SMT-Solving, Interpolation und Quantoren[1]

Tanja Schindler[2]

Abstract: Satisfiability Modulo Theories, kurz *SMT*, bezeichnet das Problem, ob eine Formel in einer sogenannten Theorie, die die erlaubten Symbole und deren Interpretation einschränkt, erfüllbar ist. SMT-Solver werden zum Beispiel in der Softwareverifikation, Testgenerierung oder Programmsynthese eingesetzt. In der hier zusammengefassten Dissertation [Sc22] werden verschiedene Methoden vorgestellt, die Probleme im Bereich des SMT-Solving lösen. Zum einen wird eine Entscheidungsprozedur für die Arraytheorie mit Constant Arrays vorgestellt, welche sich zum Beispiel zur Modellierung von initialisiertem Speicher eignen. Darauf aufbauend wird eine Interpolationsmethode für diese Theorie entwickelt. Interpolanten werden beispielsweise eingesetzt, um in der Softwareverifikation Kandidaten für Schleifeninvarianten zu generieren. Zuletzt wird eine Instanziierungsmethode vorgestellt, um quantifizierte Probleme zu lösen, die zum Beispiel bei der Verifikation von Sortieralgorithmen auftreten. Alle vorgestellten Methoden sind im Open-Source SMT-Solver SMTInterpol implementiert.

1 Einführung

Satisfiability Modulo Theories, kurz *SMT*, bezeichnet das Entscheidungsproblem, ob eine prädikatenlogische Formel erster Stufe in einer gegebenen Theorie erfüllbar ist. Die Theorie legt dabei fest, welche Symbole in der Formel vorkommen dürfen und wie sie interpretiert werden müssen. So lassen sich Aussagen zum Beispiel über Ganzzahlen oder Arrays formulieren. Programme, die dieses Problem lösen, werden SMT-Solver genannt. Sie werden zum Beispiel im Bereich der Softwareverifikation, der Testgenerierung, oder der Programmsynthese eingesetzt.

Beispielsweise kann das Problem, ob ein Fehler über einen bestimmten Programmpfad erreichbar ist, in eine Formel in einer geeigneten Kombination von Theorien übersetzt werden. Ist die Formel erfüllbar, bedeutet dies, dass es eine fehlerhafte Programmausführung gibt. Um die Erfüllbarkeit einer Formel zu entscheiden, wird in den meisten SMT-Solvern der DPLL($\mathcal{T}$)-Algorithmus [Ga04] eingesetzt. Dabei wird ein Modell für die aussagenlogische Abstraktion der Eingabeformel in konjunktiver Normalform gesucht, indem den Literalen (genauer gesagt, den Atomen, die in der Abstraktion als aussagenlogische Variablen betrachtet werden) der Formel nach und nach Wahrheitswerte zugewiesen werden. Dies geschieht entweder durch *Unit Propagation* aus einer Klausel, in der allen anderen Literalen bereits der Wert falsch zugewiesen wurde und wegen der deshalb ein bestimmter Wert an das übrige Literal zugewiesen werden muss, oder durch eine *Entscheidung*. Spezialisierte Theorie-Solver überprüfen, ob die aus der Literalbelegung resultierende konjunktive Formel

[1] Englischer Titel der Dissertation [Sc22]: „SMT Solving, Interpolation, and Quantifiers“

[2] Université de Liège, Belgien, Tanja.Schindler@uliege.be

in der Theorie erfüllbar ist. Dafür implementieren sie, sofern die Theorie entscheidbar ist, eine Entscheidungsprozedur. Findet ein Theorie-Solver einen Theoriekonflikt, also Literale, die zusammen in der Theorie $\mathcal{T}$ zum Widerspruch führen, so liefert er ein *Theorielemma* zurück, das ausschließt, dass diese Literale gleichzeitig gelten. Dieses Lemma wird in die Suche miteingebunden, nachdem eine für den Konflikt verantwortliche Entscheidung rückgängig gemacht wurde. Effiziente Theorie-Solver können die Suche nach einem aussagenlogischen Modell auch unterstützen, indem sie selbst Propagationen liefern, also Literale, die unter dem aktuellen aussagenlogischen Modell in der Theorie wahr sein müssen, und können so falsche Entscheidungen vermeiden. Theoriepropagationen sind zudem wichtig, um effizient mehrere Theorien kombinieren zu können.

In der Dissertation wird eine Entscheidungsprozedur für eine Erweiterung der Arraytheorie mit sogenannten *Constant Arrays* [St01] vorgestellt. Die Arraytheorie legt fest, wie sich Schreib- und Lesezugriffe auf Arrays verhalten, und kann in der Programmverifikation beispielsweise verwendet werden, um Speicher oder Array-Datenstrukturen zu modellieren. Constant Arrays, die an jeder Stelle das gleiche Element speichern, können dabei initialisierten Speicher darstellen. Existierende Entscheidungsprozeduren für Constant Arrays erzeugen viele neue Formeln, und insbesondere neue Terme, was problematisch für Interpolation ist. Die vorgestellte Entscheidungsprozedur erweitert eine existierende Entscheidungsprozedur [CH15], die auf sogenannten *Weak-Equivalence-Graphen* basiert, die effizient Schreibrelationen zwischen Arrays darstellen. Zwei Arrays, die durch Schreibketten verbunden sind, werden weak equivalent genannt. Dank der Graphen reichen wenige Regeln, um Konflikte in der Arraytheorie zu finden. Dabei müssen keine neuen Terme erzeugt werden, und auch nur diejenigen Formeln, die im Konflikt zu einem aktuellen partiellen Modell stehen. Im Rahmen der Dissertation wurden diese Regeln für Constant Arrays erweitert, und die Korrektheit der Entscheidungsprozedur bewiesen. Dieser Teil der Dissertation basiert auf der Publikation [HS19] und wird in Abschnitt 2 weiter beschrieben.

Für den Korrektheitsbeweis eines Programms benötigen Programmverifikationswerkzeuge induktive Schleifeninvarianten, also Formeln, die zu Beginn einer Schleife und dann nach jedem Schleifendurchlauf gelten. Gute Schleifeninvarianten zu finden ist meist der schwierigste Teil der Programmverifikation. Hierbei können sogenannte *Craig-Interpolanten* helfen. Für eine unerfüllbare Konjunktion $A \wedge B$ ist eine Craig-Interpolante eine Formel, die aus A folgt, die B widerspricht, und die nur Symbole enthält, die sowohl in A als auch in B vorkommen. Interpolanten können schrittweise aus Resolutionsbeweisen für die Unerfüllbarkeit einer Formel erzeugt werden. Als Kandidaten für Schleifeninvarianten eignen sich Interpolanten, die aus einem Beweis dafür, dass ein Pfad mit einer festen Anzahl von Schleifendurchläufen nicht ausführbar ist, generiert werden [Mc03]. Um Unentscheidbarkeit beim Überprüfen der Induktivitätseigenschaft von Invariantenkandidaten zu vermeiden, ist es hilfreich, wenn die Interpolanten quantorenfrei sind. Existierende Interpolationsalgorithmen können oft nicht mit Termen umgehen, die sowohl Symbole enthalten, die nur in A vorkommen, als auch solche, die nur in B vorkommen. Wie oben erwähnt, erzeugen viele Entscheidungsprozeduren für die Arraytheorie neue Terme. Um

also solche problematischen Terme zu vermeiden, werden für existierende Interpolationsmethoden für die Arraytheorie die Entscheidungsprozeduren an die Partitionierung A, B angepasst. Manche Programmverifikationsalgorithmen benötigen aber Interpolanten für mehrere Partitionierungen A, B einer Formel, was für diese Methoden bedeutet, dass für jede Partitionierung ein neuer Beweis erzeugt werden muss.

In der Dissertation wird ein Interpolationsalgorithmus für die Arraytheorie entwickelt, der, basierend auf der *Proof-Tree-Preserving-Interpolation*-Methode [CHN13], aus unabhängig von der Partitionierung A, B erzeugten Beweisen quantorenfreie Interpolanten berechnen kann. Dazu werden Algorithmen beschrieben, die für die Arraylemmas aus der Weak-Equivalence-basierten Entscheidungsprozedur Interpolanten berechnen. Sie nutzen die spezielle Form der Lemmas, um Schreibketten, die komplett in A oder komplett in B liegen, zusammenfassen. Die Schwierigkeit besteht darin, die relevanten Eigenschaften in Termen auszudrücken, die in der Interpolante erlaubt sind. Dieser Teil der Dissertation basiert auf den Publikationen [HS18] und [HS19], und wird in Abschnitt 3 beschrieben.

Schließlich werden für die Modellierung vieler Probleme in der Programmverifikation quantifizierte Formeln benötigt. Zum Beispiel lässt sich nur so die Korrektheit eines Sortieralgorithmus beschreiben. Auch Schleifeninvarianten müssen häufig eine Aussage für beliebig viele Schleifendurchläufe treffen und benötigen dafür quantifizierte Formeln. Zudem lassen sich auch Theorien, die der SMT-Solver nicht unterstützt, behandeln, indem die quantifizierten Axiome explizit der Eingabeformel hinzugefügt werden. SMT-Solver behandeln quantifizierte Probleme typischerweise dadurch, dass quantifizierte Formeln geeignet instanziiert werden, sodass sie nur das quantorenfreie Problem lösen müssen. Da Logik erster Stufe im Allgemeinen unentscheidbar ist, wird zumeist versucht, die Unerfüllbarkeit einer Formel zu zeigen. Potentiell hilfreiche Instanzen der quantifizierten Formeln werden in vielen SMT-Solvern mithilfe von *E-Matching* [DNS05] erzeugt. Die Idee dahinter ist, dass Instanzen quantifizierter Formeln vor allem dann nützlich sind, wenn sie existierende Terme weiter einschränken. Durch E-Matching werden für bestimmte Terme mit freien Variablen (sogenannte *Pattern*) Substitutionen der Variablen gefunden, für die äquivalente Terme im quantorenfreien Teil der Formel existieren. Äquivalent bedeutet hierbei, dass der gefundene Term und der instanziierte Term im Abschluss unter Transitivität der Gleichheit und Kongruenz von Funktionen in der gleichen Äquivalenzklasse liegen. E-Matching-basierte Instanziierung kann dennoch, je nach Wahl der Pattern, zu viele nutzlose Instanzen erzeugen.

In der Dissertation wird eine Instanziierungsmethode vorgestellt, die sich des E-Matchings bedient, um Kandidaten für Instanzen zu finden. Jedoch werden diese zusätzlich im aktuellen partiellen Modell ausgewertet, was durch die gefundenen äquivalenten Terme ermöglicht wird. Dadurch lassen sich solche Instanzen finden, die tatsächlich im Konflikt zum Modell stehen, oder die zu Propagationen führen. Damit kann ein Quantorenmodul ähnlich wie ein Theorie-Solver in den DPLL($\mathcal{T}$)-Algorithmus integriert werden. Dieser Teil der Dissertation basiert auf der Publikation [HS21] und wird in Abschnitt 4 beschrieben.

2 Entscheidungsprozedur für Constant Arrays

Die Arraytheorie, eingeführt von McCarthy [Mc62], definiert Funktionen *rd* und *wr*, die Lese- und Schreibzugriffe auf Arrays darstellen. Arrays entsprechen dabei Funktionen von einer Index- in eine Elementmenge. Ein Schreibterm $wr(a, i, v)$ bezeichnet das Array, das dem Array a an allen Indizes außer an i gleicht, wo es stattdessen v enthält. Das Element in a an der Stelle i wird duch den Leseterm $rd(a, i)$ bezeichnet. Es gilt Arraykongruenz, das heißt, für $i = j$ sind $rd(a, i)$ und $rd(a, j)$ gleich. Die folgenden Axiome formalisieren, dass ein Schreibzugriff nur eine Stelle modifiziert, und dass das geschriebene Element aus dem modifizierten Array an der entsprechenden Stelle ausgelesen werden kann. Desweiteren sind zwei Arrays gleich, wenn die enthaltenen Elemente paarweise gleich sind.

$$\forall\, a, i, v.\ rd(wr(a, i, v), i) = v \tag{idx}$$

$$\forall\, a, i, j, v.\ i \neq j \rightarrow rd(wr(a, i, v), j) = rd(a, j) \tag{read-over-write}$$

$$\forall\, a, b.\ (\forall\, i.\ rd(a, i) = rd(b, i)) \rightarrow a = b \tag{ext}$$

In der Dissertation wird die Arraytheorie mit Constant Arrays [St01] betrachtet. Sie definiert eine Funktion *const*, die ein Array darstellt, das an jeder Stelle das gleiche Element enthält.

$$\forall\, i, v.\ rd(const(v), i) = v \tag{const}$$

In der vorgestellten Entscheidungsprozedur gilt zusätzlich die Beschränkung, dass es unendlich viele Indizes geben muss. Andernfalls kann nicht unabhängig von der Kardinalität der Indexmenge entschieden werden, ob eine Formel erfüllbar ist oder nicht.

Weak-Equivalence-basierte Entscheidungsprozedur. Die Frage, ob eine Formel in der Arraytheorie erfüllbar ist, lässt sich mithilfe von geeigneten Instanziierungen der Array-Axiome beantworten. Dabei werden jedoch, neben vielen Formeln, viele neue Terme der Form $rd(a, i)$ erzeugt, welche insbesondere für Craig-Interpolation (siehe Abschnitt 3) problematisch sind. Eine von Christ und Hoenicke [CH15] entwickelte Entscheidungsprozedur nutzt Ketten von Schreibvorgängen zwischen Arrays, um Gleichheiten zwischen Arrays oder deren Elementen herzuleiten, ohne dabei neue Terme zu erzeuen. Zwei Arrays, die durch eine Schreibkette verbunden sind, werden *weakly equivalent* genannt (Notation $a \approx b$). Solche Arrays können sich nur an den endlich vielen Indizes dieser Schreibkette unterscheiden. Insbesondere enthalten sie an einem Index i, der verschieden von all diesen Indizes ist, das gleiche Element; sie werden dann auch *weakly equivalent auf i* (Notation $a \approx_i b$) genannt. Diese Relationen können in einem speziellen Graphen, dem *Weak-Equivalence-Graphen*, in dem Knoten Kongruenzklassen von Arrays und Kanten Schreibschritten zwischen Arrays entsprechen, effizient dargestellt werden. Für die Entscheidungsprozedur wird angenommen, dass alle Terme, deren Gleichheit nicht durch Gleichheitsliterale (und Transitivität und Funktionskongruenz) aus der Formel folgt, ungleich sind. Mithilfe von nur zwei Regeln können dann im Weak-Equivalence-Graphen alle Konflikte gefunden werden, die aus den Axiomen (idx) bis (ext) resultieren. Eine dieser Regeln findet Konflikte, wenn im aktuellen

Modell zwei Leseterme $rd(a,i), rd(b,j)$ ungleich sind, obwohl die Arrays a, b weakly equivalent auf i sind und die Indizes i und j gleich sind:

$$\frac{a \approx_i b \qquad i \sim j \qquad rd(a,i) \nsim rd(b,j)}{\neg\, Cond(a \approx_i b) \vee i \neq j \vee rd(a,i) = rd(b,j)} \qquad \text{(read-over-weakeq)}$$

Hier ist $Cond(a \approx_i b)$ die Konjunktion der Literale der Schreibkette, die zeigt, dass $a \approx_i b$ gilt. Das erzeugte Arraylemma stellt sicher, dass entweder eine der Bedingungen für die Relation $a \approx_i b$ nicht gilt, oder i und j verschieden sind, oder die Leseterme gleich sind. Die Regeln müssen nur für die tatsächlich in der Eingabeformel vorkommenden Terme angewendet werden, es werden dabei keine neuen Terme erzeugt.[3] Aufgrund der Annahme, dass Terme, deren Gleichheit nicht aus der Formel folgt, ungleich sind, sind die erzeugten Arraylemmas nicht unbedingt Konfliktklauseln in dem Sinne, dass alle Literale im aussagenlogischen Modell falsch sind, können aber gegebenenfalls auch zur Propagation von Literalen oder Disjunktionen davon verwendet werden.

Erweiterung für Constant Arrays. In der Dissertation wurde die oben beschriebene Entscheidungsprozedur für Constant Arrays erweitert. Constant Arrays $const(v)$ sind vollständig durch das Element v definiert. Dadurch ergeben sich weitere Möglichkeiten, wann ein Theoriekonflikt auftreten kann. Ist ein Array a weakly equivalent zu einem Array $const(v)$, so ist es selbst fast konstant: Es enthält den Wert v an allen Indizes, die nicht auf der Schreibkette auftauchen. Folgende neue Regel findet Konflikte, die aus dieser Eigenschaft resultieren, mithilfe des Weak-Equivalence-Graphen.

$$\frac{a \approx_i const(v) \qquad rd(a,i) \nsim v}{\neg\, Cond(a \approx_i const(v)) \vee rd(a,i) = v} \qquad \text{(read-const-weakeq)}$$

Ist a sogar selbst ein Constant Array $const(w)$, so müssen die Elemente v und w (und damit die Arrays selbst) gleich sein, da das Array unendlich viele Indizes hat, und es somit einen Index geben muss, der nicht in der Schreibkette vorkommt und an dem die beiden Arrays das gleiche Element enthalten. Für Konflikte dieser Art wird folgende Regel eingeführt.

$$\frac{const(v) \approx const(w) \qquad v \nsim w}{\neg\, Cond(const(v) \approx const(w)) \vee v = w} \qquad \text{(const-weakeq)}$$

Die Korrektheit der erweiterten Entscheidungsprozedur wird in der Dissertation bewiesen, indem zum einen gezeigt wird, dass jede der Regeln einen Theoriekonflikt aufdeckt, und zum anderen, dass sich, wenn keine der Regeln anwendbar ist, ein Modell der Eingabeformel konstruieren lässt, für das die Axiome der Arraytheorie gelten. Zudem wurde die Entscheidungsprozedur in den SMT-Solver SMTInterpol [CHN12] implementiert. Eine experimentelle Auswertung zeigt, dass mithilfe der Constant Arrays Anfragen an den SMT-Solver signifikant vereinfacht werden können.

[3] In einem Preprocessing-Schritt müssen einige neue Leseterme erzeugt werden. Jeder dieser Terme enthält aber nur Subterme eines einzelnen in der Formel vorkommenden Terms, was *kein* Problem für Interpolation darstellt.

3 Interpolation für Arrays

Eine (Craig-)Interpolante [Cr57] für ein Paar (A, B) von Formeln, deren Konjunktion unerfüllbar ist, ist eine Formel I, die aus A folgt und in Widerspruch zu B steht, aber nur Symbole enthält, die sowohl in A als auch in B vorkommen (*gemeinsame* Symbole). Bei letzterer Bedingung sind Symbole, die von einer Theorie interpretiert werden, ausgenommen. Interpolanten lassen sich aus von SMT-Solvern erzeugten Resolutionsbeweisen berechnen, indem für jeden Knoten beginnend von den Blättern induktiv eine sogenannte *partielle Interpolante* berechnet wird. Die Blätter in einem solchen Beweis sind entweder Eingabeklauseln oder Theorielemmas. Für Theorielemmas werden theoriespezifische Verfahren benötigt. Die Schwierigkeit liegt darin, die für die Unerfüllbarkeit relevanten Folgerungen aus der Formel A in gemeinsamen Termen auszudrücken. In der Dissertation wird ein Verfahren vorgestellt, das für Lemmas der Arraytheorie, die mit der in Abschnitt 2 beschriebenen Entscheidungsprozedur erzeugt wurden, partielle Interpolanten berechnet.

Quantorenfreie Interpolation für die Arraytheorie. In der in Abschnitt 2 beschriebenen Arraytheorie gibt es quantorenfreie Formeln, die keine quantorenfreie Interpolante besitzen. Eine von Bruttomesso et al. eingeführte Variante [BGR12], die dieses Problem löst, definiert eine Funktion *diff*, die für zwei sich unterscheidende Arrays einen Index zurückgibt, an dem sie sich unterscheiden, und sonst einen beliebigen Index. Dies wird durch eine skolemisierte Variante des Extensionalitätsaxioms (ext) formalisiert.

$$\forall\, a, b.\ rd(a, diff(a, b)) = rd(b, diff(a, b)) \rightarrow a = b \qquad \text{(ext-diff)}$$

Mithilfe der *diff*-Funktion lassen sich Indexterme konstruieren, die in der Interpolante erlaubt sind, selbst wenn die Indexterme aus der Eingabeformel nicht in der Interpolante vorkommen dürfen. Die Weak-Equivalence-basierte Entscheidungsprozedur lässt sich leicht für diese Variante erweitern [CH15].

Eine wichtige Voraussetzung für existierende Interpolationsmethoden ist, dass keine Terme im Beweis vorkommen, die sowohl Symbole enthalten, die nur in A vorkommen, als auch Symbole, die nur in B vorkommen. Wie in Abschnitt 2 angedeutet, werden solche Terme in der Weak-Equivalence-basierten Entscheidungsprozedur unabhängig von der Partitionierung in A und B gar nicht erzeugt. Sie kann jedoch Gleichheiten erzeugen, bei denen die eine Seite nur in A und die andere nur in B vorkommt. Solche Literale sind aber in der Proof-Tree-Preserving-Interpolation-Methode [CHN13] erlaubt. Um in diese Methode integriert zu werden, müssen die partiellen Interpolanten der Arraylemmas von einer bestimmten Form sein, die sich aber natürlicherweise ergibt.

Interpolation für Weak-Equivalence-basierte Lemmas der Arraytheorie. In der Dissertation wird beschrieben, wie partielle Interpolanten für die vier Typen von Arraytheorie-Lemmas für verschiedene Partitionierungen A, B berechnet werden können. Im Folgenden werden einige der Ideen dafür erläutert.

Die Grundidee ist, dass sich die jedem Lemma zugrunde liegenden Schreibketten zwischen Arrays a und b so aufteilen lassen, dass Teilstücke komplett in A oder komplett in B liegen (also deren Symbole alle in A bzw. alle in B vorkommen), und diese Teilstücke dann in Formeln zusammenfasst werden, die nur *gemeinsame* Symbole verwenden. Diese Formeln können zum Beispiel ausdrücken, dass die Arrays an den Enden eines in A liegenden Teilstücks an einer bestimmten Stelle das gleiche Element enthalten. Die Schwierigkeit besteht darin, dass nicht immer gemeinsame Terme in der Eingabeformel vorhanden sind, die in der Interpolante verwendet werden dürfen. Enthält beispielsweise ein Lemma aus Regel (read-over-weakeq) eine Schreibkette, deren Indizes nur in A vorkommen, während die Indizes i und j nur in B vorkommen, dann existiert kein gemeinsamer Index.

Mithilfe der *diff*-Funktion und Schreibketten zwischen gemeinsamen Arraytermen lassen sich Indexterme als neu konstruierte gemeinsame Terme repräsentieren. Gilt zum Beispiel die Arraygleichheit $s = wr(s', k, v)$, so gilt auch, dass $diff(s, s')$ der Index k ist oder die Arrays s und s' gleich sind, und in beiden Fällen lässt sich s in s' umschreiben, indem an den Index $diff(s, s')$ das Element aus s' an dieser Stelle geschrieben wird, also $wr(s, diff(s, s'), rd(s', diff(s, s'))) = s'$. Für eine Schreibkette der Länge m wird dies durch eine Formel $\mathrm{weq}(s, s', m, F[x])$ verallgemeinert. Dabei bezeichnet $F[x]$ eine Formel, die für jeden beliebigen *diff*-Term in dieser Umschreibkette gelten soll, sofern das aus s umgeschriebene Array noch nicht gleich s' ist. Im oben angeführten Beispiel kann so das Teilstück in A zusammengefasst werden, indem für F eine Formel gewählt wird, die sicherstellt, dass die *diff*-Terme verschieden von i sein müssen.

Aus einer Schreibkette der Länge m zwischen einem Constant Array $const(v)$ und einem beliebigen anderen Array s lässt sich mithilfe der *diff*-Funktion auch eine Repräsentation des Wertes v in gemeinsamen Symbolen (und damit des Constant Arrays) konstruieren. Da das Array s selbst fast überall v enthält, ist das Element in s an der Stelle $diff(s, s)$ schon ein guter Kandidat. Sind $rd(s, diff(s, s))$ und v verschieden, so kann $diff(s, s)$ nur einer der m Schreibindizes sein. Das Constant Array mit Wert $v' = rd(s, diff(s, s))$ unterscheidet sich an unendlich vielen Stellen von s, nämlich allen außer $diff(s, s)$ und möglicherweise den Indizes auf der Schreibkette. Schreibt man $const(v')$ nach und nach mithilfe der *diff*-Funktion zu s um, so muss die *diff*-Funktion nach spätestens $m - 1$ Umschreibschritten einen Index zurückliefern, an dem in s der Wert v steht, da nach jedem Schritt, in dem keine Stelle gefunden wird, an der s den Wert v enthält, einer der übrigen $m - 1$ Schreibindizes zurückgegeben wird und somit das nächste umgeschriebene Array an dieser Stelle mit s übereinstimmt.

In der Dissertation wird die Korrektheit der Interpolationsmethode bewiesen. Desweiteren wird die Komplexität der Interpolanten in der Größe des Lemmas untersucht. Die Interpolationsmethode wurde in SMTInterpol implementiert und evaluiert.

4 Instanziierungsmethode für Quantifizierte Formeln

Ein verbreiteter Ansatz in SMT-Solvern, mit quantifizierten Formeln umzugehen, ist es, zunächst ein Modell für den quantorenfreien Teil der Formel zu suchen, und dann Instanzen der quantifizierten Formeln hinzuzufügen, mit dem Ziel dieses Modell zu widerlegen. Für den Erfolg dieser Methode ist entscheidend, welche Instanzen dafür gewählt werden. Instanzen, die sofort zu einem Konflikt führen, sind nützlich, da sie zeigen, dass die aktuelle Literalbelegung ungeeignet ist. Ebenso können Instanzen, die bereits existierende Terme enthalten, nützlich sein, da sie diese weiter beschränken. Dies ist die Idee hinter E-Matching-basierter Instanziierung [DNS05]. Die Dissertation stellt eine Instanziierungsmethode vor, die E-Matching verwendet, aber damit gezielt Instanzen findet, die zu einem Konflikt oder zu Propagationen führen. Im Folgenden wird die Theorie der Gleichheit $\mathcal{T}_{\mathcal{EUF}}$ betrachtet, in der außer dem Gleichheitssymbol nur *uninterpretierte* Funktionssymbole erlaubt sind, die durch das Kongruenzaxiom beschränkt sind.

E-Matching. E-Matching löst folgendes Problem: Gegeben ein Term $p[x_1, \ldots, x_n]$ (ein *Pattern*) mit freien Variablen $\{x_1, \ldots, x_n\}$, finde einen Term t im E-Graphen und eine Substitution $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, sodass t und $p\sigma$ kongruent zueinander sind. Der E-Graph ist dabei der Graph, der durch Kongruenzabschluss aus den Gleichheiten des aussagenlogischen Modells konstruiert wird. Ein Term passt in ein Pattern, wenn er in einer Kongruenzklasse mit einem Term liegt, der das gleiche Funktionssymbol wie das Pattern hat, und dessen Argumente zu den Argumenten des Patterns passen; für Variablen passen beliebige Terme. Das E-Matching-Problem kann auch für eine Liste von Pattern erweitert werden. Es wird dann eine gemeinsame Substitution gesucht, unter der es jeweils einen passenden Term für jedes Pattern gibt.

Bei E-Matching-basierter Instanziierung wird für jede quantifizierte Formel ein Pattern gewählt, das alle Variablen der Formel enthält. Für jede Lösung wird die entsprechende Instanz zum quantorenfreien Teil der Formel hinzugefügt. Oft werden so allerdings auch irrelevante Instanzen erzeugt. Für die Formel

$$\forall x, y, z.\ f(x, y) \neq c \vee f(y, z) \neq c \vee f(x, z) = c$$

eignet sich bespielsweise das Pattern $f(x, y), f(y, z)$. Angenommen, der quantorenfreie Teil des Problems ist folgende Konjunktion:

$$f(a, b) = a \wedge f(b, b) = b \wedge f(b, c) = c \wedge a = c \wedge b \neq c$$

Eine mögliche Lösung des E-Matching-Problems sind die Terme $f(b, c), f(a, b)$ für die Substitution $\{x \mapsto b, y \mapsto c, z \mapsto b\}$, da $f(y, z)$ unter dieser Substitution kongruent zu $f(a, b)$ ist. Mit der entsprechenden Instanz $f(b, c) \neq c \vee f(c, b) \neq c \vee f(b, b) = c$ lässt sich schnell ein Widerspruch zum quantorenfreien Teil zeigen. Es gibt jedoch vier weitere Lösungen. Die Lösung $f(a, b), f(b, c)$ und $\{x \mapsto a, y \mapsto b, z \mapsto c\}$ erzeugt beispielsweise eine Instanz, die den neuen Term $f(a, c)$ enthält, der für das quantorenfreie Problem keinerlei Relevanz hat.

Suche nach Konflikt- und Propagationsinstanzen. Die zentrale Idee der in der Dissertation vorgestellten Instanziierungsmethode ist, dass mitilfe von E-Matching nicht nur Kandidaten für Instanzen gefunden, sondern diese auch direkt im aktuellen partiellen Modell ausgewertet werden können. Die Methode arbeitet mit allquantifizierten Klauseln, die der Einfachheit halber als Disjunktion von Literalen mit freien Variablen angesehen werden werden:

$$C : \ell_1[x_1, \ldots, x_n] \vee \ldots \vee \ell_m[x_1, \ldots, x_n]$$

Ziel ist es, eine Substitution $t_1, \ldots, t_n$ für die Variablen $x_1, \ldots, x_n$ zu finden, sodass die aktuelle Literalbelegung impliziert, dass keines der instanziierten Literale wahr sein kann. Die entsprechende Instanz wird *Konfliktinstanz* genannt. Dies ist nicht unbedingt ein Konflikt auf aussagenlogischer Ebene in dem Sinne, dass allen Literalen bereits der Wert falsch zugewiesen wurde. Der Theorie-Solver für $\mathcal{T}_{\mathcal{EUF}}$ kann aber mithilfe der Instanz schnell den Konflikt mit dem aussagenlogischen Modell herleiten. Zusätzlich sollen auch solche Substitutionen gefunden werden, für die alle bis auf eines der instanziierten Literale im partiellen Modell falsch sind. Die entsprechende Instanz ist dann eine *Propagationsinstanz*.

Für eine Gleichheit $p_1 = p_2$ zwischen zwei Termen, oder deren Negation, wird das E-Matching-Problem für das Pattern p_1, p_2 gelöst. Mithilfe der gefundenen Terme t_1 und t_2 lässt sich dann der Wahrheitswert des Literals im aktuellen Modell bestimmen: Sind die Terme t_1 und t_2 in derselben Kongruenzklasse, so ist die instanziierte Gleichheit wahr und die Ungleichheit falsch, und umgekehrt, wenn es eine Ungleichheit zwischen den Termen gibt. Andernfalls wird der Wert als unit festgelegt, da die Ungleichheit keine Konsequenz der aktuellen Literalbelegung ist, sondern nur im partiellen Modell der Gleichheitstheorie gilt. Im oben für E-Matching angeführten Beispiel kann schon, nachdem den Literalen $f(a, b) = a, f(b, b) = b, f(b, c) = c, a = c$ der Wert wahr zugewiesen wurde, zum Beispiel die Instanziierung für $\{x \mapsto a, y \mapsto b, z \mapsto b\}$ verworfen werden, da das Literal $f(x, z) = c$ mit äquivalentem Term $f(a, b)$ zu wahr auswertet. Dagegen führt die Instanziierung für $\{x \mapsto b, y \mapsto c, z \mapsto b\}$ zu einer Propagation, da die ersten beiden Literale zu falsch auswerten und das dritte zu unit. Ist aber auch dem Literal $b \neq c$ der Wert wahr zugewiesen, so ist diese Instanz eine Konfliktinstanz.

In der Dissertation wird beschrieben, inwieweit die Suche nach Konflikt- und Propagationsinstanzen inkrementell erfolgen kann. Es wird gezeigt, dass sie unter gewissen Annahmen theoretisch in der Lage ist, jede Konfliktinstanz für ein gegebenes partielles Modell der quantorenfreien Formel zu finden. Außerdem werden Erweiterungen der Methode für Formeln mit Arithmetik diskutiert. Die Instanziierungsmethode wurde in SMTInterpol implementiert. Eine experimentelle Evaluation zeigt, dass die Methode im Vergleich zu einer konventionellen E-Matching-basierten Instanziierungsmethode mehr Probleme lösen kann und dafür wesentlich weniger Instanzen benötigt. Eine offene Frage ist, inwieweit komplementäre Instanziierungsmethoden, die für Vollständigkeit in entscheidbaren Fragmenten benötigt werden, die vorgestellte Methode beeinflussen.

Literatur

[BGR12] Bruttomesso, R.; Ghilardi, S.; Ranise, S.: Quantifier-Free Interpolation of a Theory of Arrays. Log. Methods Comput. Sci. 8/2, 2012.

[CH15] Christ, J.; Hoenicke, J.: Weakly Equivalent Arrays. In: FroCos. Bd. 9322. Lecture Notes in Computer Science, Springer, S. 119–134, 2015.

[CHN12] Christ, J.; Hoenicke, J.; Nutz, A.: SMTInterpol: An Interpolating SMT Solver. In: SPIN. Bd. 7385. Lecture Notes in Computer Science, Springer, S. 248–254, 2012.

[CHN13] Christ, J.; Hoenicke, J.; Nutz, A.: Proof Tree Preserving Interpolation. In: TACAS. Bd. 7795. Lecture Notes in Computer Science, Springer, S. 124–138, 2013.

[Cr57] Craig, W.: Three Uses of the Herbrand-Gentzen Theorem in Relating Model Theory and Proof Theory. J. Symb. Log. 22/3, S. 269–285, 1957.

[DNS05] Detlefs, D.; Nelson, G.; Saxe, J. B.: Simplify: a theorem prover for program checking. J. ACM 52/3, S. 365–473, 2005.

[Ga04] Ganzinger, H.; Hagen, G.; Nieuwenhuis, R.; Oliveras, A.; Tinelli, C.: DPLL(T): Fast Decision Procedures. In: CAV. Bd. 3114. Lecture Notes in Computer Science, Springer, S. 175–188, 2004.

[HS18] Hoenicke, J.; Schindler, T.: Efficient Interpolation for the Theory of Arrays. In: IJCAR. Bd. 10900. Lecture Notes in Computer Science, Springer, S. 549–565, 2018.

[HS19] Hoenicke, J.; Schindler, T.: Solving and Interpolating Constant Arrays Based on Weak Equivalences. In: VMCAI. Bd. 11388. Lecture Notes in Computer Science, Springer, S. 297–317, 2019.

[HS21] Hoenicke, J.; Schindler, T.: Incremental Search for Conflict and Unit Instances of Quantified Formulas with E-Matching. In: VMCAI. Bd. 12597. Lecture Notes in Computer Science, Springer, S. 534–555, 2021.

[Mc03] McMillan, K. L.: Interpolation and SAT-Based Model Checking. In: CAV. Bd. 2725. Lecture Notes in Computer Science, Springer, S. 1–13, 2003.

[Mc62] McCarthy, J.: Towards a Mathematical Science of Computation. In: IFIP Congress. North-Holland, S. 21–28, 1962.

[Sc22] Schindler, T.: SMT solving, interpolation, and quantifiers, Diss., University of Freiburg, Freiburg im Breisgau, Germany, 2022.

[St01] Stump, A.; Barrett, C. W.; Dill, D. L.; Levitt, J. R.: A Decision Procedure for an Extensional Theory of Arrays. In: LICS. IEEE Computer Society, S. 29–37, 2001.

Tanja Schindler studierte bis 2015 Mathematik an der Universität Freiburg und der Universität Göteborg (Erasmus-Jahr). Danach wechselte sie zur Informatik und forschte von 2016 bis 2022 als Doktorandin an der Professur für Softwaretechnik am Institut für Informatik in Freiburg. Während der Promotion verbrachte sie einen dreimonatigen Forschungsaufenthalt bei SRI International in New York. Sie promovierte 2022 zum Thema SMT-Solving. Zur Zeit arbeitet sie als Postdoc am Institut Montefiore an der Université de Liège.