323, 2008.

[WTM10] Tichakorn Wongpiromsarn, Ufuk Topcu und Richard M. Murray. Receding horizon control for temporal logic specifications. In *HSCC*, Seiten 101–110, 2010.

Rüdiger Ehlers, geboren 1982, studierte von 2002 bis 2006 Informatik an der Technischen Universität Dortmund. Nach dem Abschluss als Diplom-Informatiker und einem zusätzlichen Master-Studium über die Schnittstelle zwischen Mathematik und Informatik an der University of Oxford in England begann er ein Promotionsstudium an der Universität des Saarlandes, das er 2013 abschloss. Von September 2012 bis August 2013 war er kollaborativer Wissenschaftler der University of California at Berkeley und der Cornell University in den Vereinigten Staaten von Amerika. Nach einer anschließenden Beschäftigung als postdoktoraler wissenschaftlicher Mitarbeiter an der Universität Kassel ist er seit Februar 2014 Nachwuchsgruppenleiter am Fachbereich Informatik der Universität Bremen.