Automatisierte Garantierte Blockchain-Technologie Verifizierung[1]

Rodrigo Otoni [2]

Ausgezeichnete Informatikdissertationen 2023 – Gesellschaft für Informatik
DOI: https://doi.org/10.18420/Diss2023-19

Abstract: Blockchain-Technologien haben sowohl in der Wissenschaft als auch in der Industrie große Aufmerksamkeit auf sich gezogen. Da sie als Mittel zum Halten und Manipulieren von Finanzwerten verwendet werden, sind sie ein beliebtes Ziel von Angriffen. In der Vergangenheit sind bereits Vermögenswerte in der Größenordnung von Millionen von US-Dollar verloren gegangen. Vor diesem Hintergrund ist die Fähigkeit, sicherzustellen, dass keine Schwachstellen vorhanden sind, von entscheidender Bedeutung. Die hier zusammengefasste Dissertation stützt sich auf die jüngsten Fortschritte im Bereich der symbolischen Modellprüfung, insbesondere auf Techniken, die auf Satisfiability Modulo Theories (SMT) und Constrained Horn Clauses (CHC) basieren, um den Bedarf an einer automatisierten Überprüfung von Blockchain-Technologien zu decken. Der Blockchain-Bereich wird sowohl auf der Plattform- als auch auf der Anwendungsebene durch TLA+-Spezifikationen und Solidity-Smart-Contracts angegangen, und die Verwendung von Korrektheitszeugnissen zur Zertifizierung der Ergebnisse von Solvern, die für die betreffenden Logikfragmente bestimmt sind, wird als Mittel zur Gewährleistung der Korrektheit untersucht.

1 Einführung

Verteilte Ledger, die den Blockchain-Technologien zugrunde liegen, ermöglichen sichere Transaktionen zwischen misstrauischen Parteien, ohne dass eine Aufsichtsbehörde, wie z. B. eine Bank, eingeschaltet werden muss. Ihr Aufstieg verspricht, die etablierte Art und Weise, in der Einzelpersonen und Institutionen miteinander interagieren, zu verändern und einen billigeren und sichereren Weg zum Austausch von Waren und Dienstleistungen zu bieten. Im Kern kann man sich Blockchain-Plattformen als eine große Anzahl geografisch verteilter, replizierter Zustandsmaschinen vorstellen. Jede Replik enthält eine Kopie eines Ledgers, wobei ein verteilter Algorithmus, der oft als Protokoll bezeichnet wird, dafür sorgt, dass die Aktualisierung des Hauptbuchs auf konsistente Weise erfolgt. Solche Algorithmen sind oft äußerst komplex, da sie den Konsens in der Gegenwart von beliebigen (sogenannten Byzantinischen) Fehlern sicherstellen müssen, wobei verschiedene Blockchain-Plattformen oft ihr eigenes spezielles Protokoll haben. Die Verwendung von verteilten Ledgern konzentrierte sich zunächst auf die Schaffung austauschbarer Token in Form von Kryptowährungen, z. B. Bitcoin. Als das Interesse an diesem Bereich wuchs, begannen Blockchain-Plattformen auch Programme zu unterstützen, die in der Lage sind, als Code geschriebene vertragliche Vereinbarungen automatisch auszuführen, und zwar in Form von *Smart Contracts*. Technisch gesehen handelt es sich bei Smart Contracts um verteilte Programme, die auf Blockchain-Plattformen eingesetzt werden, was zu einer Reihe von

1 Engl. Titel der Dissertation: "Automated Verification of Blockchain Technologies with Correctness Guarantees"

2 Università della Svizzera italiana, Lugano, Schweiz, otonir@usi.ch, https://orcid.org/0000-0003-1097-2367

Besonderheiten führt, insbesondere dass sich die Argumentation über diese Programme von der Argumentation über allgemeine Programme deutlich unterscheidet. Ein gutes Beispiel für solche Unterschiede ist die transaktionale Natur von Smart Contracts, die sicherstellt, dass Funktionsausführungen entweder erfolgreich beendet oder alle Änderungen rückgängig gemacht werden, da die Auswirkungen dieser Ausführungen in der Blockchain gespeichert werden. Diese Neuerungen bilden die Grundlage für die Entstehung eines völlig neuen Ökosystems, in dem Entwickler Anwendungen über Smart Contracts erstellen und sich auf Blockchains verlassen, um Token wie vorgesehen sicher zu übertragen.

Aufgrund des ihnen innewohnenden finanziellen Aspekts sind Blockchain-Technologien ein beliebtes Ziel von Angriffen, die versuchen können, Schwachstellen auszunutzen entweder in Smart Contracts-Anwendungen oder in den Blockchain-Plattformen, auf die sie sich stützen. Die Fähigkeit, sicherzustellen, dass sowohl in Smart-Contract-Anwendungen als auch in Blockchain-Plattformen keine Schwachstellen vorhanden sind, ist daher von größter Bedeutung, wobei die formale Verifizierung ein idealer Ansatz zur Erreichung dieses Ziels ist. Die formale Überprüfung ist ein weit gefasster Begriff, der sich auf eine Reihe von logikbasierten Techniken bezieht, die darauf abzielen, mathematisch sicherzustellen, dass sich ein Programm oder System entsprechend einer Spezifikation verhält. Die verschiedenen formalen Verifikationstechniken unterscheiden sich in ihrem Automatisierungsgrad, ihrem Anwendungsbereich und ihrer Effizienz und eignen sich oft am besten für bestimmte Situationen. *Modellprüfung* ist eine Technik, die im Laufe der Jahre viele Erfolge verzeichnen konnte, die vor allem durch den Turing Award 2007 anerkannt wurden. Im Wesentlichen besteht die Modellprüfung darin, ein Programm oder System zusammen mit einer gewünschten Eigenschaft in einem geeigneten mathematischen Formalismus auszudrücken und dann zu prüfen, ob alle Verhaltensweisen die gegebene Eigenschaft einhalten. Trotz aller Erfolge bleibt die Modellprüfung in der Praxis aufgrund der begrenzten Skalierbarkeit, die durch das bekannte Problem der Zustandsexplosion hervorgerufen wird, eine schwierige Aufgabe. Diese Aufgabe, die in Fällen, die ausschließlich aus sequentiellem Verhalten bestehen, bereits äußerst schwierig ist, wird in einem verteilten Umfeld, in dem die Kommunikation zwischen verschiedenen Knoten umfangreich ist und über potenziell unzuverlässige Kanäle erfolgt, noch viel schwieriger. In dem Bestreben, dieses Problem besser in den Griff zu bekommen, wurden Modellprüfungsansätze vorgeschlagen, die auf symbolischen Schlussfolgerungen mittels logischer Formeln basieren. Symbolische Schlussfolgerungen werden in der Regel in Fragmenten von first-order logic (FOL) gemacht, da diese in der Lage sind, viele interessante Verifikationsprobleme zu repräsentieren, wobei die Wahl des Fragments auf einem Kompromiss zwischen Aussagekraft und Effizienz beruht. Automatisierte Argumentation von FOL Formeln erfolgt über Logiklöser, wobei jeder dieser Löser für eine oder mehrere FOL Fragmente zuständig ist. Die bekanntesten Löser-Kategorien sind die der Boolean satisfiability (SAT) und satisfiability modulo theories (SMT), die sich jeweils auf Formeln im Satzfragment von FOL und in Erweiterungen davon mit Theorien wie Arithmetik, Arrays und Bit-Vektoren konzentrieren. Eine weitere Kategorie von Interesse ist die der Löser, die auf constrained Horn clauses (CHC) arbeiten, ein FOL Fragment, das der Hoare-Logik entspricht und in der Praxis vielfach verwendet wurde.

Wir wollen, dass die formale Verifikation uns in die Lage versetzt, unseren Programmen und Systemen zu vertrauen, ob sie nun im Blockchain-Bereich angesiedelt sind oder nicht, aber dazu müssen wir fähig sein, den Verifikationstechniken und -werkzeugen selbst zu vertrauen. Trotz ihres umfangreichen Einsatzes in der Verifikation, wobei ein ausdrückliches Ziel darin besteht, Fehler zu verhindern, die von Angreifern ausgenutzt werden können, sind Logiklöser selber keineswegs immun gegen Fehler. Dies wurde durch die wiederholte Aufdeckung von Problemen bei Tool-Wettbewerben in den letzten Jahren deutlich, wobei viele moderne Löser widersprüchliche Ergebnisse lieferten. Vor diesem Hintergrund sind Garantien für die Ergebnisse von Lösern von größter Bedeutung. Ein Ansatz zur Erreichung dieses Ziels ist die formale Verifizierung des Löser-Codes. Trotz der starken Garantien, die dieser Ansatz bietet, verursacht er hohe Kosten für die Verifizierung des bestehenden Codes und aller zukünftigen Änderungen daran und verhindert möglicherweise viele Optimierungen, die für die Leistung der Löser wesentlich sind. Ein anderer, weniger invasiver Ansatz besteht darin, die Ergebnisse der Löser zu zertifizieren, anstatt die Löser selbst zu verifizieren. Dies setzt voraus, dass ein Löser zusätzlich zu seiner Standardausgabe auch ein Zeugnis erbringt, der von einem unabhängigen Werkzeug zur Validierung des gegebenen Ergebnisses verwendet werden kann. Derzeit bewegt sich die Gemeinschaft in Richtung des zweiten Ansatzes, gefolgt von SAT und SMT Wettbewerben, wobei viele Zeugnisformate vorgeschlagen werden, um die Ergebnisse von SAT und SMT Lösern zu prüfen.

Zusammenfassend lässt sich sagen, dass die Notwendigkeit, Schwachstellen zu verhindern, die von Angreifern ausgenutzt werden können, in den letzten Jahren zu erheblichen Forschungsanstrengungen geführt hat, dass aber noch viele Herausforderungen bestehen. Die hier zusammengefassten Beiträge der Dissertation [Ot23d], die im Folgenden aufgeführt sind, befassen sich mit vier dieser Herausforderungen und können in die Kategorien automatische Überprüfung und Korrektheitsgarantien unterteilt werden (vgl. Abb. 1).

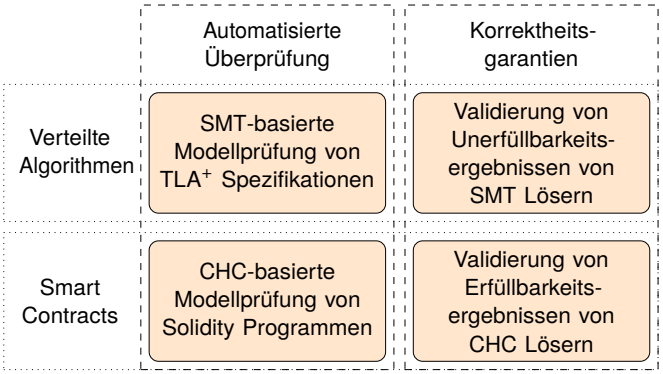

Abb. 1: Beiträge der Dissertation.

- Ein skalierbarer SMT-basierter Ansatz zur Modellprüfung von TLA$^+$ Spezifikationen wurde entwickelt, um die Effizienz der Überprüfung verteilter Systeme zu erhöhen, insbesondere solcher, die Byzantinisch fehlertolerant sind (vgl. Abs. 2).
- Es wurde eine groß angelegte Evaluierung eines CHC-basierten Modellprüfungsansatzes für Solidity-Programme durchgeführt und eine Erweiterung dieses Ansatzes

vorgeschlagen, die bei der präzisen Darstellung der Besonderheiten von Smart Contracts während der Überprüfung hilft (vgl. Abs. 3).

- Es wurde ein neuartiges Format für SMT-Zeugnisse der Unerfüllbarkeit entwickelt, bei dem die Kompaktheit im Vordergrund steht, um das Problem der Zeugnisgrößen zu lösen (vgl. Abs. 4).
- Ein neuartiger beweisgestützter Ansatz für die Validierung von CHC-Erfüllbarkeitszeugnissen wurde entwickelt, um die Zertifizierungsgarantien zu stärken (vgl. Abs. 5).

2 Modellprüfung von TLA$^+$ Spezifikationen

Die Fähigkeit sicherzustellen, dass ein verteiltes System, wie z. B. eine Blockchain, korrekt funktioniert, ist von größter Bedeutung. Ein bekannter Ansatz zur Erreichung dieses Ziels besteht darin, Protokolle mit TLA$^+$ zu spezifizieren, eine auf dem temporal logic of actions (TLA) basierende Spezifikationssprache, die es dem Benutzer ermöglicht, das erwartete Verhalten eines Systems zu beschreiben und gleichzeitig von Implementierungsdetails zu abstrahieren, die keine Auswirkungen auf die Eigenschaften auf hoher Ebene haben. TLA$^+$ ist sowohl im akademischen Bereich als auch in der Industrie weit verbreitet, mit einer Handvoll von Werkzeugen, die zur Unterstützung von Verifikationsaufgaben zur Verfügung stehen. Trotz des Interesses und der jüngsten Fortschritte bleibt die Verifikation verteilter Systeme notorisch schwierig. Dies liegt daran, dass die Ausführung verteilter Algorithmen naturgemäß zahlreiche potenzielle Verschachtelungen von Schritten zulässt, wobei die Zustandsräume im Allgemeinen exponentiell mit der Anzahl der Teilnehmer wachsen.

Symbolische Modellprüfung von TLA$^+$ Spezifikationen wurde angeführt von Konnov et al. [KKT19], die eine Kodierung von TLA$^+$ in SMT-Bedingungen über uninterpretierte Konstanten und Arithmetik vorgeschlaben haben. Ihr Werkzeug, APALACHE, war sehr erfolgreich, stieß aber bei der Bearbeitung besonders komplexer Spezifikationen auf Schwierigkeiten. Ein Schlüsselaspekt des Ansatzes von APALACHEist die Kodierung von TLA$^+$-Datenstrukturen als uninterpretierte Konstanten, die verhindert, dass die in der Eingabespezifikation vorhandene Strukturinformation in der kodierenden SMT-Formel reflektiert wird, was sich negativ auf die Leistung auswirkt. Die strukturellen Informationen, die nicht an den SMT-Löser weitergeleitet werden, haben das Potenzial, die Lösungseffizienz erheblich zu verbessern. Vor diesem Hintergrund haben wir eine alternative SMT-Kodierung vorgeschlagen, die die SMT-Theorie der Arrays voll ausnutzt um die wichtigsten TLA$^+$-Datenstrukturen zu kodieren. Unsere neuartige Kodierung besteht aus einem neuen abstrakten Reduktionsssystem, um Formeln zu generieren, die dem Löser nicht nur mehr Informationen liefern, sondern auch wesentlich kompakter sind als zuvor. Einige Operationen über TLA$^+$-Datenstrukturen, die zuvor eine Menge von Beschränkungen generierten, die quadratisch in der Größe der Datenstrukturen waren, führen stattdessen zu einer konstanten Anzahl von Beschränkungen (vgl. Tab. 1). Die vorgeschlagene Kodierung wurde in APALACHE implementiert und mit der ursprünglichen Kodierung von APALACHE

und dem expliziten Zustandsaufzählungsansatz des Modellprüfers TLC verglichen, wobei drei in TLA$^+$ spezifizierte asynchronen Protokollen verwendet wurden. Die Auswertung zeigt, dass die Einbettung struktureller Informationen in die SMT-Formeln einen signifikanten positiven Einfluss auf die Leistung hat, mit einer Verbesserung der Modellprüfungszeit um bis zu einer Größenordnung. Die Ergebnisse dieser Arbeit wurden in TACAS'23 [Ot23c] veröffentlicht.

Tab. 1: Anzahl der von jeder SMT-Kodierung erzeugten Beschränkungen zur Modellierung der wichtigsten TLA$^+$-Konstrukte.

Konstrukt	Konstanten	Arrays
Menge Aufzählung	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Menge Filter	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Menge Map	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Menge Mitgliedschaft	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Menge Gleichheit	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$
Funkt. Definition	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Funkt. Domäne	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Funkt. Gleichheit	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$
Funkt. Update	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Funkt. Anwendung	$\mathcal{O}(n)$	$\mathcal{O}(n)$

3 Modellüberprüfung von Solidity-Programmen

Durch ihren Einsatz auf Blockchain-Plattformen weisen Smart Contracts eine Reihe von Besonderheiten auf, die bei allgemeinen Programmen nicht zu finden sind. In Bezug auf die Ausführung ist ihr transaktionaler Charakter, bei dem die Ausführung einer Funktion entweder erfolgreich abschliesst oder alle Änderungen rückgängig gemacht werden, der Hauptunterschied zu herkömmlichen Programmen. Was den Code selbst betrifft, so ist er (1) nach der Bereitstellung unveränderlich, was die Behebung von Schwachstellen verhindert, (2) öffentlich sichtbar, so dass potenzielle Angreifer nach Code-Exploits suchen können, und (3) frei verfügbar, was bedeutet, dass jeder Benutzer mit seiner Schnittstelle interagieren kann. Die Kombination dieser Merkmale macht die Überprüfung von Smart Contracts unerlässlich. Die wichtigste Programmiersprache, die speziell für die Implementierung von Smart Contracts entwickelt wurde, ist derzeit Solidity, und auf diese Sprache haben wir uns konzentriert. Unsere anfängliche Untersuchung ergab, dass bestehende Arbeiten verschiedene Zwischenrepräsentationen verwenden, um verfügbare Verifizierungswerkzeuge wiederzuverwenden, was uns dazu veranlasste, eine direkte Kodierung von Solidity in CHC zu entwickeln, um die Notwendigkeit fehleranfälliger und potenziell kostspieliger Übersetzungen zu vermeiden [Ma20]. Unser Ansatz basiert auf einer direkten Modellierung, d.h. die Verifikationsbedingungen werden im Zielformalismus direkt aus dem Vertrag generiert, wobei domänenspezifisches Wissen verwendet wird. Die Kodierung ermöglicht die Überprüfung von Assertionen im Quellcode, die als Vertragsspezifikationen betrachtet werden. Durch die Überprüfung von Assertionen können Entwickler nicht nur sicherstellen, dass keine allgemein bekannten Schwachstellen vorhanden sind, indem sie Assertionen einfügen, die das Verhalten von Angreifern blockieren, sondern auch die funktionale Korrektheit überprüfen. Mit Hilfe von CHC muss, sobald die Menge der erreichbaren Vertragszustände bestimmt ist, entweder eine Vertragsinvariante festgestellt werden, die beweist, dass eine bestimmte Eigenschaft auch nach einer unbegrenzten Anzahl von

Transaktionen gilt, oder ein counterexample (CEX) gefunden werden, der eine Verletzung der Eigenschaft zeigt. Vertragsinvarianten sind Bedingungen über die Variablen des Vertrags, die immer gelten und somit von den Entwicklern verwendet werden können, um ihre Absichten für den Code zu bestätigen, während CEXs Sequenzen von Transaktionen sind, die zu einem Assertionsfehler führen, was bei der Behebung von Schwachstellen hilft.

Die direkte Kodierung in CHC ermöglicht eine unbegrenzte Modellprüfung, was bedeutet, dass Schwachstellen, die nach einer beliebigen endlichen Folge von Transaktionen auftreten, erfasst werden können, aber es entstehen Formeln, deren Lösung im allgemeinen Fall unentscheidbar ist. Aufgrund dieser wichtigen theoretischen Einschränkung ist die Bewertung ihres Verhaltens in praktischen Szenarien von entscheidender Bedeutung. Zu diesem Zweck wurde unser Tool, das die Kodierung implementiert, SOLICITOUS, das jetzt als CHC Model Checking Engine des offiziellen Solidity-Compilers fungiert, umfassend evaluiert mittels Ausführung von 22446 realen Verträgen auf der Ethereum-Blockchain. Darüber hinaus wurde ein Vergleich mit drei öffentlich verfügbaren Tools zur automatischen Verifizierung von Solidity-Assertions durchgeführt, nämlich SOLC-VERIFY (SRI), VERISOL (Microsoft) und MYTHRIL (ConsenSys). Die Ergebnisse (vgl. Tab. 2) zeigen, dass CHC-basierte Modellprüfung von Smart Contracts realisierbar ist und dass SOLICITOUS die anderen Tools übertrifft. Schließlich haben wir eine Erweiterung der Kodierung vorgeschlagen, um die genaue Reihenfolge und die Argumente von Funktionsaufrufen für eine bessere CEX-Generierung zu erfassen. Die Ergebnisse wurden in ACM TOPS [Ot23a] veröffentlicht.

Tab. 2: Bewertungsergebnisse ausgewählter Tools zur Überprüfung von Smart Contracts. Die verschiedenen Tools unterstützen eine unterschiedliche Anzahl von Benchmarks. Ein Benchmark gilt als verifiziert, wenn er als sicher oder unsicher eingestuft wird. Die besten Ergebnisse sind hervorgehoben.

	SOLICITOUS	SOLC-VERIFY	VERISOL	MYTHRIL
Benchmarks	**22446**	13310	12402	**22446**
Sicher	**6651**	103	201	0
Unsicher	8	0	9	**21**
Nicht schlüssig	2970	5601	230	728
Zeitüberschreitung	9058	3163	4032	19511
Werkzeug-Absturz	3759	4443	7930	2186
Geprüft	~**29%**	~0.7%	~1%	~0.09%

4 Validierung der Unerfüllbarkeitsergebnisse von SMT-Lösern

Automatisierte Reasoning-Engines für FOL sind eine Kernkomponente einer Vielzahl unterschiedlicher Ansätze zur symbolischen Modellprüfung. Ziel dieser Ansätze ist es, Techniken aus der Computerlogik zu nutzen, um riesige Suchräume effizient zu durchqueren, um festzustellen, ob Hardware- oder Softwaresysteme mit ihren Spezifikationen übereinstimmen. Viele Modellüberprüfungsprogramme reduzieren das Überprüfungsproblem letztlich auf Erfüllbarkeitsabfragen in FOL, die in einer für SMT-Löser geeigneten Form ausgedrückt

werden. Solche Abfragen können entweder erfüllbar sein, kurz SAT (nicht zu verwechseln mit Boolean satisfiability), was bedeutet, dass es eine Zuordnung zu den Variablen der SMT-Formel gibt, so dass sie als *wahr* ausgewertet wird, ansonsten unbefriedigend, kurz UNSAT. Da die Verwendung von SMT-Lösungen immer häufiger wird, wird die Korrektheit der Schlussfolgerungen des Lösers entscheidend. Effiziente Schlussfolgerungsalgorithmen sind jedoch komplex und können selbst Fehler enthalten, wie bei den jährlichen SMT-Wettbewerben oft aufgedeckt wird. Die Verwendung von Zeugnissen ist ein Mittel, um dringend benötigte Korrektheitsgarantien zu geben. Zeugnisse für zufriedenstellende Ergebnisse werden SAT-Modelle genannt, während Zeugnisse für nicht zufriedenstellende Ergebnisse UNSAT-Beweise genannt werden. Einige SMT-Löser, z.B. Z3, CVC4 und VERIT, bieten die Möglichkeit, UNSAT-Beweise in einem gegebenen Beweissystem zu erzeugen, was unter anderem dem Bedürfnis nach einer Erhöhung des Vertrauens in SMT-Löser entgegenkommen kann. Die Idee ist, dass die Beweise mit externen Prüfern wiedergegeben werden können. Ursprünglich waren dies Theorembeweiser wie COQ. Diese Beweisformate neigen jedoch dazu, große Zeugnisse zu erzeugen, was ihre Integration in SMT-basierte Werkzeuge erschwert.

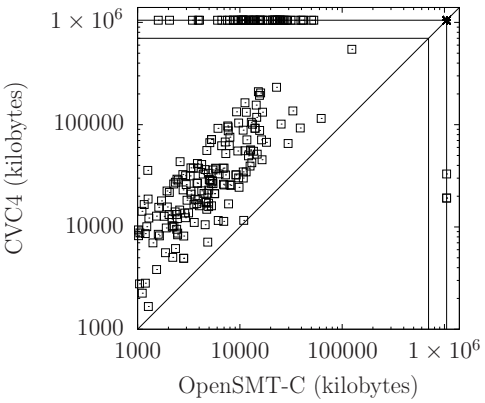

Abb. 2: QF_LRA Proof-Größen, die obere und die rechte Zeile stehen für Wertgrenze, Zeitüberschreitung und Speicherüberschreitung.

Wir haben ein neuartiges Beweisformat vorgeschlagen, mit dem Ziel, kompakte Beweise zu erzeugen, wobei der Schwerpunkt auf UNSAT-Beweisen für quantifikatorfreie Fragmente von FOL liegt. Die Beweise in unserem Format beginnen mit einer directed acyclic graph-Darstellung der Eingabeformel und enden im Wesentlichen in einem leeren Resolventen einer Auflösungswiderlegung. Das DRAT-Format wurde als Basis verwendet, um über das propositionale Fragment von FOL zu argumentieren, und wurde mit realen arithmetischen Zertifikaten basierend auf dem Farkas-Lemma, Lemmata für einfache Formen von Gomory-Schnitten, einer natürlichen Formalisierung der Gleichheit von Funktionen und einer Axiomatisierung der Tseitin-Transformation und der De-Morgan-Regeln erweitert. Der Schwerpunkt lag auf den Theorien von quantifier-free linear real arithmetic (QF_LRA), quantifier-free linear integer arithmetic (QF_LIA) und quantifier-free uninterpreted functions (QF_UF), die den meisten anderen SMT-Theorien zugrunde liegen. Wir haben den SMT-Löser OPENSMT instrumentiert, um Beweise in unserem Format zu erzeugen, und den Theory-Specific Witness Checker (TSWC) implementiert, um sie zu überprüfen. Die Auswertung der 16052 Benchmarks, die im SMT-LIB Repository für die drei unterstützten Theorien verfügbar sind, zeigt, dass das Format zu kleineren Beweisen führt, die mit einem geringen Overhead während des Lösens erzeugt werden können verglichen mit anderen Beweise erzeugenden Lösern, und effizient überprüft werden können. Zur Veranschaulichung zeigt Abb. 2 den

Kontrast zwischen den Größen der Beweise, die von dem instrumentierten OpenSMT und CVC4 erzeugt werden. Diese Ergebnisse wurden in DAC'21 [Ot21] veröffentlicht.

5 Validierung der Erfüllbarkeitsrgebnisse von CHC-Lösern

Verschiedene Fragmente von FOL eignen sich zur Unterstützung spezifischer Überprüfungsaufgaben. Ein Fragment von besonderem praktischen Interesse ist CHC, das zur Unterstützung von Schlussfolgerungen über das Verhalten von allgemeinen Programmen und nebenläufigen Systemen sowie von Smart Contracts verwendet wurde (vgl. Abschnitt 3). CHC-Löser wie Eldarica, Golem und Spacer, dienen zum Beispiel als Back-End-Reasoning-Engines von Verifikationstools für Programme, die in C/C++, Java, Rust und Solidity geschrieben sind, sowie für Android-Anwendungen. Trotz ihrer umfangreichen Verwendung in der Verifikation sind CHC-Löser selbst nicht vor Fehlern gefeit. Der jährlich stattfindende CHC-Wettbewerb CHC-COMP stieß auf ähnliche Probleme wie seine Gegenstücke SAT und SMT, da sich die konkurrierenden Löser bei bestimmten Benchmarks nicht einig waren. Die Eingabe eines CHC-Lösers ist eine Konjunktion logischer Implikationen, die uninterpretierte Prädikate enthält, wobei die Aufgabe des Lösers darin besteht, zu entscheiden, ob *falsch* abgeleitet werden kann oder nicht. Wenn dem so ist, gilt die Eingabe als unbefriedigend, ansonsten gilt sie als befriedigend. Ein Zeugnis für ein UNSAT-Ergebnis (nicht zufriedenstellend wie beim SMT Lösen), ein UNSAT-Beweis, sollte eine Erklärung enthalten, wie *falsch* abgeleitet werden kann, während ein Zeugnis für ein SAT-Ergebnis (zufriedentstellend), ein SAT-Modell, Interpretationen für alle Prädikate enthalten sollte dass alle Klauseln *wahr* werden lässt, mit der Folge, dass *falsch* nicht ableitbar ist.

Während die Erzeugung von Zeugnissen, seien es Modelle oder Beweise, ein gemeinsames Merkmal moderner CHC-Löser ist, sind die Bemühungen um die Validierung von Zeugnissen derzeit noch begrenzt. Die Validierung von Modellen kann über eine Reihe von SMT-Abfragen erfolgen. Da die Validierung von CHC-Modellen durch das Lösen von SMT-Problemen unterstützt wird, gilt die gleiche Sorge hinsichtlich der Korrektheit der Ergebnisse von CHC-Lösern auch der Validierung selbst, d. h. der Korrektheit der Ergebnisse von SMT-Lösern. Um dieses Problem anzugehen, haben wir einen zweischichtigen Validierungsansatz vorgeschlagen, der zusätzliche Garantien für die erzielten Ergebnisse bietet (vgl. Abb. 3). Die erste Schicht, die aus den für die Modellvalidierung verantwortlichen SMT-Abfragen besteht, wird durch eine zweite Schicht ergänzt, die aus der Erzeugung und Überprüfung von SMT-Beweisen besteht, wobei das erhaltene Ergebnis an den Benutzer oder das Werkzeug weitergeleitet wird, das mit dem CHC-Solver interagiert. Der Ansatz ist generisch in Bezug auf FOL-Theorien und Löser und ist außerdem sehr modular, so dass verschiedene SMT-Löser bei der Validierung verwendet werden können, was die Sicherheit weiter erhöht. Um den Ansatz praktikabel zu machen und eine Evaluierung durchzuführen, haben wir das modulAr consTrained Horn clauses modEl validatioN frAmework (ATHENA) entwickelt, das verschiedene Kombinationen von hochmodernen CHC- und SMT-Lösern bedienen kann. Konkret wurde der Rahmen verwendet, um die von drei CHC-Lösern erzeugten Modelle zu validieren, wobei jedes erzeugte Modell

von fünf Beweise-erzeugenden SMT-Lösern validiert wurde. Darüber hinaus wurden alle Beweise, die in den derzeit von automatisierten Proof Checkern unterstützten Formaten erzeugt wurden, überprüft. Alle 955 linear integer arithmetic-Benchmarks aus dem CHC-Wettbewerb von 2022 wurden für die Bewertung verwendet. Die Ergebnisse zeigen auf, dass die Modellvalidierung in der Praxis verwendet werden kann, wobei die Mehrheit der Modelle mit den verfügbaren Werkzeugen validiert werden kann, aber die Größe der Modelle problematisch ist. Darüber hinaus wurden neun Fehler in den bei der Evaluierung verwendeten Werkzeugen gefunden, von denen drei ungültige Modelle und Beweise betrafen, was die Notwendigkeit einer Zertifizierung von Verifikationswerkzeugen bestätigt. Die Ergebnisse dieser Arbeit wurden in iFM'23 [Ot23b] veröffentlicht.

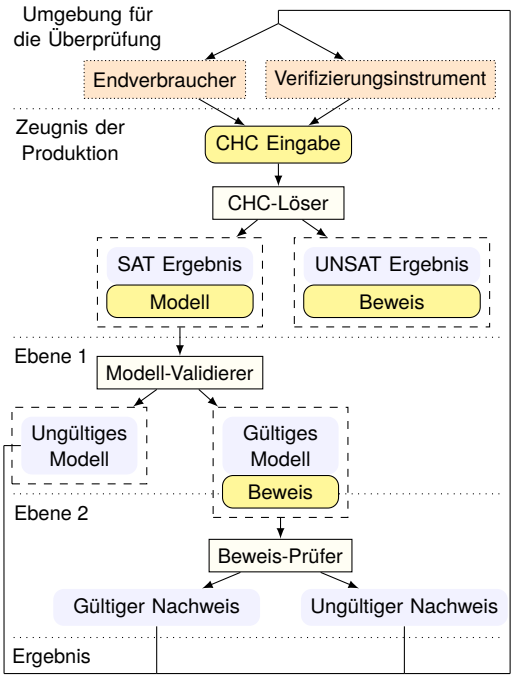

Abb. 3: Zweistufiger Validierungsansatz.

6 Ausblick

Die beiden wichtigsten Beobachtungen, die sich aus den Ergebnissen der Dissertation ableiten lassen, sind: 1. Die Anwendung formaler Verifikation auf Blockchain-Technologien kann sowohl praktikabel als auch skalierbar gemacht werden kann. 2. Die Validierung der Ergebnisse von Logiklösern kann effizient auf automatisierte und leichtgewichtige Weise durchgeführt werden. Aus der geleisteten Arbeit ergeben sich viele neue interessante Forschungsansätze. Betreffend den Modellprüfungsansatz für TLA$^+$, so sind seine Erweiterung zur strukturerhaltenden Kodierung der übrigen Datenstrukturen eine direkte Folgemaßnahme. Eine andere, wesentlich aufwendigere Richtung, ist die Entwicklung eines effizienten Ansatzes für die unbeschränkte Modellprüfung für TLA$^+$. Zu den direkten Folgemaßnahmen für den Modellprüfungsansatz für Solidity gehört die Verbesserung der Kodierung, um Szenarien wie Aufrufe von bekanntem externen Code zu behandeln. Für die Unerfüllbarkeitsbeweise von SMT ist die Hinzufügung von Unterstützung für andere Theorien eine klare Richtung. Für die CHC-Modellvalidierung ist die Unterstützung anderer Theorien ebenfalls eine klare Richtung, wobei eine ergänzende Richtung die Entwicklung eines Ansatzes für die Validierung von CHC-Unzufriedenheitsbeweisen ist. Schließlich ist

die Einbettung der Zeugniserzeugung und -validierung in Verifikationswerkzeuge sowohl für Beweise als auch für Modelle eine interessante, relevante, wie auch unerforschte Richtung.

Literaturverzeichnis

[KKT19] Konnov, I.; Kukovec, J.; Tran, T.-H.: TLA+ Model Checking Made Symbolic. Proc. of the ACM on Programming Languages 3 (OOPSLA), S. 1–30, 2019.

[Ma20] Marescotti, M. et al.: Accurate Smart Contract Verification Through Direct Modelling. In: 9th Inter. Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA). S. 178–194, 2020.

[Ot21] Otoni, R. et al.: Theory-Specific Proof Steps Witnessing Correctness of SMT Executions. In: 58th ACM/IEEE Design Automation Conference (DAC). S. 541–546, 2021.

[Ot23a] Otoni, R. et al.: A Solicitous Approach to Smart Contract Verification. ACM Transactions on Privacy and Security 26 (2), S. 1–28, 2023.

[Ot23b] Otoni, R. et al.: CHC Model Validation with Proof Guarantees. In: 18th Inter. Conf. on integrated Formal Methods (iFM). S. 62–81, 2023.

[Ot23c] Otoni, R. et al.: Symbolic Model Checking for TLA+ Made Faster. In: 29th Inter. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). S. 126–144, 2023.

[Ot23d] Otoni, R. B.: Automated Verification of Blockchain Technologies with Correctness Guarantees, Verfügbar unter `https://sonar.ch/usi/documents/326503`, PhD Dissertation, Schweiz: Università della Svizzera italiana, 2023.

Rodrigo Otoni studierte Informatik an der Universidade Federal de Sergipe (BSc) und der Universidade Federal de Pernambuco (MSc), beide in Brasilien. Während dieser Zeit war er Gaststudent an der University of York (UK), nachdem er vom brasilianischen Ministerium für Wissenschaft und Technologie ein Stipendium erhalten hatte. Seinen Doktortitel erwarb er 2023 an der Università della Svizzera italiana, Schweiz, wo er derzeit als Postdoktorand tätig ist. Seine Forschung konzentriert sich auf die Überschneidung zwischen formaler Verifikation und Bereichen wie verteilte Systeme, Sicherheit und Programmiersprachen.