Universität Paderborn

Fakultät für Elektrotechnik, Informatik und Mathematik

Masterarbeit

Bounded Model Checking für partielle Systeme

Nils Timm

Vorgelegt bei:
Prof. Dr. Heike Wehrheim und Prof. Dr. Hans Kleine Büning

Paderborn, Juni 2009

Erklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Paderborn, den 15. Juni 2009

Inhaltsverzeichnis

Kapitel 1.

Einleitung

Software findet eine immer größer werdende Verbreitung als Bestandteil von eingebetteten Systemen. Anwendungsbereiche sind Flugzeuge, Kraftwerke, medizinische Apparaturen sowie diverse elektronische Geräte des täglichen Gebrauchs. Wie aus diesen Beispielen bereits hervorgeht, sind mit dem Einsatz solcher Systeme oftmals sicherheitskritische Aufgaben verbunden. Softwarefehler können extreme Kosten oder sogar den Tod von Menschen verursachen. Daher stellen Verifikationsverfahren, formale Beweismethoden zur Überprüfung der Korrektheit von Software, einen wichtigen Forschungsbereich der Softwaretechnik dar. Die Forschung auf diesem Gebiet wird vor allem dadurch immer wieder vor neue Herausforderungen gestellt, dass die Komplexität zum Einsatz kommender Softwaresysteme kontinuierlich steigt. Insbesondere Parallelität und die damit verbundene Interaktion von Einzelkomponenten eines Gesamtsystems führen zu Zustandsräumen extremer Größe. Diese müssen zum Nachweis von Systemeigenschaften meist vollständig durchsucht werden. Die Schwierigkeit mit der Handhabung solcher Zustandsräume bei der Verifikation wird als das so genannte *State Explosion Problem* [CGJ+01] bezeichnet.

Als etabliertes Verifikationsverfahren existiert neben dem Theorembeweisen das *Model Checking* [CGP99], welches im Gegensatz zu ersterem vollautomatisch und somit ohne die Notwendigkeit einer Benutzerinteraktion abläuft. Model Checking beruht auf der Überprüfung eines Systemmodells bezüglich der Gültigkeit einer formalen Spezifikation. Die Spezifikation beschreibt dabei die gestellten Anforderungen an das System – beispielsweise Sicherheits- oder Lebendigkeitseigenschaften. Sie wird in Temporallogik formuliert. Als Modell des zu Grunde liegenden Systems wird üblicherweise eine Kripke Struktur verwendet. Diese definiert sich über Zustände, darin geltende atomare Propositionen bezüglich Systemeigenschaften, sowie Transitionen zwischen Zuständen und bildet auf diese Weise das Systemverhalten nach. Zum Nachweis der Korrektheit, beziehungsweise der Gültigkeit der temporallogischen Anforderungen für das Modell, ist im Allgemeinen die vollständige Durchsuchung des Zustandsraumes erforderlich. Naive Model Checking Verfahren scheitern bereits für eine Vielzahl realistisch großer Systeme an dem State Explosion Problem. Die Modellprüfung schlägt aufgrund der hohen Komplexität der Systembeschreibung fehl. Daher ist die Entwicklung von Techniken zur effizienten Handhabung großer Zustandsräume das Kerngebiet der Forschung im Bereich der Verifikation.

Große Fortschritte wurden bereits durch die Einführung des *symbolischen Model Check-*

ing [BCM+92] erzielt. Entsprechende Verfahren bauen das Systemmodell nicht explizit auf, sie verwenden stattdessen eine kodierte und somit kompakte Darstellung des Zustandsraumes. Insbesondere binäre Entscheidungsdiagramme (BDDs) haben sich zur impliziten Repräsentation des Modells etabliert. Einen neueren Ansatz bildet das *Bounded Model Checking* [BCC+03]. Hier wird das Model Checking Problem auf das aussagenlogische Erfüllbarkeitsproblem (SAT) reduziert. Diese Methode hat in jüngster Zeit von den erheblichen Fortschritten der SAT-Solver Technologie profitiert [PBG05].

Symbolisches Model Checking hat die Klasse effizient verifizierbarer Systeme signifikant vergrößert. Doch auch BDD- und SAT-basierte Verifikationstechniken besitzen bezüglich der Bewältigung des State Explosion Problems gewisse Grenzen. Daher sind nicht nur Verfahren zur effizienten Repräsentation des Systemmodells, sondern auch solche zur tatsächlichen Verkleinerung des Zustandsraumes, Gegenstand der Forschung. *Reduktionstechniken* finden insbesondere im Bereich paralleler Systeme Anwendung. Das kompositionelle Model Checking [Kai92] beruht beispielsweise auf einer komponentenweisen Verifikation des Gesamtsystems und umgeht damit den vollständigen Aufbau eines Systemmodells. Eine eigene Klasse von Reduktionsverfahren bilden *Abstraktionstechniken* [CGL92]. Diesen liegt die Idee zu Grunde, dass zum Nachweis gewisser Eigenschaften keine vollständige Systembeschreibung benötigt wird. Abstraktionen bilden das Systemverhalten lediglich in einem begrenzten Umfang nach. Somit wird eine Verkleinerung des Zustandsraumes auf Kosten der Präzision des Modells erzielt.

Das klassische Model Checking setzt einen vollständig bekannten Zustandsraum voraus – ein Erfordernis, das im Falle abstrakter Systembeschreibungen nicht erfüllt ist. Darüber hinaus existieren auch Systeme, deren erreichte Zustände sich aus einem selbstadaptiven Verhalten ergeben und daher initial unbekannt sind [KM07]. Die Verifikation solcher Systeme mit nur teilweise bekannten Zustandsräumen lässt sich durch das *mehrwertige Model Checking* [CDE+03] realisieren. Den begrenzten Informationen über das zu Grunde liegende System beziehungsweise der Unschärfe einer Abstraktion wird dabei durch die Verwendung *partieller* Kripke Strukturen Rechnung getragen. Transitionen und atomare Propositionen in Zuständen können hier Werte einer mehrwertigen Logik annehmen, also neben *wahr* oder *falsch* beispielsweise auch *unbekannt* sein. Gleiches gilt für die Gültigkeit temporallogischer Formeln.

Mehrwertige Model Checking Probleme lassen sich sowohl direkt anhand der partiellen Systembeschreibung lösen [CDE+03], als auch auf eine konstante Anzahl klassischer Model Checking Instanzen reduzieren [GC03]. Letzterer Ansatz ist insbesondere deshalb von Interesse, da nach der Reduktion auf effiziente Standard Model Checker zurückgegriffen werden kann. Bounded Model Checking Techniken für partielle Kripke Strukturen liegen ebenfalls im Blickpunkt aktueller Forschung. Während [AY04] ein Verfahren beschreibt, welches bereits auf der Modellebene die Reduktion auf eine Reihe binärer Probleminstanzen vornimmt, behält der Ansatz von [Weh08] die Mehrwertigkeit bis zur Ebene der aussagenlogischen Kodierung bei und erzeugt daraus abschließend zwei Instanzen des klassischen Erfüllbarkeitsproblems.

Die Technik aus [Weh08] stellt eine vielversprechende Herangehensweise zur Verifikation nur teilweise bekannter Systeme dar. Die dort beschriebene Übersetzung eines dreiwertigen Model Checking Problems in eine aussagenlogische Formel ist eine direkte Generalisierung des ursprünglichen Bounded Model Checking Konzepts aus [BCC+99] auf die Dreiwertigkeit. Für die Gültigkeit atomarer Propositionen in Zuständen steht mit *unbekannt* ein zusätzlicher Wahrheitswert zur Verfügung. Die beiden notwendigen Erfüllbarkeitstests lassen sich durch die Verwendung effizienter Standard SAT-Solver realisieren. Allerdings ist das Verfahren bisher nur für Kripke Strukturen mit partieller Beschriftungsfunktion definiert. Diese Funktion legt den Wahrheitswert der Propositionen in den einzelnen Zuständen fest.

Ist der Zustandsraum eines Systems nicht vollständig bekannt, so kann dies auch die Existenz von Transitionen betreffen. Je präziser die unbekannten Eigenschaften des betrachteten Systems auf der Modellebene wiedergegeben werden, desto eher lassen sich beim mehrwertigen Model Checking definite Aussagen bezüglich der Spezifikation treffen. Ein wesentliches Qualitätsmerkmal eines solchen Verfahrens ist also neben der Effizienz auch die Ausdrucksstärke der formalen Sprache, die den partiellen Systembeschreibungen zu Grunde liegt.

In dieser Arbeit wird daher eine Erweiterung der Technik aus [Weh08] entwickelt, die für Kripke Strukturen mit dreiwertiger Beschriftungs- *und* Transitionsfunktion definiert ist und somit auch die Modellierung unbekannter Zustandsübergänge gestattet. Dies umfasst sowohl einen konzeptionellen als auch einen praktischen Teil.

Die Konzeption ist verbunden mit der Definition des BMC-Problems für die erweiterte Form dreiwertiger Kripke Strukturen sowie der Entwicklung einer aussagenlogischen Kodierung, welche auch der partiellen Transitionsfunktion Rechnung trägt. Zudem erfolgt der Beweis, dass diese Kodierung ein korrektes Bounded Model Checking auf Basis zweier SAT-Instanzen ermöglicht.

Der praktische Teil dieser Arbeit beinhaltet die Umsetzung des Konzepts. Darin eingeschlossen ist die Evaluation geeigneter Datenstrukturen zur Repräsentation aussagenlogisch kodierter BMC-Probleme sowie geeigneter Transformationstechniken zur Umformung der Kodierung in ein von SAT-Solvern verarbeitbares Format. Darüber hinaus ist die Java-Implementierung eines Bounded Model Checkers für dreiwertige Kripke Strukturen Bestandteil dieser Arbeit. Diese ermöglicht die Überprüfung temporallogischer Eigenschaften anhand partieller Systembeschreibungen, die mit dem Abstraktionsverfahren 3Spot [SWW09] erzeugt wurden.

Die weitere Arbeit gliedert sich wie folgt:

- Im nachfolgenden Kapitel werden grundlegende Aspekte des Model Checking und des Bounded Model Checking erläutert, die zum Verständnis dieser Arbeit hilfreich sind.

- In Kapitel 3. wird die entwickelte Bounded Model Checking Technik für Kripke Strukturen mit dreiwertiger Beschriftungs- und Transitionsfunktion umfassend vorgestellt. Zunächst wird das dreiwertige Model Checking Problem definiert und die Bounded Semantiken der Temporallogik LTL werden im Kontext partieller Kripke Strukturen betrachtet. Darauf aufbauend wird für das dreiwertige BMC-Problem eine aussagenlogische Kodierung mit zusätzlicher Konstanten *unbekannt* definiert, welche sich auf zwei klassische SAT-Instanzen reduzieren lässt. Zudem wird die Korrektheit dieses Ansatzes bewiesen.

- Die praktische Umsetzung der entwickelten Bounded Model Checking Technik wird im vierten Kapitel behandelt. Dazu werden zunächst die einzelnen Schritte der Technik näher betrachtet und effiziente Umsetzungsverfahren vorgestellt. Anschließend erfolgt eine Beschreibung des im Rahmen dieser Arbeit implementierten Bounded Model Checkers.

- Das abschließende Kapitel fasst die wichtigsten Ergebnisse und Erkenntnisse dieser Arbeit zusammen und gibt zudem einen Ausblick auf sinnvolle weiterführende Forschungen im Bereich Bounded Model Checking für partielle Kripke Strukturen.

Kapitel 2.

Grundlagen

In diesem Kapitel erfolgt eine kurze Einführung in das temporallogische Model Checking. Es werden grundlegende Begriffe und Definitionen erläutert, die für das Verständnis der weiteren Arbeit hilfreich sind. Insbesondere das Bounded Model Checking wird näher betrachtet.

2.1. Temporallogisches Model Checking

Model Checking [CGP99] ist ein vollautomatisches Verfahren zur Verifikation von Hardware- und Softwaresystemen. Es beruht auf der Überprüfung einer zustandsbasierten Systembeschreibung bezüglich der Gültigkeit formal spezifizierter Eigenschaften. Als Systembeschreibung wird üblicherweise eine Kripke Struktur verwendet, welche den Zustandsraum des betrachteten Systems modelliert. Übergänge zwischen Zuständen werden als Transitionen bezeichnet. Während auf der Systemebene Zustände durch konkrete Variablenwerte und Programmzählerpositionen definiert sind, betrachtet man auf der Modellebene lediglich atomare Propositionen bezüglich der Variablenwerte. Die Proposition '$x = 0$' entspricht beispielsweise der Aussage, dass die Variable x den Wert 0 besitzt.

Definition 2.1.1: Kripke Struktur
Eine Kripke Struktur *über einer Menge von atomaren Propositionen AP ist ein Tupel $M = (S, S_0, R, L)$, wobei gilt*

- *S ist eine endliche Zustandsmenge,*
- *$S_0 \subseteq S$ ist eine Menge von Startzuständen,*
- *$R \subseteq S \times S$ ist eine totale Transitionsrelation. Es wird vorausgesetzt, dass R die Bedingung $\forall s \exists s' : (s, s') \in R$ erfüllt und somit jeder Zustand eine ausgehende Transition besitzt.*
- *$L : S \to 2^{AP}$ ist eine Beschriftungsfunktion. Sie ordnet den Zuständen eine Menge atomarer Propositionen aus AP zu.*

Zur Beschreibung des sequentiellen Verhaltens des modellierten Systems betrachtet man Pfade der Kripke Struktur. Ein Pfad $\pi = (s_0, s_1, ...)$ ist eine unendliche Sequenz von Zuständen, wobei gilt $s_0 \in S_0$ und $\forall s_i, s_{i+1} : (s_i, s_{i+1}) \in R$. Demnach beginnt jeder Pfad in einem initialen Zustand der Kripke Struktur und jedes Zustandspaar entlang des Pfades charakterisiert eine gültige Transition. Ferner bezeichnet $\pi(i)$ den i-ten Zustand eines Pfades π, während $\pi^i = (s_i, s_{i+1}, ...)$ den i-ten Suffix beschreibt. Pfade und die in den einzelnen Pfadzuständen geltenden atomaren Propositionen bilden die Grundlage für die Überprüfung formal spezifizierter Eigenschaften anhand der Kripke Struktur.

Da Anforderungen an das System insbesondere mit zeitlichen Abläufen verbunden sind, verwendet man zur Spezifikation dieser Anforderungen eine temporale Logik. Solche Logiken stellen eine Erweiterung der klassischen Aussagenlogik dar. Neben den Junktoren $\neg, \vee$ und $\wedge$ stehen hier zusätzliche Temporaloperatoren zur Verfügung, mit denen sich zeitlich bedingte Aussagen formulieren lassen. Die in der Praxis am häufigsten vertretenen temporalen Logiken sind die *Linear Time Temporal Logic* (LTL) [MP92] und die *Computation Tree Logic* (CTL) [INH96]. Während LTL die Formulierung linearer zeitlicher Abfolgen ermöglicht, lassen sich mit CTL verzweigte Zeitfolgen beschreiben. Die beiden Logiken besitzen bezüglich der Menge spezifizierbarer Eigenschaften sowohl gemeinsame, als auch jeweils exklusive Elemente. Die Vereinigung von LTL und CTL bezeichnet man mit CTL*.

Im Weiteren wird die Linear Time Temporal Logic betrachtet. Wichtige Operatoren sind hierbei **X** (Next), **G** (Globally), **F** (Finally) und **U** (Until). Beim Model Checking werden LTL-Formeln über der Menge atomarer Propositionen der betrachteten Kripke Struktur definiert. Zur Überprüfung der Gültigkeit werden Pfade betrachtet. So drückt die Formel $\mathbf{X}p$ beispielsweise aus, dass die Proposition p im Nachfolger des aktuellen Zustands gilt. $\mathbf{G}p$ bedeutet, dass p in allen Zuständen des Pfades gilt.

Definition 2.1.2: LTL-Semantiken
Sei π ein Pfad einer Kripke Struktur M und φ eine LTL-Formel. Dann ist $[\pi \models \varphi]$ (φ gültig entlang π) wie folgt definiert:

$$
\begin{aligned}
[\pi \models p] &= p \in L(\pi(0)) \\
[\pi \models \neg\varphi] &= \neg[\pi \models \varphi] \\
[\pi \models \varphi_1 \wedge \varphi_2] &= [\pi \models \varphi_1] \wedge [\pi \models \varphi_2] \\
[\pi \models \varphi_1 \vee \varphi_2] &= [\pi \models \varphi_1] \vee [\pi \models \varphi_2] \\
[\pi \models \mathbf{X}\varphi] &= [\pi^1 \models \varphi] \\
[\pi \models \mathbf{G}\varphi] &= \bigwedge_{i \in \mathbb{N}} [\pi^i \models \varphi]
\end{aligned}
$$

$$[\pi \models \mathbf{F}\varphi] \quad = \quad \bigvee_{i\in\mathbb{N}}[\pi^i \models \varphi]$$
$$[\pi \models \varphi_1 \mathbf{U} \varphi_2] \quad = \quad \bigvee_{i\in\mathbb{N}}([\pi^i \models \varphi_2] \wedge \bigwedge_{j=0}^{i-1}[\pi^j \models \varphi_1])$$

Diese Semantiken beschreiben die Gültigkeit temporallogischer Formeln für einzelne Pfade π einer Kripke Struktur M. Gilt $[\pi \models \varphi]$, so bezeichnet man π auch als *Zeugen* beziehungsweise *Zeugenpfad* für φ. Das Model Checking Problem ist die Frage nach der Gültigkeit von LTL-Formeln für ganze Kripke Strukturen.

Definition 2.1.3: Model Checking Problem
Die Gültigkeit einer LTL-Formel φ für eine Kripke Struktur M ist definiert durch

$$[M \models \varphi] := \bigwedge_{\pi,\, \pi\ Pfad\ von\ M} [\pi \models \varphi]$$

Nachfolgend werden die zuvor beschriebenen Grundlagen des LTL-Model Checking noch einmal anhand eines Beispiels veranschaulicht. Ein bekanntes Problem in vielen Softwaresystemen ist der exklusive Zugriff auf Ressourcen. Mehrere Prozesse versuchen einen kritischen Abschnitt zu betreten, in dem sich höchstens ein Prozess gleichzeitig befinden darf. Abbildung 2.1.1 zeigt eine Kripke Struktur M, die das entsprechende System für zwei Prozesse A und B modelliert. Die erwünschte Sicherheitseigenschaft „Niemals beide Prozesse gleichzeitig im kritischen Abschnitt“ lässt sich wie folgt in LTL formulieren: $\varphi = \mathbf{G}(\neg A_{CR} \wedge \neg B_{CR})$. Initial befindet sich kein Prozess im kritischen Abschnitt. Über die Transitionen sind Zustände erreichbar in denen sich entweder *nur A* oder *nur B* im kritischen Abschnitt befindet. Der Zustand s_3, in dem beide Prozesse im kritischen Abschnitt sind, ist unerreichbar. Es gilt offensichtlich für alle gültigen Pfade π: $[\pi \models \varphi]$ und somit auch $[M \models \varphi]$.

Eine weitere Anforderung an das System, die sich durch LTL ausdrücken lässt ist, dass jeder Prozess unendlich oft den kritischen Abschnitt betritt: $\psi = \mathbf{G}(\mathbf{F}(A_{CR})) \wedge \mathbf{G}(\mathbf{F}(B_{CR}))$. Dies ist eine so genannte Lebendigkeitseigenschaft. Für den Pfad $\pi = (s_0, s_1, s_0, s_1, ...)$ gilt ψ *nicht*: $[\pi \not\models \varphi]$, denn die Proposition B_{CR} ist an keiner Position von π gültig. Somit ist ψ auch für M ungültig: $[M \not\models \varphi]$. Einen solchen Pfad, der die zu überprüfende Eigenschaft widerlegt, nennt man deshalb auch *Gegenbeispiel*.

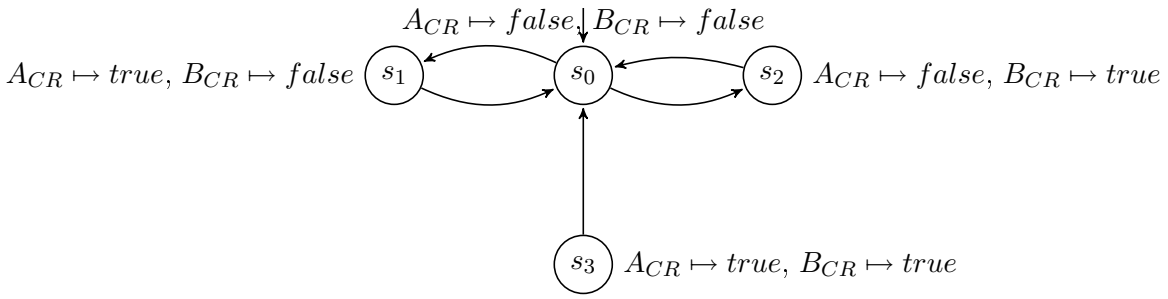

Abbildung 2.1.1: Kripke Struktur

Das Model Checking Problem ist die Frage nach der Gültigkeit einer temporallogischen Formel φ für eine Kripke Struktur M: $[M \models \varphi]$. Zur Entscheidung dieses Problems existieren verschiedene Ansätze. Ein solcher Ansatz zur Verifikation von LTL-Eigenschaften ist das automatenbasierte Model Checking. Automaten lassen sich zur Repräsentation von Sprachen unendlicher Wörter einsetzen. Sowohl Kripke Strukturen M, als auch LTL-Formeln φ beschreiben Sprachen $LANG(M)$ und $LANG(\varphi)$. Jeder unendliche Pfad π einer Kripke Struktur stellt ein Wort $w(\pi)$ über dem Alphabet 2^{AP} dar. Eine Zustandssequenz $\pi = (s_0, s_1, ...)$ korrespondiert zu dem Wort $w(\pi) = (L(s_0)L(s_1)...)$. Für die Kripke Struktur in Abbildung 2.1.1 ergibt sich für $\pi = (s_0, s_1, ...)$ beispielsweise das Wort $w(\pi) = (\{\}\{A_{CR}\}...)$.

Die Sprache einer LTL-Formel φ entspricht der Menge aller Wörter $w(\pi)$, für die gilt $[\pi \models \varphi]$. Für das Model Checking werden Kripke Struktur und LTL-Formel jeweils in die korrespondierenden Automaten überführt. Offensichtlich gilt $[M \models \varphi]$ genau dann, wenn alle Pfade π von M Zeugen für φ sind. Dies ist gleichbedeutend mit der Eigenschaft $LANG(M) \subseteq LANG(\varphi)$. Gilt diese Teilmengenbeziehung, so ist die Schnittmenge von $LANG(M)$ mit der komplementären Sprache von φ leer. Ein automatenbasierter LTL-Model Checker überprüft daher den Wahrheitswert von $LANG(M) \cap LANG(\neg\varphi) = \emptyset$. (Es gilt $LANG(\neg\varphi) = \overline{LANG(\varphi)}$.)

Ein konkreter Model Checker, der dieser Vorgehensweise folgt ist SPIN [Hol97]. Dieser gestattet die Verifikation von Systemen, die mittels der C-ähnlichen Programmiersprache PROMELA modelliert werden. Atomare Propositionen lassen sich als boolesche Aussagen über Variablenwerten definieren. Auf dieser Basis ist die Überprüfung von LTL-Eigenschaften möglich. Automatenbasiertes Model Checking ist ein explizites Verfahren. Der Zustandsraum des betrachteten Systems wird explizit aufgebaut und zum Nachweis der spezifizierten Eigenschaft durchsucht. Diese Tatsache beschränkt das automatenbasierte Model Checking auf die Verifikation vergleichsweise kleiner Systeme.

Symbolische Model Checker [BCM+92] verwenden hingegen eine implizite und somit kompaktere Repräsentation des Zustandsraumes. Weit verbreitet ist das CTL-Model Checking anhand binärer Entscheidungsdiagramme (BDDs). Diese Datenstrukturen er-

möglichen eine boolesche Kodierung der Transitionsrelation. Für eine Systembeschreibung, dargestellt als BDD, und eine CTL-Formel φ wird ein weiteres Entscheidungsdiagramm aufgebaut, welches genau die Zustände repräsentiert in denen φ gilt. Ein solches Werkzeug ist beispielsweise NuSMV [CCG+02]. BDD-basierte Model Checker sind in der Lage, weitaus größere Modelle zu verifizieren, als dies bei expliziten Model Checkern der Fall ist. Jedoch besitzen auch diese symbolischen Verfahren gewisse Grenzen bezüglich Handhabung komplexer Systeme.

Eine alternative Herangehensweise stellt das Bounded Model Checking (BMC) [BCC+99] dar. Diese Verifikationstechnik basiert auf der Reduktion des Model Checking Problems auf das aussagenlogische Erfüllbarkeitsproblem und kann somit unter Verwendung von SAT-Solvern realisiert werden. BMC ist eine zu BDD-basierten Verfahren komplementäre Methode. Jede dieser beiden Techniken besitzt für gewisse Probleminstanzen Effizienzvorteile gegenüber der anderen. Auch NuSMV verfügt in der aktuellen Version über eine BMC-Komponente. Bounded Model Checking bildet die Grundlage für das in dieser Arbeit entwickelte Verfahren und wird daher im nachfolgenden Abschnitt umfassend vorgestellt.

2.2. Bounded Model Checking

Bounded Model Checking (BMC) [BCC+99] ist eine Variante des klassischen Model Checking, dessen Probleminstanzen sich auf das aussagenlogische Erfüllbarkeitsproblem (SAT) reduzieren lassen. Somit profitiert dieses Verfahren von effizienten SAT-Solvern. Das Model Checking Problem ist die Frage nach der Gültigkeit einer temporallogischen Formel für *alle* Pfade einer Kripke Struktur. Die im vorherigen Abschnitt vorgestellten Verfahren weisen eben diese Gültigkeit nach, oder widerlegen sie durch die Rückgabe eines Gegenbeispiels in Form eines Pfades, für den die Formel nicht gilt. Bounded Model Checking basiert hingegen auf der direkten Suche solcher Gegenbeispiele. Soll nachgewiesen werden, dass eine Formel φ für eine Kripke Struktur M gilt ($[M \models \varphi]$), so wird beim BMC überprüft, ob in M ein Pfad π mit $[\pi \models \neg\varphi]$ existiert. Ist dies der Fall so stellt π ein Gegenbeispiel für $[M \models \varphi]$ dar. Bounded Model Checking beruht also auf dem Nachweis *existentieller* Eigenschaften. Somit ist ein differenzierter Begriff des Model Checking Problems notwendig.

Definition 2.2.1: Universelles und Existentielles Model Checking Problem
Der universelle Wert *einer LTL-Formel φ für eine Kripke Struktur M ist definiert durch*

$$[M \models_A \varphi] := \bigwedge_{\pi,\ \pi\ Pfad\ von\ M} [\pi \models \varphi]$$

Der existentielle Wert *einer LTL-Formel φ für eine Kripke Struktur M ist definiert durch*

$$[M \models_E \varphi] := \bigvee_{\pi,\ \pi\ Pfad\ von\ M} [\pi \models \varphi]$$

Jedes universelle Model Checking Problem besitzt ein äquivalentes existentielles Model Checking Problem. Es gilt:

$$[M \models_A \varphi] = \neg[M \models_E \neg\varphi]$$

Diese Gleichung verdeutlicht noch einmal das Prinzip des BMC. Die universelle Gültigkeit einer Eigenschaft lässt sich durch den Nachweis der existentiellen Gültigkeit der negierten Eigenschaft widerlegen. Die Beschränkung auf existentielle Probleme ist begründet durch die Tatsache, dass beim Bounded Model Checking nur Pfadpräfixe einer festen Länge $k \in \mathbb{N}$ betrachtet werden. Anhand solcher unvollständigen Pfade ist der direkte Nachweis universeller Eigenschaften der Kripke Struktur nicht möglich. Konkrete BMC-Implementierungen gehen iterativ bezüglich k vor. Sie starten mit $k = 0$ und suchen somit im ersten Iterationsschritt nach einem Gegenbeispiel der Länge Null für die nachzuweisende temporallogische Formel. Existiert ein solches Gegenbeispiel, beziehungsweise ein Zeugenpfad für die negierte Formel, so ist die Gültigkeit der Eigenschaft widerlegt. Andernfalls wird der so genannte Bound k schrittweise inkrementiert und die Suche so lange fortgeführt, bis ein entsprechend längeres Gegenbeispiel gefunden, oder eine festgelegte obere Schranke für k erreicht wurde. Abbildung 2.2.1 zeigt eine illustrative Darstellung dieser iterativen Vorgehensweise.

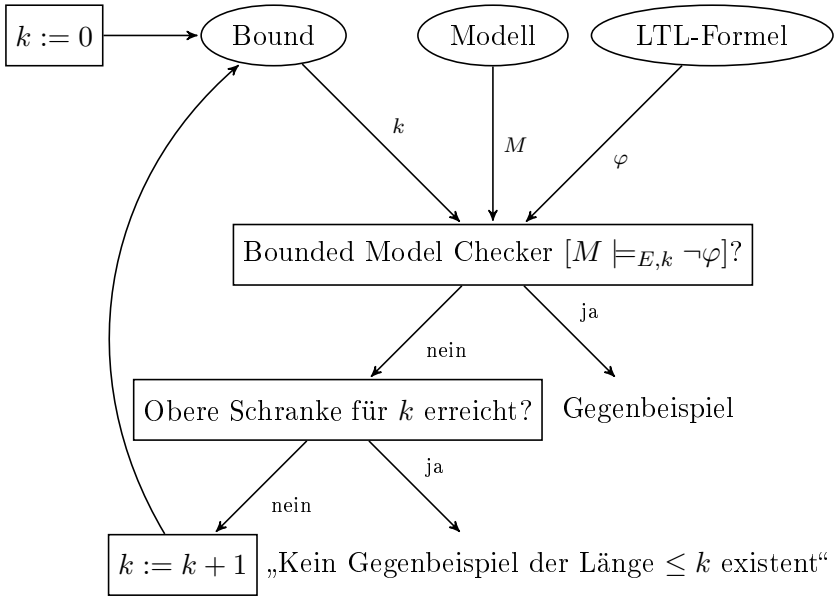

Abbildung 2.2.1: Iteratives Bounded Model Checking

Es ist zu beachten, dass der durch k beschränkte Präfix eines Pfades endlich ist, jedoch durchaus einen unendlichen Pfad repräsentieren kann, sofern innerhalb des Präfixes eine Schleife existiert, genauer gesagt eine Transition von der letzten Präfixposition zu einer vorherigen vorhanden ist. Besitzt ein Präfix allerdings keine Schleife, dann lassen sich über globale Eigenschaften des zu Grunde liegenden Pfades keine Aussagen treffen. Daher ist es beim Bounded Model Checking notwendig, zwischen Pfadpräfixen *mit* und *ohne* Schleife zu differenzieren.

Definition 2.2.2: k-Schleife
Sei π ein Pfad einer Kripke Struktur M. Für $l, k \in \mathbb{N}$ mit $l \leq k$ besitzt π eine (k,l)-Schleife, *falls gilt $(\pi(k), \pi(l)) \in R$ und π ist von der Form $u \cdot v^{\omega}$, wobei $u = (\pi(0), ..., \pi(l-1))$ und $v = (\pi(l), ..., \pi(k))$. π besitzt eine* k-Schleife, *falls ein $l \in \mathbb{N}$ existiert mit $l \leq k$ für das π eine (k,l)-Schleife besitzt.*

Abbildung 2.2.2 veranschaulicht noch einmal die Unterschiede zwischen Pfadpräfixen mit und ohne k-Schleife. 2.2.1(a) zeigt den 2-Präfix eines Pfades π. In jedem Zustand gilt die atomare Proposition p. s_2 besitzt jedoch keine ausgehende Transition zu einem vorherigen Zustand des Präfix. Somit ist nicht bekannt, ob für s_2 ein Nachfolgezustand existiert in dem ebenfalls p gilt. Der Pfadpräfix ist also kein Zeuge für die LTL-Formel $\mathbf{G}p$, respektive kein Gegenbeispiel für $\mathbf{F}\neg p$. Der 3-Präfix von π (2.2.1(b)) besitzt jedoch eine Transition (s_3, s_2) und damit auch eine $(3,2)$-Schleife. Demnach repräsentiert dieser Präfix den unendlichen Pfad $\pi = (s_0, s_1, s_2, s_3, s_2, s_3, ...)$. Zudem gilt auch in Zustand s_3 die Proposition p. Der Pfadpräfix der Länge 3 ist somit ein hinreichendes Gegenbeispiel für $\mathbf{F}\neg p$.

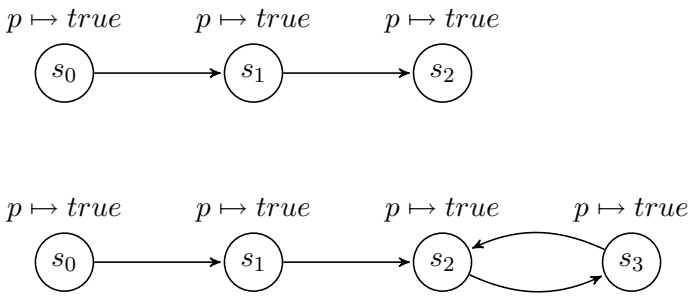

Abbildung 2.2.2: 2-Präfix (a) und 3-Präfix (b) eines Pfades π

Die Gültigkeit einer LTL-Formel für einen Pfadpräfix ist also abhängig vom Vorhandensein einer Schleife. Dementsprechend existieren beim Bounded Model Checking keine allgemeinen LTL-Semantiken, sondern man unterscheidet hier zwischen Präfixen mit und ohne Schleife.

Definition 2.2.3: Bounded LTL-Semantiken
Sei φ eine LTL-Formel, π ein Pfad ohne k-Schleife *für $k \in \mathbb{N}$ und eine Kripke Struktur M. Ferner sei $i \in \mathbb{N}$ die aktuelle Position in π. Dann ist $[\pi \models_k^i \varphi]$ wie folgt definiert:*

$$
\begin{array}{lcl}
[\pi \models_k^i p] & = & p \in L(\pi(i)) \\
[\pi \models_k^i \neg\varphi] & = & \neg[\pi \models_k^i \varphi] \\
[\pi \models_k^i \varphi_1 \wedge \varphi_2] & = & [\pi \models_k^i \varphi_1] \wedge [\pi \models_k^i \varphi_2] \\
[\pi \models_k^i \varphi_1 \vee \varphi_2] & = & [\pi \models_k^i \varphi_1] \vee [\pi \models_k^i \varphi_2] \\
[\pi \models_k^i \mathbf{X}\varphi] & = & [\pi \models_k^{i+1} \varphi] \text{ mit } [\pi \models_k^{k+1} \varphi] = false \\
[\pi \models_k^i \mathbf{G}\varphi] & = & false \\
[\pi \models_k^i \mathbf{F}\varphi] & = & \bigvee_{j=i}^{k} [\pi \models_k^j \varphi] \\
[\pi \models_k^i \varphi_1 \mathbf{U} \varphi_2] & = & \bigvee_{j=i}^{k} [(\bigwedge_{n=i}^{j-1} [\pi \models_k^n \varphi_1]) \wedge [\pi \models_k^j \varphi_2]]
\end{array}
$$

Falls π eine k-Schleife besitzt, dann ist $[\pi \models_k^i \varphi] := [\pi^i \models \varphi]$, zudem ist $[M \models_{E,k} \varphi] := \bigvee_{\pi,\, \pi \, Pfad \, von \, M} [\pi \models_k^0 \varphi]$

$[M \models_{E,k} \varphi]$ beschreibt also die Aussage: In der Kripke Struktur M existiert ein Pfadpräfix der Länge k, welcher ein Zeuge für die LTL-Formel φ ist. Die Restriktion, nur Präfixe einer festen Länge zu betrachten hat zur Folge, dass Bounded Model Checking im Allgemeinen nicht vollständig ist. Zieht man jedoch für den Bound $k \in \mathbb{N}$ alle möglichen Werte in Betracht, so ist BMC äquivalent zum klassischen Model Checking und somit auch vollständig. Es gilt das nachfolgende Theorem aus [BCC+99].

Theorem 2.2.1:
Sei φ eine LTL-Formel und M eine Kripke Struktur. Dann gilt $[M \models_E \varphi]$ genau dann wenn $\exists k \in \mathbb{N}$ mit $[M \models_{E,k} \varphi]$.

Betrachtet man einen Pfad π, der gemäß den Bounded LTL-Semantiken ein Zeuge der Länge k für eine Formel φ ist, so ist π auch unter den Semantiken des klassischen Model Checking ein Zeuge für φ. Dies gilt, da die Bounded Semantiken restriktiver als die unbeschränkten Semantiken sind. Umgekehrt gilt, dass jede Kripke Struktur M eine endliche

Anzahl von Zuständen besitzt und somit auch jeder unendliche Pfad π von M. Jedes π endet folglich in einer Schleife. Ist ein solcher Pfad π ein Zeuge für eine LTL-Formel φ, dann existiert auch ein $k \in \mathbb{N}$, so dass der k-Präfix von π ebenfalls die Zeugeneigenschaft erfüllt.

Bounded Model Checking ist also ein korrektes und bei einer inkrementellen Vorgehensweise auch vollständiges Verfahren. Besonderheit des BMC-Problems ist es, dass sich dieses auf das aussagenlogische Erfüllbarkeitsproblem reduzieren lässt. Dazu wird eine aussagenlogische Formel $[\![M, \varphi]\!]_k$ konstruiert, welche genau dann erfüllbar ist, wenn gilt $[M \models_{E,k} \varphi]$. Jede erfüllende Bewertung dieser Formel beschreibt einen Zeugenpfadpräfix der Länge k für φ. Bounded Model Checking lässt sich also unter der Verwendung effizienter SAT-Solver realisieren. Die Definition der aussagenlogischen Kodierung eines BMC-Problems findet sich in [BCC$^+$99]. In dieser Arbeit werden jedoch mehrwertige Bounded Model Checking Probleme betrachtet. Damit verbunden ist auch Einführung eines erweiterten Erfüllbarkeitsbegriffs, welcher in Abschnitt 3.3. thematisiert wird. Das nachfolgende Kapitel behandelt die Entwicklung eines Bounded Model Checking Verfahrens für partielle Kripke Strukturen. Solche Kripke Strukturen besitzen eine dreiwertige Beschriftungs- und Transitionsfunktion. Die Gültigkeit atomarer Propositionen in Zuständen kann nicht nur *wahr* oder *falsch*, sondern auch *unbekannt* sein. Gleiches gilt für das Vorhandensein von Zustandsübergängen. Dies erfordert sowohl eine Anpassung der Definition der Bounded LTL-Semantiken, als auch die Konstruktion einer speziellen aussagenlogischen Kodierung mit einem zusätzlichen Wahrheitswert.

Kapitel 3.

Konzeption

Das klassische temporallogische Model Checking [CGP99] setzt einen vollständig bekannten Zustandsraum voraus, üblicherweise repräsentiert durch eine Kripke Struktur. Im Gegensatz zu vollständigen Systembeschreibungen ermöglichen partielle Kripke Strukturen die Modellierung von Zustandsräumen mit unbekannten Eigenschaften, wie sie sich beispielsweise durch Abstraktionen von komplexen Softwaresystemen oder durch inkonsistente Spezifikationen ergeben. Abstraktionsverfahren [GC06] nehmen eine wichtige Rolle unter den Reduktionstechniken im Model Checking ein. Sie erlauben die Verkleinerung des Zustandsraumes und stellen somit einen Ansatz dar, dem so genannten *State Explosion Problem* [CGJ+01] entgegenzuwirken. Dies geschieht auf Kosten der Präzision der Modellierung.

Zur Verifikation solcher partieller Systembeschreibungen kommt das mehrwertige Model Checking (englisch: *multi-valued Model Checking – MVMC*) [BG99] zum Einsatz. Grundlage dieser Technik sind Aussagenlogiken mit mehr als zwei Wahrheitswerten, wie die dreiwertige kleenesche Logik [Fit94]. Zur Lösung von mehrwertigen Model Checking Problemen existieren in der Praxis zwei verschiedene Herangehensweisen. So verarbeitet beispielsweise der Model Checker χChek [CDE+03] *direkt* eine gegebene mehrwertige Probleminstanz. Eine Alternative ist die Reduktion des MVMC-Problems auf eine feste Anzahl von zweiwertigen Model Checking Problemen. Diese Methode bietet den Vorteil, die eigentliche Modellprüfung unter der Verwendung etablierter Standard Model Checker durchzuführen. Solche Model Checker setzen zur Repräsentation des Zustandsraumes entweder binäre Entscheidungsdiagramme (BDDs) [BCM+92] ein oder aber sie reduzieren das Model Checking Problem auf das Erfüllbarkeitsproblem der zweiwertigen Aussagenlogik (SAT). Letzterer Ansatz fällt in das Forschungsgebiet des Bounded Model Checking (BMC) [BCC+03], welches in jüngster Zeit von den erheblichen Fortschritten der SAT-Solver Technologie profitiert hat [PBG05].

In [Weh08] wurde erstmals ein BMC-Verfahren für partielle Systembeschreibungen vorgestellt, das die Mehrwertigkeit des Problems bis zur Ebene der aussagenlogischen Kodierung aufrecht erhält und daraus abschließend zwei Standard SAT-Instanzen erzeugt. So ermöglicht diese Technik die Verifikation nur teilweise bekannter Systeme unter der Ausnutzung effizienter SAT-Solver und stellt damit einen neuen Forschungsansatz zur Bewältigung des State Explosion Problems dar. Zur Modellierung des Zustandsraumes

werden in [Weh08] Kripke Strukturen mit partieller Beschriftungsfunktion verwendet. Der Wahrheitswert einer atomaren Proposition in einem Zustand kann demnach *wahr*, *falsch* oder *unbekannt* sein. Die Unschärfe einer inkonsistenten Spezifikation oder der Abstraktion eines Softwaresystems kann allerdings auch das Vorhandensein von Transitionen betreffen, so dass nicht bekannt ist, ob zwei Zustände untereinander direkt erreichbar sind. Dieser Fall wird in [Weh08] bisher nicht berücksichtigt.

Im Hinblick auf die praktische Umsetzung eines MVMC-Verfahrens ist es von großem Interesse, die unbekannten Eigenschaften des betrachteten Systems möglichst präzise auf der Modellebene zu formulieren und somit auch unbekannte Zustandsübergänge einzubeziehen. Daher wird in diesem Kapitel eine Erweiterung das Verifikationstechnik aus [Weh08] entwickelt, welche für Kripke Strukturen mit partieller Beschriftungs- *und* Transitionsfunktion ausgelegt ist.

3.1. Model Checking für partielle Kripke Strukturen

Nachfolgend wird das Konzept des Model Checking für partielle Kripke Strukturen grundlegend vorgestellt. Zudem wird die Reduktion des dreiwertigen Model Checking Problems auf zwei klassische Model Checking Probleme gezeigt.

Eine partielle Kripke Struktur beschreibt einen Zustandsraum mit unbekannten Komponenten. Dies kann sowohl die Gültigkeit atomarer Propositionen in einzelnen Zuständen betreffen [BG99], als auch das Vorhandensein von Transitionen zwischen Zustandspaaren [KP02]. Zudem liegt jeder Definition einer partiellen Kripke Struktur eine beliebig mehrwertige Logik zu Grunde. Diese legt fest, wie viele Grade an Ungewissheit sich modellieren lassen. Im einfachsten Fall steht genau *ein* zusätzlicher Wahrheitswert $\bot$ mit der Bedeutung *unbekannt* zur Verfügung. Weitere Abstufungen sind jedoch ebenso möglich (Siehe z. B. [CED01]).

Die Bounded Model Checking Technik aus [Weh08] ist für Kripke Strukturen mit dreiwertiger Beschriftungsfunktion definiert. In dieser Arbeit wird das Verfahren auf Kripke Strukturen mit zusätzlich dreiwertiger Transitionsfunktion ausgedehnt.

Definition 3.1.1: Partielle Kripke Struktur
Eine partielle Kripke Struktur *über einer Menge von atomaren Propositionen AP ist ein Tupel $M = (S, S_0, R, L)$, wobei gilt*

- *S ist eine endliche Zustandsmenge,*
- *$S_0 \subseteq S$ ist eine Menge von Startzuständen,*
- *$R : S \times S \to \{true, \bot, false\}$ ist eine Transitionsfunktion. Es wird vorausgesetzt, dass R die Bedingung $\forall s \exists s' : R(s, s') > false$ erfüllt und somit jeder Zustand eine ausgehende Transition mit dem Wert true oder $\bot$ besitzt.*

- $L : S \times AP \rightarrow \{true, \bot, false\}$ *ist eine Beschriftungsfunktion. Sie liefert einen Wahrheitswert für die Gültigkeit von atomaren Propositionen in den Zuständen der partiellen Kripke Struktur.*

Eine Kripke Struktur ist vollständig, *falls gilt* $\forall s \in S, p \in AP : L(s,p) \in \{true, false\}$ *und* $\forall s \in S, s' \in S : R(s,s') \in \{true, false\}$.

Das sequentielle Verhalten des zu Grunde liegenden Systems lässt sich anhand von Pfaden der partiellen Kripke Struktur beschreiben. Ein Pfad $\pi = (s_0, s_1, ...)$ von M ist eine unendliche Sequenz von Zuständen, wobei gilt $s_0 \in S_0$ und $\forall s_i, s_{i+1} : R(s_i, s_{i+1}) > false$. Ein gültiger Pfad kann nach dieser Definition also auch Transitionen mit Wahrheitswert $\bot$ enthalten [KP02]. Ferner definiert $\pi(i)$ den i-ten Zustand eines Pfades π, während $\pi^i = (s_i, s_{i+1}, ...)$ den i-ten Suffix von π beschreibt.

Pfade einer Kripke Struktur werden zur Überprüfung der Gültigkeit temporallogischer Formeln betrachtet. Die nachfolgenden Abschnitte beschränken sich dabei auf die lineare Temporallogik (LTL) [MP92]. LTL-Formeln werden definiert über atomaren Propositionen aus AP unter der Verwendung der Junktoren $\neg, \vee, \wedge$ sowie der Temporaloperatoren **X** (*Next*), **G** (*Globally*), **F** (*Finally*) und **U** (*Until*). Durch die Dreiwertigkeit von Transitionen und Beschriftungen einer partiellen Kripke Struktur ergeben sich für die Gültigkeit von temporallogischen Formeln die möglichen Wahrheitswerte $true, \bot$ und $false$. Daher wird zur Interpretation vorkommender boolescher Operatoren die kleenesche Logik L_3 [Fit94] verwendet, welche durch die Wahrheitstafeln in Abbildung 3.1.1 beschrieben wird. Für die Wahrheitsordnung $false \leq \bot \leq true$ gilt: Die Konjunktion berechnet das *Minimum* ihrer Argumente, die Disjunktion das *Maximum*.

$\wedge$	$true$	$\bot$	$false$
$true$	$true$	$\bot$	$false$
$\bot$	$\bot$	$\bot$	$false$
$false$	$false$	$false$	$false$

$\vee$	$true$	$\bot$	$false$
$true$	$true$	$true$	$true$
$\bot$	$true$	$\bot$	$\bot$
$false$	$true$	$\bot$	$false$

$\neg$	
$true$	$false$
$\bot$	$\bot$
$false$	$true$

Abbildung 3.1.1: Wahrheitstafeln für L_3

Nachfolgend werden die LTL-Semantiken für Pfade partieller Kripke Strukturen beschrieben. Im Gegensatz zu den Semantiken für Pfade vollständiger Kripke Strukturen kann es hier durch die erweiterte Beschriftungsfunktion L zum Auftreten des Wahrheitswerts $\bot$ für atomare Propositionen p und daraus zusammengesetzte komplexere Formeln φ kommen. Dieser Fall wurde bereits in [Weh08] berücksichtigt. Die Ausdehnung der Mehr-

wertigkeit auf die Transitionsfunktion erfordert zudem, die LTL-Semantiken in Abhängigkeit von R zu definieren. Besitzt ein Pfad eine Transition mit Wahrheitswert $\perp$, so ist dies auch bei der Überprüfung temporallogischer Eigenschaften zu berücksichtigen. Die folgende Definition orientiert sich an den Semantiken der mehrwertigen Temporallogik mv-CTL* [KP02].

Definition 3.1.2: LTL-Semantiken
Sei π ein unendlicher Pfad einer partiellen Kripke Struktur M und φ eine LTL-Formel. Dann ist $[\pi \models \varphi]$ wie folgt definiert:

$$
\begin{aligned}
[\pi \models p] &= L(\pi(0), p) \\
[\pi \models \neg\varphi] &= \neg[\pi \models \varphi] \\
[\pi \models \varphi_1 \wedge \varphi_2] &= [\pi \models \varphi_1] \wedge [\pi \models \varphi_2] \\
[\pi \models \varphi_1 \vee \varphi_2] &= [\pi \models \varphi_1] \vee [\pi \models \varphi_2] \\
[\pi \models \mathbf{X}\varphi] &= R(\pi(0), \pi(1)) \wedge [\pi^1 \models \varphi] \\
[\pi \models \mathbf{G}\varphi] &= \textstyle\bigwedge_{i \in \mathbb{N}}([\pi^i \models \varphi] \wedge R(\pi(i), \pi(i+1))) \\
[\pi \models \mathbf{F}\varphi] &= \textstyle\bigvee_{i \in \mathbb{N}}([\pi^i \models \varphi] \wedge \bigwedge_{j=0}^{i-1} R(\pi(j), \pi(j+1))) \\
[\pi \models \varphi_1 \mathbf{U} \varphi_2] &= \textstyle\bigvee_{i \in \mathbb{N}}([\pi^i \models \varphi_2] \wedge \bigwedge_{j=0}^{i-1} \left[[\pi^j \models \varphi_1] \wedge R\left(\pi(j), \pi(j+1)\right)\right])
\end{aligned}
$$

Im Folgenden werden die Besonderheiten der LTL-Semantiken für partielle Kripke Strukturen noch einmal anhand eines Beispiels erläutert. Abbildung 3.1.2 zeigt eine Kripke Struktur, die sowohl atomare Propositionen in Zuständen, als auch Transitionen mit Wert $\perp$ besitzt. Für alle Pfade π, welche mit dem Präfix (s_0, s_1, s_2) beginnen, gilt $[\pi \models p\mathbf{U}q] = \perp$. Der Präfix ist offensichtlich ein Zeuge für die Eigenschaft $p\mathbf{U}q$, er enthält jedoch die Transition (s_1, s_2) mit $R(s_1, s_2) = \perp$, so dass sich insgesamt für die Gültigkeit des LTL-Ausdrucks *unbekannt* ergibt. Es gilt hingegen $[\pi \models \mathbf{X}p] = \mathit{true}$ – die Transition (s_1, s_2) ist hier nicht relevant, da schon der kürzere Präfix (s_0, s_1) ein Zeuge für $\mathbf{X}p$ ist. Für den Pfad $\pi' = (s_0, s_2, ...)$ gilt $[\pi' \models \mathbf{G}p] = \perp$, da der Wert von $L(s_2, p)$ nicht bekannt ist.

Zur Vereinfachung werden Transitionen mit Wert $\perp$ nachfolgend auch $\perp$-*Transitionen* genannt. Pfade, die mindestens eine $\perp$-Transition enthalten, heißen dementsprechend $\perp$-*Pfade*.

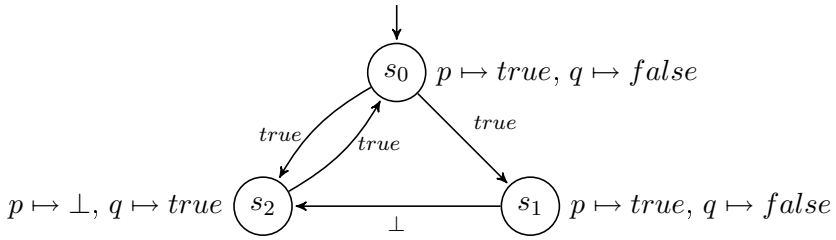

Abbildung 3.1.2: Partielle Kripke Struktur

Die zuvor beschriebenen Semantiken betreffen die Gültigkeit von LTL-Formeln für einzelne Pfade. Beim Model Checking ist insbesondere die Gültigkeit für ganze Kripke Strukturen von Interesse: $[M \models \varphi]$. Im Hinblick auf das spätere Bounded Model Checking wird hier zwischen universellen und existentiellen Model Checking Problemen differenziert (Vergleiche Abschnitt 2.2.).

Definition 3.1.3: Mehrwertiges Model Checking Problem
Der universelle Wert *einer LTL-Formel φ für eine partielle Kripke Struktur M ist definiert durch*

$$[M \models_A \varphi] := \bigwedge_{\pi,\ \pi\ Pfad\ von\ M} [\pi \models \varphi]$$

Der existentielle Wert *einer LTL-Formel φ für eine partielle Kripke Struktur M ist definiert durch*

$$[M \models_E \varphi] := \bigvee_{\pi,\ \pi\ Pfad\ von\ M} [\pi \models \varphi]$$

Zur Lösung solcher MVMC-Probleme wurden Verfahren entwickelt, welche den Gültigkeitswert einer temporallogischen Formel direkt anhand einer partiellen Kripke Struktur bestimmen. Ein Beispiel hierfür ist der mehrwertige Model Checker χChek [CDE+03].

Der Fokus dieser Arbeit liegt hingegen auf der Reduktion des MVMC-Problems auf eine feste Anzahl herkömmlicher Model Checking Probleme und somit auf einem indirekten Verfahren. Eine Voraussetzung für den hier verfolgten indirekten Ansatz ist die Beschränkung auf die Klasse der positiven LTL-Formeln, LTL$^+$. LTL$^+$-Formeln enthalten kein Negationszeichen. Diese Einschränkung senkt jedoch nicht die Ausdrucksstärke. Jede LTL-Formel lässt sich in eine äquivalente LTL$^+$-Formel übersetzen, wenn sie für *komplement-geschlossene* Kripke Strukturen [BG00] ausgewertet wird. In solchen Kripke Strukturen besitzt jedes $p \in AP$ ein Komplement $\bar{p}$ und es gilt für alle Zustände $s \in S$: $L(s,p) = \neg L(s,\bar{p})$. Die Übersetzung von LTL nach LTL$^+$ umfasst dementsprechend die

Umformung in Negationsnormalform und die Ersetzung aller negierten atomaren Propositionen $\neg p$ durch $\bar{p}$. Die Formel $\neg\mathbf{G}(p \wedge \neg q)$ wird somit zu $\mathbf{F}(\bar{p} \vee q)$. Abbildung 3.1.3 zeigt die komplement-geschlossene Ausführung der partiellen Kripke Struktur aus Abbildung 3.1.2.

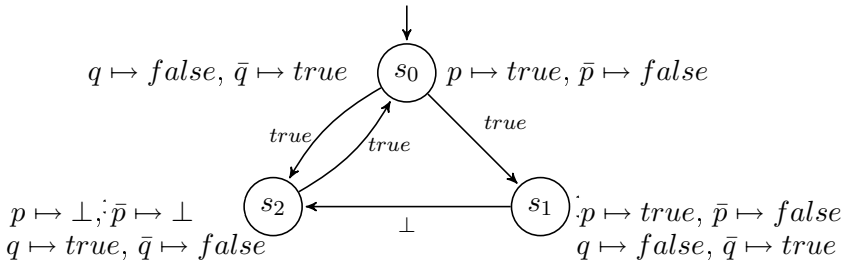

Abbildung 3.1.3: Komplement-geschlossene partielle Kripke Struktur

In [Weh08] wurde bereits die Reduktion des Model Checking Problems für Kripke Strukturen mit dreiwertiger Beschriftungsfunktion auf zwei klassische Model Checking Probleme gezeigt. Nachfolgend wird eine Generalisierung dieser Reduktion für partielle Kripke Strukturen gemäß Definition 3.1.1 vorgestellt, beginnend mit einer erweiterten Definition der Vervollständigung.

Definition 3.1.4: Vervollständigung
Für die Transitionsfunktion R einer partiellen Kripke Struktur sei R^o die optimistische Vervollständigung *und R^p die* pessimistische Vervollständigung *von R mit*

$$R^o(s,s') = \begin{cases} true & falls\ R(s,s') = \bot \\ R(s,s') & sonst \end{cases}$$

$$R^p(s,s') = \begin{cases} false & falls\ R(s,s') = \bot \\ R(s,s') & sonst \end{cases}$$

Für die Beschriftungsfunktion L einer partiellen Kripke Struktur sei L^o die optimistische Vervollständigung *und L^p die* pessimistische Vervollständigung *von L mit*

$$L^o(s,p) = \begin{cases} true & falls\ L(s,p) = \bot \\ L(s,p) & sonst \end{cases}$$

$$L^p(s,p) = \begin{cases} false & falls\ L(s,p) = \bot \\ L(s,p) & sonst \end{cases}$$

Für eine partielle Kripke Struktur $M = (S, S_0, R, L)$ *sei* $M(R^x, L^y) = (S, S_0, R^x, L^y)$ *eine vollständige Kripke Struktur mit* $x, y \in \{o, p\}$

Das nachfolgende Theorem stellt eine Verallgemeinerung der Ergebnisse aus [BG00] dar. Dort betrifft die Vervollständigung lediglich die Beschriftungsfunktion. Die Betrachtung partieller Kripke Strukturen mit dreiwertiger Transitionsfunktion erfordert zudem die Unterscheidung von universellen und existentiellen Model Checking Problemen bei der Reduktion auf zwei klassische Probleminstanzen [GC03]. $[M \models_A \varphi]$ ist nur dann wahr, wenn *alle* Pfade von M Zeugen für φ sind. Daher müssen auch alle $\perp$-Pfade diese Zeugeneigenschaft erfüllen. Zur Überprüfung wird dementsprechend R zu R^o vervollständigt. Im existentiellen Fall muss für $[M \models_E \varphi] = true$ mindestens ein Zeugenpfad *ohne* $\perp$-Transition für φ vorhanden sein, somit wird R hier zu R^p vervollständigt.

Theorem 3.1.1:
Sei M *eine partielle Kripke Struktur und* φ *eine LTL*$^+$*-Formel. Dann gilt*

$$[M \models_A \varphi] = \begin{cases} true & falls\ M(R^o, L^p) \models_A \varphi \\ false & falls\ M(R^p, L^o) \not\models_A \varphi \\ \perp & sonst \end{cases}$$

$$[M \models_E \varphi] = \begin{cases} true & falls\ M(R^p, L^p) \models_E \varphi \\ false & falls\ M(R^o, L^o) \not\models_E \varphi \\ \perp & sonst \end{cases}$$

Jedes Model Checking Problem für Kripke Strukturen mit dreiwertiger Beschriftungs- und Transitionsfunktion lässt sich also gemäß Theorem 3.1.1 auf zwei klassische Model Checking Probleme reduzieren. Ähnliche Reduktionstechniken für höherwertige Model Checking Probleme finden sich in [KP02] und [GC03].

3.2. Bounded Semantiken

Nachfolgend wird das zuvor beschriebene Model Checking für partielle Kripke Strukturen im Kontext des Bounded Model Checking betrachtet.

In der Praxis ist vorrangig die Lösung universeller Model Checking Probleme von Interesse. Das Prinzip des Bounded Model Checking ist es hingegen, Zeugenpfade für temporallogische Eigenschaften zu finden und somit die existentielle Gültigkeit zu überprüfen. Jedes universelle Model Checking Problem lässt sich jedoch in ein äquivalentes

existentielles Problem transformieren [BCC+99].

Satz 3.2.1:
Sei M eine partielle Kripke Struktur und φ eine LTL-Formel. Dann gilt:

$$[M \models_A \varphi] = \neg[M \models_E \neg\varphi]$$

Die Überprüfung der universellen Gültigkeit von LTL-Formeln mittels Bounded Model Checking entspricht also einer Suche nach der Existenz von Gegenbeispielen.

Grundidee des BMC ist es, bei der Suche nach Zeugenpfaden beziehungsweise Gegenbeispielen nur endliche Präfixe von Pfaden zu betrachten. Der Bound $k \in \mathbb{N}$ beschränkt dabei die Anzahl der berücksichtigten Präfixpositionen. Während des Model Checking wird k schrittweise inkrementiert – solange bis ein Gegenbeispiel gefunden oder eine festgelegte obere Schranke für k erreicht wurde. Der durch k beschränkte Präfix ist endlich, er kann jedoch durchaus einen unendlichen Pfad repräsentieren, sofern innerhalb des Präfixes eine Schleife existiert. Besitzt ein Präfix allerdings keine Schleife, so lässt sich über das infinite Verhalten des zu Grunde liegenden Pfades keine Aussage treffen. Daher ist es beim Bounded Model Checking notwendig, zwischen Pfadpräfixen *mit* und *ohne* Schleife zu unterscheiden.

Definition 3.2.1: k-Schleife
Sei π ein Pfad einer partiellen Kripke Struktur M. Für $l, k \in \mathbb{N}$ mit $l \leq k$ besitzt π eine (k,l)-Schleife, *falls gilt $R(\pi(k), \pi(l)) > false$ und π ist von der Form $u \cdot v^\omega$, wobei $u = (\pi(0), ..., \pi(l-1))$ und $v = (\pi(l), ..., \pi(k))$. π besitzt eine* k-Schleife, *falls ein $l \in \mathbb{N}$ existiert mit $l \leq k$ für das π eine (k,l)-Schleife besitzt.*

Aufgrund der partiellen Transitionsfunktion R besitzt ein Pfad π also auch dann eine (k,l)-Schleife, wenn es sich bei $(\pi(k), \pi(l))$ um eine $\perp$-Transition handelt.

Aus der Definition der k-Schleife, insbesondere der Tatsache, dass jeder Pfad mit k-Schleife in einem Zyklus endet, ergibt sich das folgende Korollar.

Korollar 3.2.1:
Sei π ein Pfad, $j \in \mathbb{N}$ und π besitzt eine j-Schleife. Dann gilt $\forall k, k \geq j : \pi$ besitzt eine k-Schleife.

Die Gültigkeit einer LTL-Formel für einen Pfadpräfix ist also abhängig vom Vorhandensein einer Schleife. Dies spiegelt sich auch in der differenzierten Definition der Bounded LTL-Semantiken wider. Genau wie bei den unbeschränkten Semantiken im vorherigen Abschnitt muss auch hier die Transitionsfunktion R berücksichtigt werden.

Definition 3.2.2: Bounded LTL$^+$-Semantiken
Sei φ eine LTL$^+$-Formel, π ein Pfad ohne k-Schleife *für $k \in \mathbb{N}$ und eine komplementgeschlossene Kripke Struktur M. Ferner sei $i \in \mathbb{N}$ die aktuelle Position in π. Dann ist $[\pi \models_k^i \varphi]$ wie folgt definiert:*

$$
\begin{array}{lcl}
[\pi \models_k^i p] & = & L(\pi(i), p) \\
[\pi \models_k^i \bar{p}] & = & L(\pi(i), \bar{p}) \\
[\pi \models_k^i \varphi_1 \wedge \varphi_2] & = & [\pi \models_k^i \varphi_1] \wedge [\pi \models_k^i \varphi_2] \\
[\pi \models_k^i \varphi_1 \vee \varphi_2] & = & [\pi \models_k^i \varphi_1] \vee [\pi \models_k^i \varphi_2] \\
[\pi \models_k^i \mathbf{X}\varphi] & = & R(\pi(i), \pi(i+1)) \wedge [\pi \models_k^{i+1} \varphi] \text{ mit } [\pi \models_k^{k+1} \varphi] = false \\
[\pi \models_k^i \mathbf{G}\varphi] & = & false \\
[\pi \models_k^i \mathbf{F}\varphi] & = & \bigvee_{j=i}^{k} [\bigwedge_{n=i}^{j-1} R(\pi(n), \pi(n+1)) \wedge [\pi \models_k^j \varphi]] \\
[\pi \models_k^i \varphi_1 \mathbf{U} \varphi_2] & = & \bigvee_{j=i}^{k} [\bigwedge_{n=i}^{j-1} ([\pi \models_k^n \varphi_1] \wedge R(\pi(n), \pi(n+1))) \wedge [\pi \models_k^j \varphi_2]]
\end{array}
$$

Falls π eine k-Schleife besitzt, dann ist $[\pi \models_k^i \varphi] := [\pi^i \models \varphi]$, zudem ist $[M \models_{E,k} \varphi] := \bigvee_{\pi,\, \pi \, Pfad \, von \, M} [\pi \models_k^0 \varphi]$

Die hier beschriebenen Bounded Semantiken stellen eine Approximation der existentiellen Semantiken aus Definition 3.1.3 bezüglich der Wahrheitsordnung $false \leq \bot \leq true$ dar. Für ein festes k kann der Wert von $[M \models_{E,k} \varphi]$ nur kleiner oder gleich dem Wert von $[M \models_E \varphi]$ sein (Vergleiche [Weh08]). Dies verdeutlicht sich an dem nachfolgenden Beispiel. Sei M die partielle Kripke Struktur in Abbildung 3.2.1. Zur Überprüfung von $[M \models_A \mathbf{G}p]$ wird iterativ nach einem Gegenbeispiel gesucht, genauer gesagt nach einem Zeugen für $[M \models_E \mathbf{F}\bar{p}]$. Für die Bounded Semantiken ergibt sich

$[M \models_{E,0} \mathbf{F}\bar{p}] = false$
$[M \models_{E,1} \mathbf{F}\bar{p}] = \bot$
$[M \models_{E,2} \mathbf{F}\bar{p}] = true$

Aus $[M \models_{E,1} \mathbf{F}\bar{p}] = \bot$ folgt, dass $[M \models_E \mathbf{F}\bar{p}]$ nicht *falsch* sein kann. Somit liegt bereits ein Gegenbeispiel für $[M \models_A \mathbf{G}p]$ vor. Ein Bounded Model Checker, der bezüglich k inkrementell vorgeht, kann also mit der Rückgabe eines Gegenbeispiels terminieren, sobald gilt $[M \models_{E,k} \varphi] > false$.

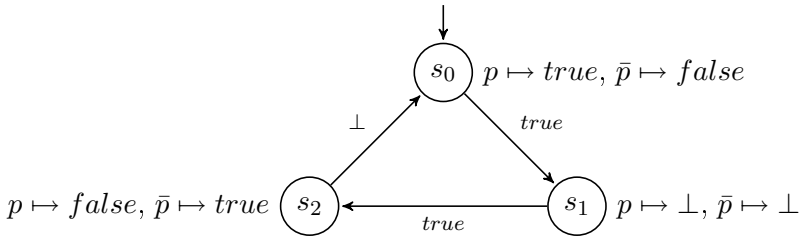

Abbildung 3.2.1: Komplement-geschlossene partielle Kripke Struktur

Aus der Definition der Bounded LTL$^+$-Semantiken lässt sich zudem das folgende Lemma herleiten, welches für den Korrektheitsbeweis der SAT-Reduktion eines BMC Problems von Bedeutung ist (Vergleiche Abschnitt 3.3.).

Lemma 3.2.1:
Sei π ein Pfad einer partiellen Kripke Struktur M, φ eine LTL$^+$-Formel und $k \in \mathbb{N}$. Zudem sei $[\pi \models^i_k \varphi]$ definiert gemäß den Bounded LTL$^+$-Semantiken für Pfade ohne *k-Schleife. Dann gilt*

$$[\pi \models^i_k \varphi] \leq [\pi \models^i_{k+1} \varphi]$$

Beweis: Siehe Anhang.

3.3. Reduktion auf das aussagenlogische Erfüllbarkeitsproblem

Nachdem die Semantiken des Bounded Model Checking im vorherigen Abschnitt beschrieben wurden, geht es im nächsten Schritt darum, das BMC-Problem auf das aussagenlogische Erfüllbarkeitsproblem zu reduzieren. Im zweiwertigen Fall wird hierzu für eine vollständige Kripke Struktur M, eine LTL-Formel φ und einen Bound k die aussagenlogische Formel $F = [\![M, \varphi]\!]_k$ konstruiert, welche genau dann erfüllbar ist, wenn gilt $[M \models_{E,k} \varphi]$ [BCC$^+$99]. Bei der Verwendung von Bounded Model Checking zur Verifikation partieller Kripke Strukturen muss jedoch berücksichtigt werden, dass mit $\bot$ ein dritter möglicher Wahrheitswert von $[\![M, \varphi]\!]_k$ existiert. In [Weh08] wird dazu ein neues Erfüllbarkeitsproblem sat_3 definiert und die Konstruktion von $[\![M, \varphi]\!]_k$ angepasst, so dass gilt:

$$[[\![M, \varphi]\!]_k sat_3] = [M \models_{E,k} \varphi]$$

Der Erfüllbarkeitstest für sat_3 ist auf zwei normale Erfüllbarkeitstests reduzierbar, somit lassen sich Standard SAT-Solver zur Lösung von sat_3-Problemen einsetzen.

Die in [Weh08] beschriebene SAT-Kodierung von Bounded Model Checking Problemen ist für Kripke Strukturen mit partieller Beschriftungsfunktion definiert. Im Folgenden wird eine Erweiterung dieser Kodierung vorgestellt, welche auch Kripke Strukturen mit dreiwertigen Transitionen berücksichtigt.

Die Konstruktion von $[\![M, \varphi]\!]_k$ gliedert sich in die Übersetzung der Kripke Struktur und des LTL-Ausdrucks in die aussagenlogischen Formeln $[\![M]\!]_k$ und $[\![\varphi]\!]_k^0$. Die Teilformel $[\![M]\!]_k$ gewährleistet, dass nur gültige Pfadpräfixe betrachtet werden. Zur booleschen Kodierung der Zustände von M werden zunächst n Atome $Atoms = \{A, B, ...\}$ eingeführt, so dass gilt $2^{n-1} < |S| \leq 2^n$. Zudem sei $L(Atoms)$ die Menge aller aussagenlogischen Formeln über $Atoms$ und den Konstanten $true, false$ mit den zulässigen Junktoren $\neg, \wedge, \vee$. $L^{\perp}(Atoms)$ enthalte zusätzlich $\perp$ als Konstante. Eine *Bewertung* ist eine Abbildung $I : Atoms \to \{true, false\}$, $I(F)$ liefert dementsprechend den Wahrheitswert der Formel F für I. Es sei darauf hingewiesen, dass Atome nicht mit $\perp$ bewertet werden können. Hinsichtlich der späteren Reduktion auf zwei Standard SAT-Instanzen ist $\perp$ hier nur als Konstante zugelassen.

Definition 3.3.1: Zustandskodierung
Die Zustandskodierung *einer Kripke Struktur ist eine injektive Abbildung $c : S \to L(Atoms)$, wobei $c(s)$ eine Konjunktion von Literalen ist. Somit lässt sich $c(s)$ auch als eine Bewertung für die Menge Atoms betrachten.*

Eine zulässige Zustandskodierung für die Kripke Struktur in Abbildung 3.2.1 ist beispielsweise $c(s_0) = \neg A \wedge \neg B$, $c(s_1) = \neg A \wedge B$ und $c(s_2) = A \wedge \neg B$. Zur Konstruktion von $[\![M]\!]_k$ werden Zustände nicht isoliert betrachtet, sondern als Bestandteil von Pfadpräfixen. Daher wird die Zustandskodierung um einen Indexwert $i \in \{0, ..., k\}$ ergänzt, welcher die aktuelle Präfixposition beschreibt. $c(s_1)_i = \neg A_i \wedge B_i$ bedeutet, dass an Position i der Zustand s_1 vorliegt. Als erweiterte Atommenge ergibt sich somit $Atoms = \{A_0, B_0, ..., A_k, B_k, ...\}$.

Die nachfolgende Definition von $[\![M]\!]_k$ ergänzt die ursprüngliche Kodierung aus [Weh08], indem sie auch dem Vorhandensein von $\perp$-Transitionen Rechnung trägt.

Definition 3.3.2: Kodierung einer partiellen Kripke Struktur
Sei $M = (S, S_0, R, L)$ eine partielle Kripke Struktur. $Init_0$ sei das Prädikat, welches die Startzustände beschreibt mit

$$Init_0 = \bigvee_{s \in S_0} c(s)_0$$

zudem sei $T_{i,\,i+1}$ *die Transitionskodierung mit*

$$T_{i,\,i+1} = \bigvee_{s \in S,\, s' \in S} (c(s)_i \wedge c(s')_{i+1} \wedge R(s, s'))$$

Dann gilt für ein $k \in \mathbb{N}$

$$[\![M]\!]_k := Init_0 \wedge \bigwedge_{i=0}^{k-1} T_{i,\,i+1}$$

Für die Kripke Struktur M in Abbildung 3.2.1 ergibt sich die folgende Kodierung

$$Init_0 = \neg A_0 \wedge \neg B_0$$

$$T_{i,i+1} = (\neg A_i \wedge \neg B_i \wedge \neg A_{i+1} \wedge B_{i+1}) \vee (\neg A_i \wedge B_i \wedge A_{i+1} \wedge \neg B_{i+1}) \vee (A_i \wedge \neg B_i \wedge \neg A_{i+1} \wedge \neg B_{i+1} \wedge \bot)$$

Jede Bewertung von $[\![M]\!]_k$ charakterisiert einen Pfadpräfix – unabhängig davon, ob dieser für M zulässig ist oder nicht. Für einen Präfix π sei I_π die korrespondierende Bewertung.

$[\![M]\!]_k$ ist für $k = 1$ und die Bewertung $I_\pi = \{A_0 \rightarrow false, B_0 \rightarrow false, A_1 \rightarrow false, B_1 \rightarrow true\}$ *wahr*, denn I_π beschreibt den gültigen Pfadpräfix $\pi = (s_0, s_1)$. Für $k = 2$ und $\pi = (s_0, s_1, s_2)$ ergibt sich $I_\pi = \{A_0 \rightarrow false, B_0 \rightarrow false, A_1 \rightarrow false, B_1 \rightarrow true, A_2 \rightarrow true, B_2 \rightarrow false\}$ mit $I_\pi([\![M]\!]_2) = \bot$. Der beschriebene Präfix besitzt eine $\bot$-Transition.

Aus den Definitionen 3.3.1 und 3.3.2 ergibt sich der folgende Satz.

Satz 3.3.1:
Sei M *eine partielle Kripke Struktur,* $k \in \mathbb{N}$ *und* $[\![M]\!]_k$ *die Kodierung von* M *für Bound* k*. Dann folgt aus den Definitionen 3.3.1 und 3.3.2*

- *für alle gültigen Pfade* π *von* M*, deren* k*-Präfix* keine $\bot$*-Transition besitzt, gilt*

$$I_\pi([\![M]\!]_k) = true,$$

- *alle Bewertungen* I *mit* $I([\![M]\!]_k) = true$ *beschreiben* k*-Präfixe* ohne $\bot$*-Transition von gültigen Pfaden in* M*,*

- *für alle gültigen Pfade* π *von* M*, deren* k*-Präfix* mindestens eine $\bot$*-Transition besitzt, gilt*

$$I_\pi([\![M]\!]_k) = \bot,$$

- *alle Bewertungen* I *mit* $I([\![M]\!]_k) = \bot$ *beschreiben* k*-Präfixe* mit mindestens einer $\bot$*-Transition von gültigen Pfaden in* M*,*

- *für alle Pfade π, die für M nicht zulässig sind, da bereits ihr k-Präfix für M* ungültig *ist, gilt*
$$I_\pi([\![M]\!]_k) = false,$$
- *alle Bewertungen I mit $I([\![M]\!]_k) = false$ beschreiben* keine *gültigen k-Präfixe von Pfaden in M.*

Die hier beschriebenen Eigenschaften von $[\![M]\!]_k$ unter einer Bewertung I_π stellen eine Abweichung der Kodierung aus [Weh08] dar. Es gilt offensichtlich nicht für *alle* Pfade π von M: $I_\pi([\![M]\!]_k) = true$, sondern nur für solche *ohne* $\perp$-Transition. Diese Tatsache ermöglicht eine wesentliche Vereinfachung der Kodierung von φ, wie sich im Weiteren zeigen wird.

Wie schon in Abschnitt 3.2. erläutert, sind die LTL$^+$-Semantiken beim Bounded Model Checking davon abhängig, ob der betrachtete Pfadpräfix eine k-Schleife besitzt oder nicht. Dementsprechend geschieht auch die Übersetzung der temporallogischen Formel φ zweifach. Zunächst wird die Kodierung für Pfade *ohne* k-Schleife vorgestellt.

Definition 3.3.3: LTL$^+$-Kodierung ohne Schleife
Sei φ eine LTL$^+$-Formel und $k, i \in \mathbb{N}$ mit $i \leq k$

$$\begin{aligned}
[\![p]\!]^i_k &= \textstyle\bigvee_{s\in S} c(s)_i \wedge L(s,p) \\
[\![\bar{p}]\!]^i_k &= \textstyle\bigvee_{s\in S} c(s)_i \wedge L(s,\bar{p}) \\
[\![\varphi_1 \wedge \varphi_2]\!]^i_k &= [\![\varphi_1]\!]^i_k \wedge [\![\varphi_2]\!]^i_k \\
[\![\varphi_1 \vee \varphi_2]\!]^i_k &= [\![\varphi_1]\!]^i_k \vee [\![\varphi_2]\!]^i_k \\
[\![\mathbf{G}\varphi]\!]^i_k &= false \\
[\![\mathbf{F}\varphi]\!]^i_k &= \textstyle\bigvee_{j=i}^{k} [\![\varphi]\!]^j_k \\
[\![\mathbf{X}\varphi]\!]^i_k &= \text{if } i < k \text{ then } [\![\varphi]\!]^{i+1}_k \text{ else } false \\
[\![\varphi_1 \mathbf{U} \varphi_2]\!]^i_k &= \textstyle\bigvee_{j=i}^{k} \left([\![\varphi_2]\!]^j_k \wedge \bigwedge_{n=i}^{j-1} [\![\varphi_1]\!]^n_k \right)
\end{aligned}$$

Die hier beschriebene Übersetzung weicht lediglich für atomare Propositionen von der ursprünglichen Definition aus [BCC+99] ab. Für atomare LTL$^+$-Formeln p ergibt sich als aussagenlogische Kodierung die Disjunktion aller Zustandskodierungen, jeweils verknüpft mit dem Wahrheitswert von $L(s,p)$. Aufgrund der Dreiwertigkeit der Beschriftungsfunktion kommt es in $[\![p]\!]^i_k$ und daraus zusammengesetzten komplexeren Formeln

$[\![\varphi]\!]_k^i$ zum Auftreten der Konstanten $\bot$. So gilt für die Kripke Struktur in Abbildung 3.2.1 $[\![p]\!]_2^1 = (\neg A_1 \wedge \neg B_1 \wedge true) \vee (\neg A_1 \wedge B_1 \wedge \bot) \vee (A_1 \wedge \neg B_1 \wedge false)$.

Die Kodierung von LTL$^+$-Formeln mit den Temporaloperatoren **F**, **X** und **U** berücksichtigt *nicht* die Transitionsfunktion R. Dies steht im Gegensatz zu den Bounded Semantiken aus Definition 3.2.2, wonach der Wahrheitswert von $[\pi \models_k^i \varphi]$ davon abhängig ist, ob der k-Präfix von π eine $\bot$-Transition besitzt. Nachfolgend wird gezeigt, dass die hier gewählte SAT-Kodierung dennoch ein korrektes Bounded Model Checking gewährleistet. Grundlage ist die angepasste Kodierung der partiellen Kripke Struktur M, die es zulässt, bei der Übersetzung von φ auf die Berücksichtigung der Transitionsfunktion zu verzichten.

Unter der Voraussetzung, dass alle betrachteten Präfixe *keine* k-Schleife besitzen, ergibt sich die folgende vereinfachte Kodierung eines BMC-Problems

$$[\![M, \varphi]\!]_k = [\![M]\!]_k \wedge [\![\varphi]\!]_k^0$$

Die Korrektheit der Bounded Model Checking Verfahren aus [BCC$^+$99] und [Weh08] beruht auf der Gültigkeit der folgenden Gleichung

$$[\pi \models_k^0 \varphi] = I_\pi([\![M]\!]_k \wedge [\![\varphi]\!]_k^0)$$

In Worten bedeutet dies: I_π ist eine erfüllende Bewertung für $[\![M]\!]_k \wedge [\![\varphi]\!]_k^0$ genau dann, wenn der Pfad π einen k-Präfix besitzt, der Zeuge für φ ist. Für das Verfahren aus [Weh08] gilt eine entsprechende Generalisierung dieser Aussage, da hier der zusätzliche Wahrheitswert *unbekannt* existiert.

Die Korrektheit der in *dieser* Arbeit vorgestellten BMC Technik folgt aus einer abgeschwächten Form der vorherigen Gleichung. Es wird für die hier gewählte Kodierung gezeigt

$$[\pi \models_k^0 \varphi] = \bigvee_{j=0}^{k} I_\pi([\![M]\!]_j \wedge [\![\varphi]\!]_j^0)$$

Somit ist es zur Bestimmung des Wahrheitswertes von $[\pi \models_k^0 \varphi]$ erforderlich, die aussagenlogische Kodierung für jeden Bound $j, 0 \leq j \leq k$ zu erzeugen. Dieser scheinbare Mehraufwand kommt bei einem inkrementell vorgehenden Bounded Model Checker allerdings nicht zum Tragen. BMC Implementierungen, wie beispielsweise NuSMV [CCG$^+$02] initialisieren k mit 0 und suchen dann iterativ nach einem Gegenbeispiel.

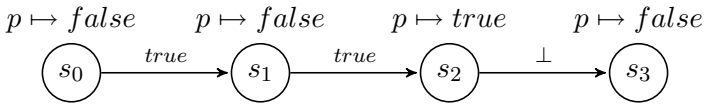

Abbildung 3.3.1: 3-Präfix eines Pfades π

Nachfolgend wird anhand eines Beispiels verdeutlicht, weshalb für die hier gewählte Kodierung lediglich die abgeschwächte Gleichung gilt. Für eine Bewertung I_π, die einen Pfad π charakterisiert, besitzt die aussagenlogische Formel $[\![M]\!]_k \wedge [\![\varphi]\!]_k^0$ im Allgemeinen nicht denselben Wahrheitswert wie $[\pi \models_k^0 \varphi]$. Exemplarisch sei hier der 3-Präfix in Abbildung 3.3.1 betrachtet. Offensichtlich gilt $[\pi \models_3^0 \mathbf{F}p] = true$, denn p ist an der Position 2 *wahr* und für alle Transitionen (s_j, s_{j+1}) mit $0 \leq j < 2$ gilt $R(s_j, s_{j+1}) = true$ – die Transition (s_2, s_3) ist für den Wert von $[\pi \models_3^0 \mathbf{F}p]$ irrelevant. Für $[\![M]\!]_3 \wedge [\![\mathbf{F}p]\!]_3^0$ ergibt sich unter der Bewertung I_π hingegen der Wahrheitswert $\perp$, da gemäß Definition 3.3.2 die Transition (s_2, s_3) mit in die Konstruktion von $[\![M]\!]_3$ einfließt. Eine Verallgemeinerung dieser Tatsache ist das folgende Lemma:

Lemma 3.3.1:
Sei π ein Pfad einer partiellen Kripke Struktur M, φ eine LTL$^+$-Formel und $k \in \mathbb{N}$. Zudem sei $[\pi \models_k^i \varphi]$ definiert gemäß den Bounded LTL$^+$-Semantiken für Pfade ohne *k-Schleife. Dann gilt*

$$[\pi \models_k^i \varphi] \geq I_\pi([\![M]\!]_k \wedge [\![\varphi]\!]_k^i)$$

Beweis: Siehe Anhang.

Ist allerdings ein inkrementelles Vorgehen bezüglich k vorausgesetzt, so ergibt sich bereits für $k = 2$: $I_\pi([\![M]\!]_2 \wedge [\![\mathbf{F}p]\!]_2^0) = true$, womit ein Zeuge für $\mathbf{F}p$ gefunden ist. Es existiert also ein kürzerer Präfix, der die Zeugeneigenschaft erfüllt und ausschließlich Transitionen mit Wahrheitswert *true* besitzt.

Lemma 3.3.2:
Sei π ein Pfad einer partiellen Kripke Struktur M, φ eine LTL$^+$-Formel, $v \in \{true, \perp, false\}$ und $k \in \mathbb{N}$. Zudem sei $[\pi \models_k^i \varphi]$ definiert gemäß den Bounded LTL$^+$-Semantiken für Pfade ohne *k-Schleife. Dann gilt*

$$[\pi \models_k^i \varphi] \geq v \quad \Rightarrow \quad \exists j, i \leq j \leq k : [\pi \models_j^i \varphi] \geq v \wedge \forall n, i \leq n < j : R(\pi(n), \pi(n+1)) \geq v$$

Beweis: Siehe Anhang.

Eine weitere Besonderheit der gewählten Kodierung betrifft das Verhältnis von $[\pi \models_k^i \varphi]$ und $I_\pi([\![\varphi]\!]_k^i)$. Für den Präfix in Abbildung 3.3.1 gilt aufgrund der $\bot$-Transition (s_2, s_3) $[\pi \models_3^0 \mathbf{F}(p\mathbf{U}\bar{p})] = \bot$. Wie bereits erwähnt, wird bei der LTL$^+$-Kodierung die Transitionsfunktion nicht berücksichtigt, somit gilt $I_\pi([\![\mathbf{F}(p\mathbf{U}\bar{p})]\!]_3^0) = true$.

Lemma 3.3.3:
Sei π ein Pfad einer partiellen Kripke Struktur M, φ eine LTL$^+$-Formel und $k \in \mathbb{N}$. Zudem sei $[\pi \models_k^i \varphi]$ definiert gemäß den Bounded LTL$^+$-Semantiken für Pfade ohne *k-Schleife. Dann gilt*

$$[\pi \models_k^i \varphi] \leq I_\pi([\![\varphi]\!]_k^i)$$

Beweis: Siehe Anhang.

Die zuvor aufgestellten Lemmata bilden die Grundlage für die Gültigkeit des nachfolgenden Theorems.

Theorem 3.3.1:
Sei π ein Pfad einer partiellen Kripke Struktur M, φ eine LTL$^+$-Formel und $k \in \mathbb{N}$. Zudem sei $[\pi \models_k^i \varphi]$ definiert gemäß den Bounded LTL$^+$-Semantiken für Pfade ohne *k-Schleife. Dann gilt*

$$[\pi \models_k^0 \varphi] = \bigvee_{j=0}^{k} I_\pi([\![M]\!]_j \wedge [\![\varphi]\!]_j^0)$$

Beweis:
Sei $[\pi \models_k^0 \varphi] = v$ mit $v \in \{true, \bot, false\}$. Dann existiert gemäß Lemma 3.3.2 ein $j, 0 \leq j \leq k$ mit

a) $[\pi \models_j^0 \varphi] \geq v$

b) $\forall n, 0 \leq n < j : R(\pi(n), \pi(n+1)) \geq v$

Als direkte Konsequenz aus b) und der Definition 3.3.2 (Kodierung einer partiellen Kripke Struktur) ergibt sich

c) $I_\pi([\![M]\!]_j) \geq v$

Nach Lemma 3.2.1 gilt $\forall k \in \mathbb{N} : [\pi \models_k^i \varphi] \leq [\pi \models_{k+1}^i \varphi]$ und somit für das betrachtete $j \leq k$ auch $[\pi \models_j^0 \varphi] \leq [\pi \models_k^0 \varphi]$. Aus a) folgt dementsprechend

d) $[\pi \models_k^0 \varphi] = [\pi \models_j^0 \varphi] = v$

e) $\forall n, 0 \leq n < k : [\pi \models_n^0 \varphi] \leq [\pi \models_k^0 \varphi]$

Zudem gilt $[\pi \models_j^0 \varphi] \leq I_\pi([\![\varphi]\!]_j^0)$ (Lemma 3.3.3) und $[\pi \models_j^0 \varphi] \geq I_\pi([\![M]\!]_j \wedge [\![\varphi]\!]_j^0)$ (Lemma 3.3.1). Unter der Berücksichtigung von d) ergibt sich

f) $I_\pi([\![\varphi]\!]_j^0) \geq v$

g) $I_\pi([\![M]\!]_j \wedge [\![\varphi]\!]_j^0) \leq v$

Es folgt aus c), d), f) und g)

$$[\pi \models_k^0 \varphi] = I_\pi([\![M]\!]_j \wedge [\![\varphi]\!]_j^0)$$

und gemäß e) und Lemma 3.3.1 auch

$$[\pi \models_k^0 \varphi] = \bigvee_{j=0}^{k} I_\pi([\![M]\!]_j \wedge [\![\varphi]\!]_j^0)$$

□

Damit ist gezeigt, dass die SAT-Übersetzung gemäß den Definitionen 3.3.2 und 3.3.3 ein korrektes Bounded Model Checking ermöglicht, sofern inkrementell bezüglich k vorgegangen wird. Bisher ist dies jedoch mit der Beschränkung auf Pfade *ohne* Schleife verbunden. Im Weiteren wird die vorgestellte Kodierung um den Fall ergänzt, dass der betrachtete Pfad eine k-Schleife besitzt.

Definition 3.3.4: Nachfolger innerhalb einer Schleife
Seien $k, l, i \in \mathbb{N}$ mit $l, i \leq k$. Dann ist der Nachfolger *$succ(i)$ von i in einer (k,l)-Schleife wie folgt definiert: $succ(i) := i + 1$ für $i < k$ und $succ(i) := l$ für $i = k$.*

Definition 3.3.5: LTL$^+$-Kodierung mit Schleife
Sei φ eine LTL$^+$-Formel und $k, i, l \in \mathbb{N}$ mit $i, l \leq k$

$$
\begin{aligned}
{}_l[\![p]\!]_k^i &= \textstyle\bigvee_{s \in S} c(s)_i \wedge L(s,p) \\
{}_l[\![\bar{p}]\!]_k^i &= \textstyle\bigvee_{s \in S} c(s)_i \wedge L(s,\bar{p}) \\
{}_l[\![\varphi_1 \wedge \varphi_2]\!]_k^i &= {}_l[\![\varphi_1]\!]_k^i \wedge_l [\![\varphi_2]\!]_k^i \\
{}_l[\![\varphi_1 \vee \varphi_2]\!]_k^i &= {}_l[\![\varphi_1]\!]_k^i \vee_l [\![\varphi_2]\!]_k^i \\
{}_l[\![\mathbf{G}\varphi]\!]_k^i &= \textstyle\bigwedge_{j=min(i,l)}^{k} {}_l[\![\varphi]\!]_k^j \\
{}_l[\![\mathbf{F}\varphi]\!]_k^i &= \textstyle\bigvee_{j=min(i,l)}^{k} {}_l[\![\varphi]\!]_k^j
\end{aligned}
$$

$$
\begin{aligned}
{}_l[\![\mathbf{X}\varphi]\!]_k^i &= {}_l[\![\varphi]\!]_k^{succ(i)} \\
{}_l[\![\varphi_1 \mathbf{U} \varphi_2]\!]_k^i &= \bigvee_{j=i}^{k} \left({}_l[\![\varphi_2]\!]_k^j \wedge \bigwedge_{n=i}^{j-1} {}_l[\![\varphi_1]\!]_k^n \right) \vee \bigvee_{j=l}^{i-1} \left({}_l[\![\varphi_2]\!]_k^j \wedge \bigwedge_{n=i}^{k} {}_l[\![\varphi_1]\!]_k^n \wedge \bigwedge_{n=l}^{j-1} {}_l[\![\varphi_1]\!]_k^n \right)
\end{aligned}
$$

Auch diese Übersetzung einer LTL$^+$-Formel in einen aussagenlogischen Ausdruck ist nicht von der Transitionsfunktion R abhängig. Es ist hinreichend, R bei der Konstruktion von $[\![M]\!]_k$ zu berücksichtigen.

Für die vollständige Kodierung eines BMC-Problems muss zunächst festgestellt werden, ob eine k-Schleife vorliegt und falls ja, für welches l. Zudem sind die Fälle $R(\pi(k), \pi(l)) = \mathit{true}$ und $R(\pi(k), \pi(l)) = \bot$ zu differenzieren. Dazu wird eine Schleifenbedingung verwendet.

Definition 3.3.6: Schleifenbedingung
Sei $k, l \in \mathbb{N}$ mit $l \leq k$ und sei ${}_lL_k = T_{k,l}$. Dann ist die L_k wie folgt definiert:

$$L_k := \bigvee_{l=0}^{k} {}_lL_k$$

Für den Wahrheitswert der Schleifenbedingung bei einer Bewertung I_π gilt somit

$$I_\pi(L_k) = \begin{cases} \mathit{true} & \mathit{falls}\ \exists l\ : R(\pi(k), \pi(l)) = \mathit{true} \\ \mathit{false} & \mathit{falls}\ \neg\exists l\ : R(\pi(k), \pi(l)) = \mathit{true} \vee R(\pi(k), \pi(l)) = \bot \\ \bot & \mathit{sonst} \end{cases}$$

Die Gesamtkodierung eines Bounded Model Checking Problems verbindet die Übersetzungen von Kripke Struktur und LTL$^+$-Formel. Anstelle der ursprünglichen Kodierung aus [BCC+99] wird hier die optimierte Variante aus [CPR+02] verwendet. Demnach kann auf die Konjunktion von $[\![\varphi]\!]_k^0$ mit der negierten Schleifenbedingung verzichtet werden, denn für alle $\varphi \in LTL^+$ und $i, l, k \in \mathbb{N}$ gilt die Implikation $[\![\varphi]\!]_k^0 \models {}_l[\![\varphi]\!]_k^0$. Somit ergibt sich eine platzeffizientere aussagenlogische Formel.

$$[\![M, \varphi]\!]_k := [\![M]\!]_k \wedge \left[[\![\varphi]\!]_k^0 \vee \bigvee_{l=0}^{k} \left({}_lL_k \wedge_l [\![\varphi]\!]_k^0 \right) \right]$$

Die Korrektheit dieser vollständigen SAT-Reduktion eines BMC Problems ergibt sich aus der Gültigkeit des nachfolgenden Theorems.

Theorem 3.3.2:
Sei π ein Pfad einer partiellen Kripke Struktur M, φ eine LTL$^+$-Formel und $k \in \mathbb{N}$. Dann gilt

$$[\pi \models_k^0 \varphi] = \bigvee_{j=0}^{k} I_\pi(\llbracket M, \varphi \rrbracket_j)$$

Beweis:
Der Beweis argumentiert anhand einer zu $\bigvee_{j=0}^{k} \llbracket M, \varphi \rrbracket_j$ äquivalenten Formel.

$$\begin{aligned}
\bigvee_{j=0}^{k} \llbracket M, \varphi \rrbracket_j &= \bigvee_{j=0}^{k} (\llbracket M \rrbracket_j \wedge \left[\llbracket \varphi \rrbracket_j^0 \vee \bigvee_{l=0}^{j} \left({}_l L_j \wedge_l \llbracket \varphi \rrbracket_j^0 \right) \right]) \\
&= \bigvee_{j=0}^{k} (\llbracket M \rrbracket_j \wedge \llbracket \varphi \rrbracket_j^0) \vee \bigvee_{j=0}^{k} \bigvee_{l=0}^{j} ({}_l L_j \wedge \llbracket M \rrbracket_j \wedge_l \llbracket \varphi \rrbracket_j^0) \\
&\overset{Korollar 3.2.1}{=} \bigvee_{j=0}^{k} (\llbracket M \rrbracket_j \wedge \llbracket \varphi \rrbracket_j^0) \vee \bigvee_{l=0}^{k} ({}_l L_k \wedge \llbracket M \rrbracket_k \wedge_l \llbracket \varphi \rrbracket_k^0)
\end{aligned}$$

Nun wird gezeigt:

$$[\pi \models_k^0 \varphi] = \bigvee_{j=0}^{k} I_\pi(\llbracket M \rrbracket_j \wedge \llbracket \varphi \rrbracket_j^0) \vee \bigvee_{l=0}^{k} I_\pi({}_l L_k \wedge \llbracket M \rrbracket_k \wedge_l \llbracket \varphi \rrbracket_k^0)$$

Nach Theorem 3.3.1 gilt $[\pi \models_k^0 \varphi] = \bigvee_{j=0}^{k} I_\pi(\llbracket M \rrbracket_j \wedge \llbracket \varphi \rrbracket_j^0)$, sofern $[\pi \models_k^0 \varphi]$ gemäß den Bounded LTL$^+$-Semantiken für Pfade *ohne* k-Schleife definiert ist. Durch die Aufhebung dieser Restriktion gilt folglich $[\pi \models_k^0 \varphi] \geq \bigvee_{j=0}^{k} I_\pi(\llbracket M \rrbracket_j \wedge \llbracket \varphi \rrbracket_j^0)$. Somit sind zwei Fälle zu unterscheiden.

1. Es gelte $[\pi \models_k^0 \varphi] = \bigvee_{j=0}^{k} I_\pi(\llbracket M \rrbracket_j \wedge \llbracket \varphi \rrbracket_j^0)$.
   So bleibt noch zu zeigen $[\pi \models_k^0 \varphi] \geq \bigvee_{l=0}^{k} I_\pi({}_l L_k \wedge \llbracket M \rrbracket_k \wedge_l \llbracket \varphi \rrbracket_k^0)$.

   Lemma 3.3.4:
   Sei π ein Pfad einer partiellen Kripke Struktur M, φ eine LTL$^+$-Formel und $k \in \mathbb{N}$. Dann gilt

   $$[\pi \models_k^i \varphi] \geq \bigvee_{l=0}^{k} I_\pi({}_l L_k \wedge \llbracket M \rrbracket_k \wedge_l \llbracket \varphi \rrbracket_k^i)$$

   Beweis: Siehe Anhang.

   Folglich gilt auch $[\pi \models_k^0 \varphi] = \bigvee_{j=0}^{k} I_\pi(\llbracket M \rrbracket_j \wedge \llbracket \varphi \rrbracket_j^0) \vee \bigvee_{l=0}^{k} I_\pi({}_l L_k \wedge \llbracket M \rrbracket_k \wedge_l \llbracket \varphi \rrbracket_k^0)$.

2. Es gelte $[\pi \models_k^0 \varphi] > \bigvee_{j=0}^{k} I_\pi([\![M]\!]_j \wedge [\![\varphi]\!]_j^0)$.
   Sei $[\pi \models_k^0 \varphi] = v$ mit $v \in \{true, \bot\}$, so gilt $\bigvee_{j=0}^{k} I_\pi([\![M]\!]_j \wedge [\![\varphi]\!]_j^0) < v$. Offensichtlich ist keine endliche Zustandssequenz $(s_0, ..., s_j)$ von π mit $j \leq k$ ein Zeuge mit Wahrheitswert v für φ. Somit muss der k-Präfix von π einen unendlichen Pfad repräsentieren, der Zeuge für φ mit Wert v ist. Daraus folgt

   a) $I_\pi([\![M]\!]_k) \geq v$

   b) $\exists l, 0 \leq l \leq k : I_\pi({}_lL_k) \geq v$

   c) Sei $I_\pi({}_lL_k) \geq v$ dann gilt $\forall l', l' \neq l : I_\pi({}_{l'}L_k) = false$

   Aus Lemma 3.3.4 geht die Gültigkeit der Ungleichung $[\pi \models_k^0 \varphi] \geq \bigvee_{l=0}^{k} I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\varphi]\!]_k^0)$ hervor. Es bleibt zu zeigen, dass in dem hier betrachteten Fall *echte* Gleichheit gilt. Dazu ist es unter der Berücksichtigung der Fakten a), b) und c) sowie des Lemmas 3.3.4 hinreichend, die Gültigkeit von $[\pi \models_k^0 \varphi] \leq I_\pi({}_l[\![\varphi]\!]_k^0)$ für ein π mit (k,l)-Schleife zu beweisen.

   Lemma 3.3.5:
   Sei π ein Pfad einer partiellen Kripke Struktur M und φ eine LTL$^+$-Formel. Zudem seien $l, k \in \mathbb{N}$ mit $l \leq k$ und π besitzt eine (k,l)-Schleife. Dann gilt

$$[\pi \models_k^i \varphi] \leq I_\pi({}_l[\![\varphi]\!]_k^i)$$

   Beweis: Siehe Anhang.

   Somit gilt auch $[\pi \models_k^0 \varphi] = \bigvee_{j=0}^{k} I_\pi([\![M]\!]_j \wedge [\![\varphi]\!]_j^0) \vee \bigvee_{l=0}^{k} I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\varphi]\!]_k^0)$.

□

Korollar 3.3.1:
Sei M eine partielle Kripke Struktur, φ eine LTL$^+$-Formel und $k \in \mathbb{N}$.

$$[M \models_{E,k} \varphi] \quad = \quad \bigvee_{\pi,\ \pi\ Pfad\ von\ M} \ \bigvee_{j=0}^{k} I_\pi([\![M, \varphi]\!]_j)$$

Es liegt also mit $\bigvee_{j=0}^{k} [\![M, \varphi]\!]_j$ eine aussagenlogische Formel vor, die genau dann erfüllbar ist, wenn das korrespondierende Bounded Model Checking Problem für eine partielle Kripke Struktur M den Wahrheitswert *true* besitzt. Ferner ist die Erfüllbarkeit von

$\bigvee_{j=0}^{k} [\![M, \varphi]\!]_j$ gemäß der kleeneschen Logik *unbekannt*, gdw. gilt $[M \models_{E,k} \varphi] = \bot$. Damit sind die Voraussetzungen gegeben, BMC-Probleme durch einen Erfüllbarkeitstest zu lösen. Bei der erzeugten SAT-Kodierung handelt es sich um eine zweiwertige aussagenlogische Formel mit zusätzlicher Konstanten $\bot$. Für diese Formelklasse findet sich in [Weh08] die Definition eines entsprechenden Erfüllbarkeitsproblems.

Definition 3.3.7: *sat_3*
Sei $F \in L^{\bot}(Atoms)$ eine aussagenlogische Formel.

$$[Fsat_3] = \begin{cases} true & falls\ \exists I : I(F) = true \\ false & falls\ \forall I : I(F) = false \\ \bot & sonst \end{cases}$$

Da sich jeder Pfad π einer partiellen Kripke Struktur M durch eine Bewertung I beschreiben lässt, gilt somit auch das folgende Korollar.

Korollar 3.3.1:
Sei M eine partielle Kripke Struktur, φ eine LTL$^+$-Formel und $k \in \mathbb{N}$.

$$[M \models_{E,k} \varphi] = \bigvee_{j=0}^{k} [[\![M, \varphi]\!]_j sat_3]$$

Die Gültigkeit dieser Gleichung ermöglicht die Realisierung eines korrekten Bounded Model Checking Verfahrens für Kripke Strukturen mit partieller Beschriftungs- und Transitionsfunktion. Bei einer inkrementellen Vorgehensweise bezüglich k entsteht zudem kein zusätzlicher Mehraufwand gegenüber der Technik aus [Weh08]:

input M, φ, k (upper bound)
for $j = 0$ to k **do**
  if $[[\![M, \varphi]\!]_j sat_3] \neq false$ **then**
    return *counterexample*
  end if
end for
return "no counterexample of length $\leq k$"

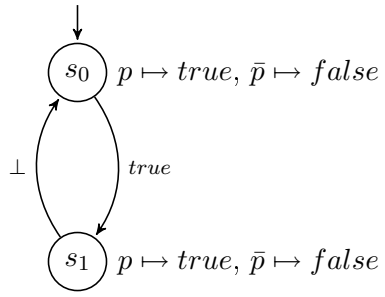

Abbildung 3.3.2: Partielle Kripke Struktur

Zur Veranschaulichung wird nachfolgend die SAT-Reduktion eines konkreten BMC-Problems gezeigt. Für die Kripke Struktur in Abbildung 3.3.2, die LTL$^+$-Formel $\mathbf{G}p$ und den Bound $k = 1$ ergibt sich als Kodierung

$$
\begin{aligned}
[\![M]\!]_1 &= (\neg A_0 \wedge \neg B_0) \wedge ((\neg A_0 \wedge \neg B_0 \wedge \neg A_1 \wedge B_1) \vee (\neg A_0 \wedge B_0 \wedge \neg A_1 \wedge \neg B_1 \wedge \bot)) \\
[\![\mathbf{G}p]\!]_1^0 &= false \\
{}_0[\![\mathbf{G}p]\!]_1^0 &= ((\neg A_0 \wedge \neg B_0) \vee (\neg A_0 \wedge B_0)) \wedge ((\neg A_1 \wedge \neg B_1) \vee (\neg A_1 \wedge B_1)) \\
{}_1[\![\mathbf{G}p]\!]_1^0 &= ((\neg A_0 \wedge \neg B_0) \vee (\neg A_0 \wedge B_0)) \wedge ((\neg A_1 \wedge \neg B_1) \vee (\neg A_1 \wedge B_1)) \\
{}_0L_1 &= (\neg A_1 \wedge \neg B_1 \wedge \neg A_0 \wedge B_0) \vee (\neg A_1 \wedge B_1 \wedge \neg A_0 \wedge \neg B_0 \wedge \bot) \\
{}_1L_1 &= (\neg A_1 \wedge \neg B_1 \wedge \neg A_1 \wedge B_1) \vee (\neg A_1 \wedge B_1 \wedge \neg A_1 \wedge \neg B_1 \wedge \bot)
\end{aligned}
$$

Es existiert keine Bewertung welche die Formel $[\![M, \mathbf{G}p]\!]_1 = [\![M]\!]_1 \wedge [[\![\mathbf{G}p]\!]_1^0 \vee \bigvee_{l=0}^{1} ({}_lL_1 \wedge {}_l[\![\mathbf{G}p]\!]_1^0)]$ *wahr* macht. Für $I = \{A_0 \rightarrow false, B_0 \rightarrow false, A_1 \rightarrow false, B_1 \rightarrow true\}$ ergibt sich jedoch der Wahrheitswert $\bot$. I beschreibt den Präfix (s_0, s_1) der über eine 1-Schleife verfügt und somit einen unendlichen Pfad $\pi = (s_0, s_1, s_0, s_1, ...)$ repräsentiert. π ist ein Zeuge für $\mathbf{G}p$, besitzt jedoch eine $\bot$-Transition, folglich gilt auch $[\pi \models_1^0 \mathbf{G}p] = \bot$.

3.4. Erfüllbarkeitstest

Im vorherigen Abschnitt wurde die Reduktion des BMC-Problems für partielle Kripke Strukturen auf das aussagenlogische Erfüllbarkeitsproblem sat_3 gezeigt. Die praktische Durchführung des Bounded Model Checking erfordert also abschließend einen Erfüllbarkeitstest. sat_3-Probleme lassen sich aufgrund der Eigenschaften der Formelklasse $L^{\bot}(Atoms)$ nicht mit klassischen mehrwertigen SAT-Solvern wie CAMA [LKM03] lösen. In [Weh08] wird ein Erfüllbarkeitstest vorgestellt, welcher auf der Reduktion von

sat_3 auf zwei klassische SAT-Instanzen basiert und somit die jüngsten Fortschritte im Bereich der SAT-Solver Technologie ausnutzt [PBG05]. Grundidee ist es, eine pessimistische und eine optimistische Vervollständigung der dreiwertigen aussagenlogischen Formel $F = [\![M, \varphi]\!]_k$ zu erzeugen. Die pessimistische Vervollständigung ergibt sich aus der Ersetzung aller Vorkommen von $\bot$ durch $false$ ($F^p = F[\bot/false]$), bei der optimistischen Vervollständigung wird $\bot$ durch $true$ substituiert ($F^o = F[\bot/true]$). Das nachfolgende Theorem aus [Weh08] verdeutlicht, wie sich ein sat_3-Problem durch zwei herkömmliche Erfüllbarkeitstests lösen lässt. Es sei darauf hingewiesen, dass diese Form der Vervollständigung nur zulässig ist, da $[\![M, \varphi]\!]_k$ ein existentielles Model Checking Problem beschreibt. Bei der Betrachtung universeller Probleme ist eine differenziertere Ersetzung erforderlich (Vergleiche Theorem 3.1.1).

Theorem 3.4.1:
Sei $F \in L^{\bot}(Atoms)$ eine aussagenlogische Formel, wobei das Negationszeichen nur vor Atomen auftritt. Dann gilt

$$[Fsat_3] = \begin{cases} true & falls\ F^p \in SAT \\ false & falls\ F^o \notin SAT \\ \bot & sonst \end{cases}$$

Beweis: Siehe [Weh08].

Dieses Theorem gilt unter der Voraussetzung, dass die Konstante $\bot$ in F nicht negiert vorkommt. Diese Anforderung ist für $[\![M, \varphi p]\!]_k$ erfüllt, wie sich aus den Definitionen von $[\![M]\!]_k$, $[\![\varphi]\!]_k^i$ sowie der Schleifenbedingung ergibt. Damit liegt eine korrekte Technik vor, Bounded Model Checking für Kripke Strukturen mit partieller Beschriftungs- und Transitionsfunktion, unter der Verwendung von Standard SAT-Solvern (z. B. MiniSat [ES05] oder Chaff [MMZ+01]) durchzuführen.

Kapitel 4.

Umsetzung

Im vorherigen Kapitel wurde eine Bounded Model Checking Technik für Kripke Strukturen mit partieller Beschriftungs- und Transitionsfunktion konzeptionell vorgestellt. Für die praktische Umsetzung ist allerdings eine Verfeinerung und Konkretisierung dieser Technik erforderlich.

Dies betrifft zum Einen das Format des Modells. Bei einer Kripke Struktur handelt es sich um ein Konstrukt, welches den Zustandsraum des modellierten Systems explizit beschreibt. Die Verwendung symbolischer Systembeschreibungen ermöglicht hingegen eine wesentlich platzeffizientere, da implizite Darstellung des Zustandsraumes.

Des Weiteren ist die Effizienz des BMC von der Größe und Klasse der generierten aussagenlogischen Formel abhängig. Etablierte SAT-Solver akzeptieren lediglich Formeln in konjunktiver Normalform (KNF) als Eingabe. Die KNF-Konvertierung kann jedoch zu einer exponentiellen Steigerung der Formellänge führen.

Im Folgenden werden daher die einzelnen Schritte der Bounded Model Checking Technik näher betrachtet und effiziente Umsetzungsverfahren vorgestellt. Abschließend erfolgt eine Beschreibung des Bounded Model Checkers für partielle Systembeschreibungen, der im Rahmen dieser Arbeit implementiert wurde. Dieser Model Checker setzt das Konzept aus Kapitel 3. um und berücksichtigt zudem die nachfolgend beschriebenen Verfeinerungen der BMC-Technik.

4.1. Beschreibung von Startzuständen und Transitionen durch mehrwertige Entscheidungsdiagramme

Im Bereich des Model Checking lassen sich zwei grundlegende Klassen von Verfahren differenzieren. Explizite Verfahren überprüfen die Gültigkeit der Spezifikation direkt anhand von Pfaden einer Kripke Struktur. Symbolische Model Checker verwenden hingegen eine implizite und dadurch kompaktere Repräsentation der Kripke Struktur. Üblicherweise wird dazu eine boolesche Kodierung des Zustandsraumes in Form von binären Entscheidungsdiagrammen (BDDs) [BCM+92] erzeugt.

Mehrwertige Entscheidungsdiagramme (englisch: *Multi-valued Decision Diagrams, MDDs*) [CDE01] sind eine Generalisierung von BDDs. Sie ermöglichen die Darstellung von Funk-

tionen einer höherwertigen Logik. Damit eigenen sich diese Datenstrukturen insbesondere für die Repräsentation von Zustandsräumen mit unbekannten Eigenschaften. So finden MDDs auch Anwendung in 3Spot [SWW09], der Implementierung eines Abstraktionsverfahrens für parallele Systeme. 3Spot erzeugt ein abstraktes Modell eines eingelesenen C-Programmes und übersetzt dieses in mehrwertige Entscheidungsdiagramme. Zum Auftreten unbekannter Eigenschaften kommt es hierbei durch eine mögliche Abhängigkeit des C-Programms zu parallel laufenden Prozessen, die in der Abstraktion nicht berücksichtigt wurden. Abschließend werden die MDDs sowie eine temporallogische Spezifikation an den mehrwertigen Model Checker χChek [CDE+03] übergeben.

Der im Rahmen dieser Arbeit entwickelte Bounded Model Checker wurde als Alternative zu χChek in 3Spot integriert. Im Folgenden wird daher erläutert, wie sich Bounded Model Checking umsetzen lässt, wenn die partielle Kripke Struktur implizit durch MDDs gegeben ist. Zustände sind hierbei nicht fest benannt, sie definieren sich durch eine Menge von atomaren Propositionen mit zugeordneten Wahrheitswerten.

Definition 4.1.1: Zustand
Sei $AP = \{p_1, ..., p_n\}$ *eine geordnete Menge atomarer Propositionen. Dann ist ein* Zustand s *über* AP *definiert durch eine Funktion* $f_s : AP \to \{true, \bot, false\}$. f_s *beschreibt, welchen Wahrheitswert die Elemente aus* AP *in* s *besitzen. Zur Darstellung eines Zustands wird ein Vektor* $s = (f_s(p_1), ..., f_s(p_n))$ *verwendet.*

Für die partielle Kripke Struktur in Abbildung 4.1.1 ergibt sich als Zustandsmenge

$$S = \{s_0, s_1\} = \{(\bot, true), (true, false)\}$$

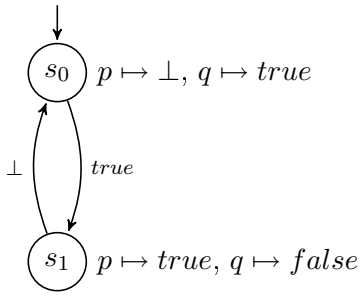

Abbildung 4.1.1: Partielle Kripke Struktur

Definition 4.1.2: Mehrwertiges Entscheidungsdiagramm
Sei D eine Menge von Wahrheitswerten und A eine Atommenge mit Ordnungsrelation $\prec$. Ein mehrwertiges Entscheidungsdiagramm *über D und A ist ein Tupel $(V, E, var, child, value)$, wobei gilt*

- *$V = V_t \cup V_n$ ist eine Menge von terminalen (V_t) und nicht-terminalen Knoten (V_n),*
- *$E \subseteq V \times V$ ist eine Menge gerichteter Kanten,*
- *$var : V_n \to A$ ist eine Beschriftungsfunktion für nicht-terminale Knoten,*
- *$child : V_n \to D \to V$ ist eine indizierte Nachfolgerfunktion für Knoten aus V_n,*
- *$value : V_t \to D$ ist eine totale Funktion, die Terminalknoten einen Wahrheitswert zuweist.*

Zudem besitzt jedes mehrwertige Entscheidungsdiagramm genau einen Wurzelknoten und auf allen Pfaden gilt die Ordnung $\prec$.

Für MDDs zur Repräsentation einer partiellen Kripke Struktur $M = (S, S_0, R, L)$ über $AP = \{p_1, ..., p_n\}$ gilt $D = \{true, \bot, false\}$ und $A = AP \cup AP'$ mit $AP' = \{p'_1, ..., p'_n\}$. AP' ist die Menge atomarer Propositionen über der die Zustände s' aller Transitionen (s, s') von M definiert sind. Ferner gilt für A die Ordnung $p_1 \prec ... \prec p_n \prec p'_1 \prec ... \prec p'_n$.

Das Entscheidungsdiagramm zur Darstellung der Transitionsfunktion sei mit M_{Trans} bezeichnet. Jeder Pfad von der Wurzel zu einem Blatt charakterisiert ein Zustandstupel (s, s') von M, wobei der Blattknoten mit dem Wahrheitswert $R(s, s')$ beschriftet ist. M_{Init} sei das MDD zur Beschreibung der Startzustände von M. Für jeden Zustand $s_0 \in S_0$ besitzt der korrespondierende Pfad einen Blattknoten mit Beschriftung *true*.

Wie auch bei binären Entscheidungsdiagrammen existieren für MDDs Reduktionstechniken, mit denen sich isomorphe Teilgraphen zusammenfassen lassen. Die Abbildungen 4.1.2 (a) und (b) zeigen die Entscheidungsdiagramme M_{Init} und M_{Trans} zur Repräsentation der expliziten Kripke Struktur in Abbildung 4.1.1. Der Übersichtlichkeit halber wird auf die Darstellung von Pfaden, deren Blatt mit *false* beschriftet ist, verzichtet.

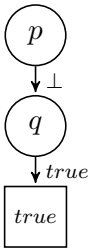

Abbildung 4.1.2 (a): MDD M_{Init}

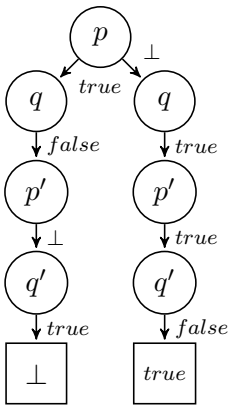

Abbildung 4.1.2 (b): MDD M_{Trans}

Nachfolgend wird die Konstruktion der aussagenlogischen Kodierung eines Bounded Model Checking Problems anhand von MDDs beschrieben. Ist also eine partielle Kripke Struktur M über AP implizit durch M_{Init} und M_{Trans} gegeben, so gilt es zunächst, für einen Bound k die Formel $[\![M]\!]_k$ zu erzeugen. Im Gegensatz zu den Annahmen im konzeptionellen Teil dieser Arbeit, liegen die Zustände von M nicht explizit vor, sondern sind über der Menge atomarer Propositionen definiert. Somit lässt sich die Zustandskodierung aus Abschnitt 3.3. hier nicht anwenden. Stattdessen orientiert sich die Kodierung an den Elementen aus AP. Für jede atomare Proposition und jede Präfixposition werden *zwei* boolesche Atome eingeführt. Dies ist notwendig, um einen Ausdruck der dreiwertigen Logik durch einen booleschen Ausdruck zu repräsentieren. Daraus ergibt sich die Menge $Atoms = \{p_A^i,\ p_B^i \mid p \in AP,\ 0 \leq i \leq k\}$. Die Formelmengen $L(Atoms)$ und $L^{\perp}(Atoms)$ sowie Bewertungen I von Formeln seien definiert wie bisher (Vergleiche Abschnitt 3.3.). Die folgende Kodierungsfunktion bildet die Grundlage zur Konstruktion von $[\![M]\!]_k$.

Definition 4.1.3: Kodierungsfunktion
Sei M eine partielle Kripke Struktur über AP, gegeben durch M_{Init} und M_{Trans} und sei $k \in \mathbb{N}$. Die Kodierung des Wahrheitswertes einer atomaren Proposition p an einer Präfixposition i wird durch die Funktion $c : AP \times \{0, ..., k\} \times \{true, \perp, false\} \to L(Atoms)$ beschrieben. c ist wie folgt definiert

$$\forall p \in AP, \forall i : 0 \leq i \leq k, \forall val \in \{true, \perp, false\} :$$

$$c(p, i, val) = \begin{cases} p_A^i & falls\ val = true \\ \neg p_A^i \wedge p_B^i & falls\ val = \perp \\ \neg p_A^i \wedge \neg p_B^i & falls\ val = false \end{cases}$$

$c(p, i, val)$ ist also die boolesche Kodierung der Aussage „Proposition p besitzt den Wahrheitswert val an der Präfixposition i“. Da der Wertebereich von val *drei* Elemente umfasst, hier aber eine zweiwertige Formel erzeugt werden soll – $\perp$ ist lediglich als Konstante zulässig – ist die Einführung von *zwei* booleschen Atomen für jede Proposition und jede Präfixposition notwendig.

Intuitiv formuliert, beschreibt jeder Pfad eines MDDs eine Konjunktion von Literalen. Für jeden nicht-terminalen Knoten v ergeben sich die Literale, repräsentiert durch $c(p, i, val)$, wobei p die Beschriftung von v ist, i die aktuelle Präfixposition und val der Wahrheitswert der Ausgangskante von v entlang des Pfades. Die Gesamtformel, die durch ein MDD dargestellt wird, entspricht der Disjunktion aller Pfadkodierungen. Für das Entscheidungsdiagramm M_{Trans} in Abbildung 4.1.2 (b) und die Position i ergibt sich die folgende aussagenlogische Formel.

$$T_{i,i+1} = (\neg p_A^i \wedge p_B^i \wedge q_A^i \wedge p_A^{i+1} \wedge \neg q_A^{i+1} \wedge \neg q_B^{i+1} \wedge true) \vee (p_A^i \wedge \neg q_A^i \wedge \neg q_B^i \wedge \neg p_A^{i+1} \wedge p_B^{i+1} \wedge q_A^{i+1} \wedge \perp)$$

Die praktische Umsetzung einer Bounded Model Checking Technik basierend auf mehrwertigen Entscheidungsdiagrammen erfordert ein gut algorithmisierbares Konvertierungsverfahren, um aus einem MDD die entsprechende aussagenlogische Formel zu erzeugen. In [She06] findet sich ein solches Verfahren für binäre Entscheidungsdiagramme, also für den zweiwertigen Fall.

Definition 4.1.4:
Sei B ein binäres Entscheidungsdiagramm. Zudem sei v der Wurzelknoten von B und es gelte $var(v) = p$, sofern B nicht konstant. Dann ist $t(B)$ die aussagenlogische Repräsentation von B mit

$$t(B) = \begin{cases} value(v) & falls\ B\ konstant \\ (p \rightarrow t(B|_{p \leftarrow true})) \wedge (\neg p \rightarrow t(B|_{p \leftarrow false})) & sonst \end{cases}$$

Unter Berücksichtigung der zuvor eingeführten Kodierungsfunktion c lässt sich t auch auf den dreiwertigen Fall erweitern, so dass sich die aussagenlogische Kodierung einer partiellen Kripke Struktur ergibt.

Definition 4.1.5: Kodierung einer partiellen Kripke Struktur anhand von MDDs
Sei M_{Init} das mehrwertige Entscheidungsdiagramm zur Repräsentation der Startzustände einer partiellen Kripke Struktur M. Zudem sei v der Wurzelknoten von M_{Init} und es gelte $var(v) = p$, sofern M_{Init} nicht konstant. Dann ist $Init_0 = t(M_{Init})$ die aussagenlogische Kodierung der Startzustände von M mit

$$t(M_{Init}) = \begin{cases} value(v) & falls\ M_{Init}\ konstant \\ (c(p,0,true) \rightarrow t(M_{Init}|_{p \leftarrow true})) & \\ \wedge(c(p,0,\bot) \rightarrow t(M_{Init}|_{p \leftarrow \bot})) & sonst \\ \wedge(c(p,0,false) \rightarrow t(M_{Init}|_{p \leftarrow false})) & \end{cases}$$

Sei M_{Trans} das mehrwertige Entscheidungsdiagramm zur Repräsentation der Transitionsfunktion einer partiellen Kripke Struktur M. Zudem sei v der Wurzelknoten von M_{Trans} und es gelte $var(v) = p$, sofern M_{Trans} nicht konstant. Dann ist $T_{i,i+1} = t(M_{Trans}, i)$ die aussagenlogische Kodierung der Transitionsfunktion von M mit

$$t(M_{Trans}, i) = \begin{cases} value(v) & falls\ M_{Trans}\ konstant \\ (c(p,i,true) \rightarrow t(M_{Trans}|_{p \leftarrow true}, i)) & \\ \wedge(c(p,i,\bot) \rightarrow t(M_{Trans}|_{p \leftarrow \bot}, i)) & falls\ p \in AP \\ \wedge(c(p,i,false) \rightarrow t(M_{Trans}|_{p \leftarrow false}, i)) & \\ (c(p,i+1,true) \rightarrow t(M_{Trans}|_{p \leftarrow true}, i)) & \\ \wedge(c(p,i+1,\bot) \rightarrow t(M_{Trans}|_{p \leftarrow \bot}, i)) & falls\ p \in AP' \\ \wedge(c(p,i+1,false) \rightarrow t(M_{Trans}|_{p \leftarrow false}, i)) & \end{cases}$$

Korollar 4.1.1:
Sei M eine partielle Kripke Struktur, repräsentiert durch die mehrwertigen Entscheidungsdiagramme M_{Init} (Startzustände) und M_{Trans} (Transitionen). Dann gilt für ein $k \in \mathbb{N}$

$$[\![M]\!]_k := t(M_{Init}) \wedge \bigwedge_{i=0}^{k-1} t(M_{Trans}, i)$$

Die nachfolgende Abbildung 4.1.3 verdeutlicht, wie die rekursive Funktion t ein mehrwertiges Entscheidungsdiagramm durchläuft.

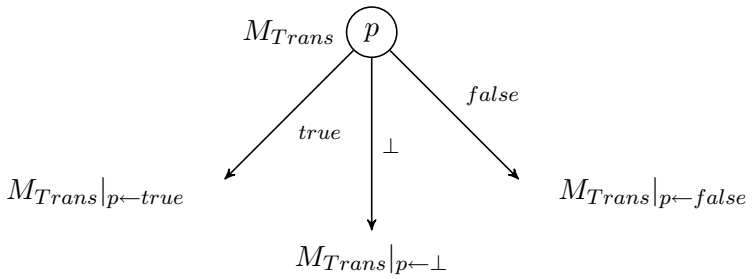

Abbildung 4.1.3: Dreiwertiges Entscheidungsdiagramm

Exemplarisch sei hier die aussagenlogische Kodierung der Startzustände anhand des Entscheidungsdiagramms M_{Init} in Abbildung 4.1.2 (a) betrachtet.

$$
\begin{aligned}
Init_0 = \quad & (p_A^0 \rightarrow false) \\
& \wedge ((\neg p_A^0 \wedge p_B^0) \rightarrow ((q_A^0 \rightarrow true) \wedge ((\neg q_A^0 \wedge q_B^0) \rightarrow false) \wedge ((\neg q_A^0 \wedge \neg q_B^0) \rightarrow false))) \\
& \wedge ((\neg p_A^0 \wedge \neg p_B^0) \rightarrow false) \\
= \quad & (\neg p_A^0) \wedge ((q_A^0 \vee \neg q_B^0) \wedge (q_A^0 \vee q_B^0)) \wedge (p_A^0 \vee p_B^0) \\
= \quad & \neg p_A^0 \wedge p_B^0 \wedge q_A^0
\end{aligned}
$$

Die Formel $Init_0$ ist für die Bewertung $I = \{p_A^0 \rightarrow false,\ p_B^0 \rightarrow true,\ q_A^0 \rightarrow true\}$ *wahr*. Gemäß der Definition der Kodierungsfunktion c beschreibt I die Aussage „Für die Präfixposition 0 gilt: der Wahrheitswert von p ist *unbekannt* und q ist *wahr*“. Damit charakterisiert I exakt die Menge der Startzustände $S_0 = \{s_0\} = \{(\bot, true)\}$ der Kripke Struktur in Abbildung 4.1.1. Offensichtlich entspricht die neu definierte aussagenlogische Kodierung einer partiellen Kripke Struktur keiner Normalform. Trotzdem lässt sich die Formel $[\![M]\!]_k$ effizient weiterverarbeiten, wie im nachfolgenden Abschnitt 4.2. erläutert.

Zunächst wird jedoch die LTL$^+$-Kodierung im Kontext mehrwertiger Entscheidungsdiagramme betrachtet. Im vorherigen Kapitel wurde gezeigt, dass sich beim Bounded

Model Checking für partielle Kripke Strukturen eine korrekte Kodierung temporallogischer Formeln *ohne* Berücksichtigung der Transitionsfunktion realisieren lässt. Einzig bei der Übersetzung atomarer Propositionen wird auf die zu Grunde liegende Kripke Struktur M zugegriffen. Dies betrifft die Zustandsmenge S sowie die Beschriftungsfunktion L von M (Vergleiche Definition 3.3.3). Sowohl S als auch L liegen bei der MDD-basierten Repräsentation des Zustandsraumes nicht explizit vor. Zudem ist durch die angepasste Konstruktion von $[\![M]\!]_k$ bereits die Atommenge $Atoms = \{p_A^i,\ p_B^i \mid p \in AP,\ 0 \leq i \leq k\}$ und durch die Kodierungsfunktion c die Semantik der Atome vorgegeben. $c(p, i, val)$ erzeugt die boolesche Kodierung der Aussage „Proposition p besitzt den Wahrheitswert val an der Präfixposition i“. Für die aussagenlogische Formel $[\![p]\!]_k^i$ soll gelten: Ist I_π eine Bewertung, die einen Pfad π charakterisiert, so besitzt $[\![p]\!]_k^i$ bewertet mit I_π den selben Wahrheitswert wie die Proposition p an der Position $\pi(i)$. Eine solche Formel lässt sich auf einfache Weise über die Kodierungsfunktion konstruieren: $(c(p, i, true) \wedge true) \vee (c(p, i, \bot) \wedge \bot) \vee (c(p, i, false) \wedge false)$. Dieser Ausdruck lässt sich kürzen, so dass für die aussagenlogische Kodierung atomarer Propositionen die folgende neue Definition ergibt. Für alle nicht-atomaren LTL$^+$-Formeln ist keine Anpassung der SAT-Übersetzung notwendig, da ihre Kodierung von der Kripke Struktur unabhängig ist.

Definition 4.1.6: LTL$^+$-Kodierung für atomare Propositionen
Sei AP die Menge atomarer Propositionen einer implizit repräsentierten Kripke Struktur M. Zudem seien $p, \bar{p} \in AP$ und $i, l, k \in \mathbb{N}$ mit $i, l \leq k$

$$
\begin{aligned}
[\![p]\!]_k^i &= p_A^i \vee (p_B^i \wedge \bot) \\
[\![\bar{p}]\!]_k^i &= \bar{p}_A^i \vee (\bar{p}_B^i \wedge \bot) \\
{}_l[\![p]\!]_k^i &= p_A^i \vee (p_B^i \wedge \bot) \\
{}_l[\![\bar{p}]\!]_k^i &= \bar{p}_A^i \vee (\bar{p}_B^i \wedge \bot)
\end{aligned}
$$

An dem 1-Präfix eines Pfades π in Abbildung 4.1.4 wird die neue Kodierung noch einmal beispielhaft gezeigt. Für den Präfix ergibt sich gemäß der Funktion c die Kodierung $T_{0,1} = \neg p_A^0 \wedge p_B^0 \wedge p_A^1$ und damit die charakterisierende Bewertung $I_\pi = \{p_A^0 \rightarrow false, p_B^0 \rightarrow true, p_A^1 \rightarrow true\}$. Die Formel $[\![\mathbf{F}p]\!]_1^0 = [p_A^0 \vee (p_B^0 \wedge \bot)] \vee [p_A^1 \vee (p_B^1 \wedge \bot)]$ bewertet mit I_π ist *wahr* und offensichtlich gilt auch $[\pi \models_1^0 \mathbf{F}p] = true$.

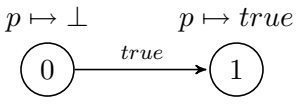

Abbildung 4.1.4: 1-Präfix eines Pfades π

Damit ist gezeigt, wie sich die SAT-Reduktion eines Bounded Model Checking Problems realisieren lässt, wenn die partielle Kripke Struktur implizit durch mehrwertige Entscheidungsdiagramme beschrieben wird. Der nachfolgende Abschnitt behandelt die Weiterverarbeitung der erzeugten aussagenlogischen Formel bis hin zum Erfüllbarkeitstest.

4.2. Effiziente Repräsentation aussagenlogischer Formeln

Die im vorherigen Abschnitt vorgestellte Kodierung einer partiellen Kripke Struktur $[\![M]\!]_k = Init_0 \wedge \bigwedge_{i=0}^{k-1} T_{i,\,i+1} = t(M_{Init}) \wedge \bigwedge_{i=0}^{k-1} t(M_{Trans}, i)$ weist keine Normalform auf und ist damit als direkte Eingabe für einen SAT-Solver ungeeignet. Im Folgenden wird daher die schrittweise Umformung von $[\![M]\!]_k$ in die Konjunktive Normalform (KNF) gezeigt, dem Standard Eingabeformat aller etablierten Solver (z. B. MiniSat [ES05], Chaff [MMZ+01]). Dazu wird exemplarisch die Teilformel $t(M_{Init})$ betrachtet.

$$
\begin{aligned}
t(M_{Init}) \quad = \quad & (p_A^0 \rightarrow t(M_{Init}|_{p \leftarrow true})) \\
& \wedge ((\neg p_A^0 \wedge p_B^0) \rightarrow t(M_{Init}|_{p \leftarrow \bot})) \\
& \wedge ((\neg p_A^0 \wedge \neg p_B^0) \rightarrow t(M_{Init}|_{p \leftarrow false}))
\end{aligned}
$$

Für eine effiziente maschinelle Verarbeitung boolescher Ausdrücke ist es zweckmäßig, diese durch die Verwendung geeigneter Datenstrukturen zu repräsentieren. Reduzierte Boolesche Schaltkreise (englisch: *Reduced Boolean Circuits, RBCs*) [ABE00] ermöglichen eine graphenbasierte Darstellung aussagenlogischer Formeln. Effizienzvorteile ergeben sich aus der Tatsache, dass isomorphe Teilgraphen hierbei zusammengefasst werden. Zudem lässt sich die repräsentierte Gesamtformel auf einfache Weise in einen erfüllbarkeitsäquivalenten KNF-Ausdruck umformen. Auch die Bounded Model Checking Komponente des etablierten Model Checkers NuSMV [CCG+02] setzt boolesche Schaltkreise zur KNF-Konvertierung ein.

Definition 4.2.1: Boolescher Schaltkreis
Sei $Var = AP \cup \{true, \bot, false\}$ *eine Atommenge. Ein* Boolescher Schaltkreis *ist ein gerichteter azyklischer Graph* (V, E)*, wobei gilt*

- $V = V_I \cup V_L$ *ist eine Menge von internen Knoten* (V_I) *und Blättern* (V_L).
- *Jeder interne Knoten* $v \in V_I$ *besitzt die folgenden Attribute: Einen binären Operator* $op(v) \in \{\leftrightarrow, \wedge\}$*, und zwei ausgehende Kanten* $left(v), right(v) \in E$.
- *Jedes Blatt* $v \in V_L$ *besitzt ein Attribut* $var(v) \in Var$.

- *Jede Kante $e \in E$ besitzt die Attribute $sign(e) \in \{true, false\}$ und $target(e) \in V$.*

Definition 4.2.2: Reduzierter Boolescher Schaltkreis
Ein Reduzierter Boolescher Schaltkreis *ist ein Boolescher Schaltkreis mit den folgenden zusätzlichen Eigenschaften:*

- *Alle isomorphen Teilgraphen sind zusammengefasst.*
- *Blätter $v \in V_L$ mit $var(v) = true$ sind nur für konstante RBCs zulässig.*
- *Für alle internen Knoten $v \in V_I$ gilt $left(v) \neq right(v)$.*
- *Falls $op(v) = \leftrightarrow$ dann gilt $sign(left(v)) = sign(right(v)) = true$.*
- *Es existiert eine totale Ordnung $\prec$, so dass für alle Knoten $v \in V_I$ gilt $left(v) \prec right(v)$.*

Die Negation von Teilformeln wird bei RBCs durch das Kanten-Attribut *sign* realisiert. Ein Teilgraph, für dessen Eingangskante e gilt $sign(e) = false$, repräsentiert einen negierten booleschen Ausdruck. Ferner beschränkt sich die Menge der zweistelligen Junktoren auf $\wedge$ und $\leftrightarrow$, so dass sich die Formel $t(M_{Init})$ nicht direkt als boolescher Schaltkreis darstellen lässt. Durch die Anpassung der Funktion t (Anwendung der De Morganschen Regeln und weitere Äquivalenzumformungen) ergibt sich für $t(M_{Init})$ ein logisch äquivalenter Ausdruck, dessen RBC-Repräsentation in Abbildung 4.2.1 zu sehen ist.

Definition 4.2.3:
Sei M_{Init} das mehrwertige Entscheidungsdiagramm zur Repräsentation der Startzustände einer partiellen Kripke Struktur M. Zudem sei v der Wurzelknoten von M_{Init} und es gelte $var(v) = p$, sofern M_{Init} nicht konstant. Dann ist $Init_0 = t(M_{Init})$ die aussagenlogische Kodierung der Startzustände von M mit

$$t(M_{Init}) = \begin{cases} value(v) & M_{Init} \text{ konstant} \\ \neg(p_A^0 \wedge \neg t(M_{Init}|_{p \leftarrow true})) & \text{sonst} \\ \wedge\neg(\neg p_A^0 \wedge \neg[\neg(p_B^0 \wedge \neg t(M_{Init}|_{p \leftarrow \bot})) \wedge \neg(\neg p_B^0 \wedge \neg t(M_{Init}|_{p \leftarrow false}))]) & \end{cases}$$

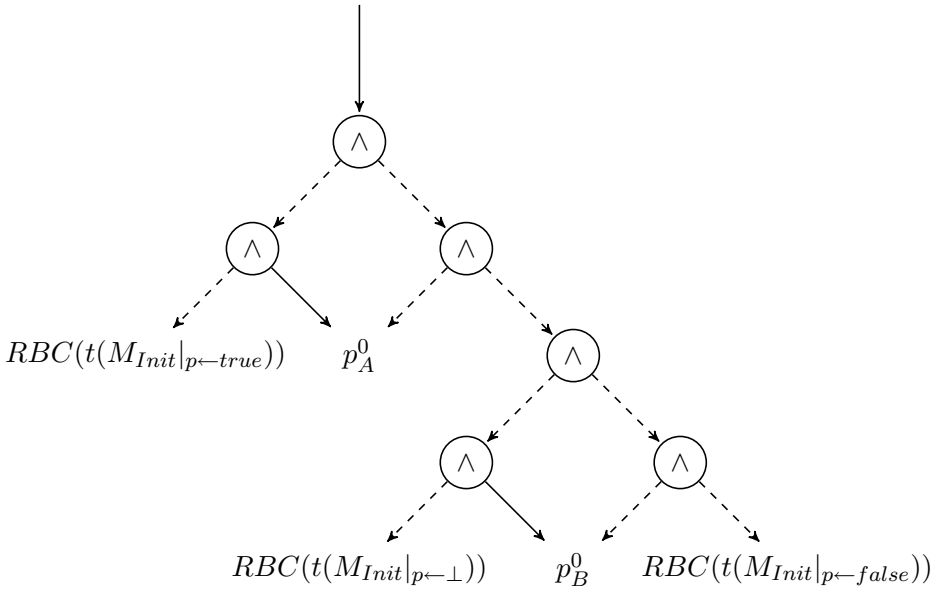

Abbildung 4.2.1: Reduzierter Boolescher Schaltkreis für $t(M_{Init})$, Kanten mit negativem Vorzeichen sind gestrichelt dargestellt

Die im Rahmen dieser Arbeit erfolgte Implementierung eines Bounded Model Checkers erzeugt reduzierte boolesche Schaltkreise direkt aus den gegebenen mehrwertigen Entscheidungsdiagrammen M_{Init} und M_{Trans}. Während die MDDs Ausdrücke der dreiwertigen Logik repräsentieren, beschreiben die RBCs zweiwertige aussagenlogische Formeln mit der zusätzlichen Konstanten $\bot$. Aus den booleschen Schaltkreisen lassen sich nun auf einfache Weise erfüllbarkeitsäquivalente Formeln in konjunktiver Normalform ableiten.

Definition 4.2.4: Konjunktive Normalform
Eine aussagenlogische Formel α ist in konjunktiver Normalform *($\alpha \in KNF$), wenn sie eine Konjunktion von Disjunktionstermen ist. Disjunktionsterme, auch Klauseln genannt, sind disjunktive Verknüpfungen von Literalen. Literale sind nicht-negierte oder negierte Atome. Eine Formel in KNF hat also die Form*

$$\bigwedge_i \bigvee_j (\neg) x_{ij}$$

Die konjunktive Normalform ist Ausgangspunkt vieler Entscheidungsprozeduren des aussagenlogischen Erfüllbarkeitsproblems (z. B. DPLL-Algorithmus [DLL62], Resolutionskalkül [Rob65]), welche wiederum die Grundlage für SAT-Solver Implementierungen bilden. Allerdings kann die Äquivalenzumformung nach KNF zu einer exponentiell längeren Formel führen. Eine Abschwächung der logischen Äquivalenz ist die Erfüllbarkeitsäquivalenz.

Definition 4.2.5: Erfüllbarkeitsäquivalenz

Zwei aussagenlogische Formeln α und β sind genau dann erfüllbarkeitsäquivalent ($\alpha \overset{SAT}{\approx} \beta$)*, wenn gilt*

$$\alpha \textit{ ist erfüllbar} \Leftrightarrow \beta \textit{ ist erfüllbar}$$

Über erfüllende Bewertungen wird hierbei keine Aussage gemacht. Insbesondere müssen erfüllbarkeitsäquivalente Formeln weder für identische Bewertungen wahr sein, noch über die gleichen Mengen vorkommender Atome verfügen.

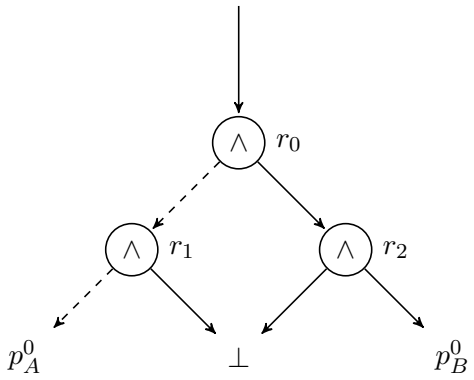

Abbildung 4.2.2: Reduzierter Boolescher Schaltkreis für die Formel $\alpha = \neg(\neg p_A^0 \wedge \bot) \wedge (\bot \wedge p_B^0)$

Aus einem RBC lässt sich wie folgt eine erfüllbarkeitsäquivalente KNF-Formel ableiten [ABE00]. Für jeden inneren Konten v_i wird ein neues Atom r_i eingeführt. r_i wird äquivalent gesetzt zu dem booleschen Ausdruck, repräsentiert durch den Teilgraphen mit Wurzel v_i. Für den Schaltkreis in Abbildung 4.2.2 ergibt sich die Formel

$$\begin{aligned}
\beta \quad &= \quad r_0 \wedge [r_0 \leftrightarrow (\neg r_1 \wedge r_2)] \wedge [r_1 \leftrightarrow (\neg p_A^0 \wedge \bot)] \wedge [r_2 \leftrightarrow (\bot \wedge p_B^0)] \\
&\approx \quad (r_0) \wedge (r_0 \vee r_1 \vee \neg r_2) \wedge (\neg r_0 \vee \neg r_1) \wedge (\neg r_0 \vee r_2) \\
&\quad \wedge (r_1 \vee p_A^0 \vee \neg \bot) \wedge (\neg r_1 \vee \neg p_A^0) \wedge (\neg r_1 \vee \bot) \\
&\quad \wedge (r_2 \vee \neg \bot \vee \neg p_B^0) \wedge (\neg r_2 \vee \bot) \wedge (\neg r_2 \vee p_B^0)
\end{aligned}$$

Die Klauselanzahl des erzeugten Ausdrucks β in konjunktiver Normalform steht in einem linearen Verhältnis zur Größe des RBCs. Für jeden nicht-terminalen Knoten beschriftet mit $\wedge$ werden drei Klauseln generiert.

Zudem gewährleistet diese Umformung einen weiteren wichtigen Fakt. Die neue Formel β ist aufgrund der vergrößerten Atomanzahl zwar nicht logisch äquivalent zum ursprünglichen Ausdruck α, jede ihrer erfüllenden Bewertungen macht jedoch auch α wahr. Diese Eigenschaft ist für das Bounded Model Checking unverzichtbar. Denn jede erfüllende Bewertung eines aussagenlogisch kodierten BMC-Problems charakterisiert ein konkretes Gegenbeispiel für die überprüfte LTL-Formel und liefert somit Hinweise, inwiefern das Modell zu verändern ist.

Damit liegt eine effiziente Technik vor, eine durch mehrwertige Entscheidungsdiagramme beschriebene Kripke Struktur aussagenlogisch zu kodieren und die Kodierung in konjunktive Normalform zu transformieren. Es lassen sich folglich aus gegebenen MDDs M_{Init} und M_{Trans}, KNF-Repräsentationen der Formel $[\![M]\!]_k$ sowie der Schleifenbedingung ${}_lL_k = T_{k,l}$ konstruieren. Im Weiteren wird gezeigt, wie dies auch für die Kodierung der temporallogischen Formel φ erreicht werden kann. Exemplarisch wird dazu die Kodierung *mit* Schleife betrachtet. Die SAT-Übersetzung für LTL$^+$-Formeln gemäß Definition 3.3.5 ist für die Operatoren **G**, **F** und **U** iterativ formuliert. Eine rekursive Definition der LTL-Kodierung findet sich in [BCC+03]. Diese ermöglicht eine sehr einfache Konstruktion eines kompakten RBCs, korrespondierend zu einer Formel ${}_l[\![\varphi]\!]_k^i$.

Definition 4.2.6: Rekursive LTL$^+$-Kodierung mit Schleife
Sei φ eine LTL$^+$-Formel und $k, i, l \in \mathbb{N}$ mit $i, l \leq k$. Dann ist $k - min(i,l) + 1$ die vorgegebene Rekursionstiefe und es gilt

$$\begin{aligned}
{}_l[\![p]\!]_k^i &= p_A^i \vee (p_B^i \wedge \bot) \\
{}_l[\![\bar{p}]\!]_k^i &= \bar{p}_A^i \vee (\bar{p}_B^i \wedge \bot) \\
{}_l[\![\varphi_1 \wedge \varphi_2]\!]_k^i &= {}_l[\![\varphi_1]\!]_k^i \wedge_l [\![\varphi_2]\!]_k^i \\
{}_l[\![\varphi_1 \vee \varphi_2]\!]_k^i &= {}_l[\![\varphi_1]\!]_k^i \vee_l [\![\varphi_2]\!]_k^i
\end{aligned}$$

$$
\begin{aligned}
{}_l[\![\mathbf{G}\varphi]\!]^i_k &= {}_l[\![\varphi]\!]^i_k \wedge_l [\![\mathbf{G}\varphi]\!]^{succ(i)}_k \\
{}_l[\![\mathbf{F}\varphi]\!]^i_k &= {}_l[\![\varphi]\!]^i_k \vee_l [\![\mathbf{F}\varphi]\!]^{succ(i)}_k \\
{}_l[\![\mathbf{X}\varphi]\!]^i_k &= {}_l[\![\varphi]\!]^{succ(i)}_k \\
{}_l[\![\varphi_1 \mathbf{U} \varphi_2]\!]^i_k &= {}_l[\![\varphi_2]\!]^i_k \vee ({}_l[\![\varphi_1]\!]^i_k \wedge_l [\![\varphi_1 \mathbf{U} \varphi_2]\!]^{succ(i)}_k)
\end{aligned}
$$

Die im Rahmen dieser Arbeit erfolgte Implementierung eines Bounded Model Checkers orientiert sich bei der Konstruktion boolescher Schaltkreise zur Kodierung von LTL$^+$-Formeln an der obigen Definition. Der Aufbau eines RBCs zur Repräsentation einer temporallogischen Formel φ geschieht rekursiv. Die nachfolgende Abbildung zeigt exemplarisch den Schaltkreis für ${}_l[\![\varphi_1 \mathbf{U} \varphi_2]\!]^i_k$.

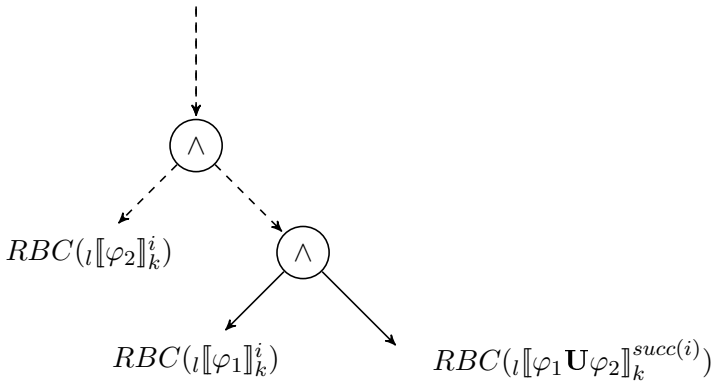

Abbildung 4.2.3: RBC für ${}_l[\![\varphi_1 \mathbf{U} \varphi_2]\!]^i_k = {}_l[\![\varphi_2]\!]^i_k \vee ({}_l[\![\varphi_1]\!]^i_k \wedge_l [\![\varphi_1 \mathbf{U} \varphi_2]\!]^{succ(i)}_k)$

Es ist somit gezeigt, wie sich die aussagenlogischen Kodierungen von Kripke Struktur und temporallogischer Formel als reduzierter boolescher Schaltkreis darstellen lassen. Des Weiteren wurde ein Verfahren vorgestellt, um aus einem RBC eine erfüllbarkeitsäquivalente Formel in konjunktiver Normalform abzuleiten, deren erfüllende Bewertungen ebenfalls die ursprüngliche Formel wahr machen. Folglich sind alle Voraussetzungen gegeben, die KNF-Repräsentation der Kodierung eines Bounded Model Checking Problems $[\![M, \varphi]\!]_k$ zu erzeugen und es kann der abschließende Erfüllbarkeitstest erfolgen.

Damit die Erfüllbarkeit einer aussagenlogischen KNF-Formel α durch einen SAT-Solver überprüft werden kann, muss α in einem entsprechenden Dateiformat vorliegen. Ein solches Format ist DIMACS-CNF [DIM93].

Definition 4.2.7: DIMACS-CNF
DIMACS-CNF ist ein Dateiformat zur Beschreibung aussagenlogischer Formeln in konjunktiver Normalform. DIMACS-CNF-Dateien beginnen mit einer Präambel, welche die Atom- und Klauselanzahl der repräsentierten KNF-Instanz angibt:

$$p\ cnf\ \ < Atomanzahl > < Klauselanzahl >$$

Nach einem Zeilenumbruch folgen die Klauseln. Boolesche Atome werden hierbei kodiert durch positive Integer-Werte. Negierte Vorkommen einer Variable i werden durch $-i$ dargestellt. Eine Klausel ist eine Folge von solchen positiven oder negativen Literalen, jeweils separiert durch ein Leerzeichen. Das Ende einer Klausel wird symbolisiert durch den Wert 0.

Eine mögliche DIMACS-CNF-Kodierung für die Formel $\alpha = (x_1 \vee x_3 \vee \neg x_4) \wedge (x_4) \wedge (x_2 \vee \neg x_3)$ ist somit:

$p\ cnf\ 4\ 3$
$1\ 3\ -4\ 0$
$4\ 0$
$2\ -3\ 0$

Für die DIMACS-CNF-Kodierung eines Bounded Model Checking Problems bleibt festzulegen, wie die Zuordnung von DIMACS-Variablen zu booleschen Atomen erfolgt. Ein aussagenlogisch kodiertes BMC-Problem über AP ist definiert über einer Menge von Grundatomen $Atoms = \{p_A^i,\ p_B^i \mid p \in AP,\ 0 \leq i \leq k\}$ mit $|Atoms| = 2 \cdot |AP| \cdot (k+1)$ und wird als reduzierter boolescher Schaltkreis dargestellt. Die Konvertierung einer durch einen RBC repräsentierten Formel nach KNF erfordert zudem die Einführung einer Menge von Zusatzatomen $Atoms_R = \{r_0, r_1, ...\}$. Jedes Atom r_j korrespondiert hierbei zu einem nicht-terminalen Knoten v_j. Somit ist die Größe von $Atoms_R$ davon abhängig, inwieweit sich der boolesche Schaltkreis durch das Zusammenfassen isomorpher Teilgraphen reduzieren lässt.

Die nachfolgend definierte Funktion beschreibt die Zuordnung von DIMACS-Variablen zu Atomen, wie sie im Rahmen der Implementierung umgesetzt wurde. Die gewählte Zuordnung bietet den entscheidenden Vorteil, dass zur Konstruktion von $[\![M]\!]_k$ lediglich die Formel $Init_0 \wedge T_{0,1}$ berechnet werden muss. Alle weiteren Teilformeln $T_{i,i+1}$ mit $1 \leq i < k$ lassen sich dann durch ein „Shiften“ der DIMACS-Kodierung von $T_{0,1}$ generieren.

Definition 4.2.8: DIMACS-Kodierung von Atomen
Sei $Atoms = \{p_A^i,\ p_B^i \mid p \in AP,\ 0 \leq i \leq k\}$ *die Menge der Grundatome und* $Atoms_R = \{r_0, r_1, ...\}$ *die Menge der Zusatzatome eines aussagenlogisch kodierten Bounded Model Checking Problems über* AP *mit Bound* $k \in \mathbb{N}$. *Zudem sei* $AP = \{p_0, p_1, ...\}$ *geordnet und es gelte* $\forall j, 0 \leq j < |AP| : index(p_j) = j$. *Dann beschreibt die Funktion* $dimacs : Atoms \cup Atoms_R \times i \to \mathbb{N}_{>0}$ *die DIMACS-Kodierung von Atomen mit*

$$dimacs(p_A^i, i) = 2 \cdot (index(p) + |AP| \cdot i) + 1$$
$$dimacs(p_A^i, i) = 2 \cdot (index(p) + |AP| \cdot i) + 2$$
$$dimacs(r_j, i) = 2 \cdot |AP| \cdot (k+1) + 1 + j + |Atoms_R| \cdot i$$

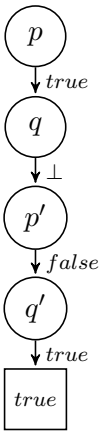

Abbildung 4.2.4: MDD M_{Trans}

Zur Veranschaulichung sei das mehrwertige Entscheidungsdiagramm M_{Trans} in Abbildung 4.2.4 betrachtet. Es gilt $AP = \{p, q\}$ und somit $index(p) = 0$ und $index(q) = 1$. Offensichtlich gilt auch die folgende aussagenlogische Kodierung der Transitionsfunktion:

$$T_{i,i+1} \quad = \quad p_A^i \wedge \neg q_A^i \wedge q_B^i \wedge \neg p_A^{i+1} \wedge \neg p_B^{i+1} \wedge q_A^{i+1}$$

Gemäß der Funktion *dimacs* ergibt sich

$$T_{0,1} \quad = \quad 1 \wedge -3 \wedge 4 \wedge -5 \wedge -6 \wedge 7$$

$$\begin{aligned} T_{1,2} \quad &= \quad (1+4) \wedge -(3+4) \wedge (4+4) \wedge -(5+4) \wedge -(6+4) \wedge (7+4) \\ &= \quad 5 \wedge -7 \wedge 8 \wedge -9 \wedge -10 \wedge 11 \end{aligned}$$

Liegt also die DIMACS-Kodierung von $T_{0,1}$ vor, so ergibt sich die Kodierung von $T_{i,i+1}$, indem die Beträge der DIMACS-Variablen der Grundatome von $T_{0,1}$ und den Wert $2 \cdot |AP| \cdot i$ geshiftet werden.

Die Formel $T_{0,1}$ enthält im Allgemeinen auch Zusatzatome. Für diese gelten gesonderte Shift-Regeln. Da die Anzahl der Grundatome mit $|Atoms| = 2 \cdot |AP| \cdot (k+1)$ fest ist, werden die Zusatzatome in der DIMACS-Darstellung fortlaufend durchnummeriert, beginnend mit $|Atoms| = 2 \cdot |AP| \cdot (k+1) + 1$. Sei nun $|R|$ die Anzahl Zusatzatome in $T_{0,1}$, dann werden diese zur Generierung von $T_{i,i+1}$ jeweils um den Wert $|R| \cdot i$ geshiftet.

4.3. Beschreibung der Implementierung

Aufbauend auf dem Konzept aus Kapitel 3. und den zuvor erläuterten Umsetzungstechniken wurde als Bestandteil dieser Arbeit ein Bounded Model Checker für partielle Kripke Strukturen implementiert und in die Anwendung 3Spot integriert. 3Spot, entwickelt in der Arbeitsgruppe Wehrheim, ist die Umsetzung eines Abstraktionsverfahrens für parallele Systeme mit kommunizierenden Komponenten. Der Zustandsraum solcher Systeme ist aufgrund der Nebenläufigkeit von Prozessen und der damit verbundenen verschränkten Ausführung von Operationen oftmals zu komplex für eine direkte und vollständige Verifikation. Abstraktionen des zu Grunde liegenden Systems stellen eine Herangehensweise dar, diesem Problem zu begegnen [CGL92]. Hierbei wird eine Reduktion des Zustandsraumes erzielt, indem dieser nicht über genaue Variablenwerte, sondern lediglich über eine Reihe von Propositionen bezüglich der Variablenwerte definiert wird [BHJ+07]. Zudem existieren speziell für parallele Systeme so genannte Spotlight-Abstraktionstechniken [WW07]. Deren Prinzip ist es, die einzelnen Komponenten des Systems in unterschiedlichen Genauigkeitsgraden darzustellen. Einzelne Prozesse stehen im „Scheinwerferlicht“ und werden in der Abstraktion sehr detailliert betrachtet, während die restlichen Prozesse nur grob beschrieben werden. Die Unschärfe abstrakter Systemrepräsentationen lässt sich sehr natürlich durch partielle Kripke Strukturen modellieren. Liegt zudem eine temporallogische Formel vor, deren Gültigkeit für das Modell überprüft werden soll, so ergibt sich ein mehrwertiges Model Checking Problem. Besitzt dieses Problem aufgrund der Unvollständigkeit der Systembeschreibung keinen definiten Wahrheitswert, lässt sich durch die Hinzunahme neuer Propositionen oder der Erhöhung des Genauigkeitsgrades einzelner Komponenten eine Verfeinerung der Abstraktion erzeugen. Die Grundidee ist es, den zu untersuchenden Zustandsraum solange schrittweise zu vergrößern, bis die Gültigkeit der temporallogischen Eigenschaft eindeutig bewiesen oder widerlegt werden kann.

Auch 3Spot arbeitet nach diesem Prinzip. Als Eingabe erhält es ein C-Programm, bestehend aus parallelen Komponenten mit gemeinsamen Variablen, sowie eine CTL-Formel φ. Die Anwendung erzeugt nun eine abstrakte Darstellung des parallelen Systems über einer Menge von Propositionen. Anschließend wird diese Abstraktion in zwei mehrwertige Entscheidungsdiagramme transformiert. Die MDDs und die CTL-Eigenschaft werden an

den Model Checker χChek übergeben. Falls dieser kein definites Ergebnis liefert, sondern einen Pfad mit Wahrheitswert *unbekannt* für φ zurück gibt, so erfolgt anhand dieses Pfades eine Abstraktionsverfeinerung. Es werden neue Propositionen zur Abstraktion hinzugefügt oder weitere Prozesse ins „Scheinwerferlicht“ gestellt.

Das Abstraktionsverfahren 3SPOT ist also eine konkrete Quelle partieller Systembeschreibungen. Die generierten MDDs sind definiert über der Menge von Wahrheitswerten $D = \{true, \bot, false\}$. Sie entsprechen den in Abschnitt 4.1. eingeführten Entscheidungsdiagrammen M_{Init} und M_{Trans} zur Repräsentation partieller Kripke Strukturen. Während das Model Checking in 3SPOT bisher ausschließlich durch χChek realisiert wurde, ist im Rahmen dieser Arbeit zusätzlich ein Bounded Model Checker integriert worden. Es stehen somit zwei alternative Verfahren zur Verfügung, die Verifikation der partiellen Systembeschreibungen durchzuführen.

Das Java-Projekt 3SPOT wurde dazu um das neu entwickelte Paket *bmc* erweitert. Objekte der darin enthaltenen Klasse *BMCInstance* beschreiben Bounded Model Checking Probleminstanzen. Als Übergabeparameter erhält eine Instanz die Entscheidungsdiagramme M_{Init} und M_{Trans}, die zugehörige Menge der Propositionen AP, sowie eine LTL$^+$-Formel φ und einen Bound k. Die MDDs werden von 3SPOT als abstrakte Beschreibung eines C-Programms generiert. Da 3SPOT bisher auf die Verifikation temporallogischer Eigenschaften aus CTL beschränkt war, das entwickelte Bounded Model Checking Konzept jedoch auf der linearen Temporallogik basiert, wurde zudem eine neue Klasse *LTL* eingeführt, über die sich LTL$^+$-Formeln instanziieren lassen. Bei der Erzeugung eines Objekts der Klasse *BMCInstance* wird nun die aussagenlogische Kodierung eines Bounded Model Checking Problems für partielle Systembeschreibungen generiert. Die Vorgehensweise entspricht dabei dem Prinzip, das in den vorangegangenen Abschnitten umfassend erläutert wurde. Die resultierende Formel im DIMACS-CNF-Format enthält aufgrund der Unvollständigkeit des betrachteten Zustandsraumes eine Konstante mit der Bedeutung *unbekannt*. Nun wird jeweils für die optimistische und die pessimistische Vervollständigung dieser Formel der Erfüllbarkeitstest durch den Aufruf des SAT-Solvers MiniSat durchgeführt (Vergleiche Abschnitt 3.4.). Ist die pessimistische Vervollständigung erfüllbar, so beschreibt die erfüllende Bewertung einen echten Zeugenpfad für φ. Existiert lediglich für die optimistische Vervollständigung eine erfüllende Bewertung, dann ist für den korrespondierenden Pfad nicht bekannt, ob dieser ein Zeuge für φ ist. Sind beide Formeln unerfüllbar, so existiert in dem betrachteten Model kein Zeugenpfad der Länge k für φ. Der Bounded Model Checker gibt abschließend den Wahrheitswert der BMC-Probleminstanz aus. Ist dieser Wert nicht *false*, wird zudem der entsprechende Pfadpräfix zurückgegeben.

```
bool lock = false;
int x = 2;

BEGIN_PROGRAM
/*Program [0]*/
int main() {
        while(!lock && x > 0) {
                x--;
        }
END: ;
}

END_PROGRAM

BEGIN_PROGRAM
/*Program [1]*/
int main() {
        lock = true;
END: ;
}

END_PROGRAM
```

Abbildung 4.3.1: Parallele C-Programme

Im Folgenden wird der Funktionsweise von 3Spot im Zusammenwirken mit dem Bounded Model Checker exemplarisch erläutert. Betrachtet sei der Quellcode in Abbildung 4.3.1. Die parallelen Prozesse 0 und 1 nutzen beide die globale Variable *lock*. Nun soll geprüft werden, ob ein Programmablauf existiert, so dass Prozess 0 terminiert. 3Spot konstruiert eine erste Abstraktion des Systems über den Programmzählern der Prozesse und der Proposition $\neg(\neg lock \wedge (x > 0))$ (negierte Schleifenbedingung). Zunächst steht nur der Prozess 0 im „Scheinwerferlicht“, somit bleibt die aktuelle Ausführungsposition von Prozess 1 dauerhaft unbekannt. Gesucht wird nun ein Zeugenpfad für die LTL-Formel $\mathbf{F}(pc\_0 = END)$. Für die betrachtete Systembeschreibung besitzt der kürzeste Pfadpräfix, auf den diese Eigenschaft zutrifft, die Länge 5. Der Model Checker liefert als Ausgabe:

```
LTL formula evaluates TRUE for the given program and bound k = 5

Propositions:
!(!lock && (x>0)), pc_0=0, pc_0=1, pc_0=2, pc_0=END, pc_1=0, pc_1=1, pc_1=END

Witness:
i=0: (F, T, F, F, F, M, M, M); -T-> i=1: (F, F, F, T, F, M, M, M);
-T-> i=2: (F, T, F, F, F, M, M, M); -T-> i=3: (T, F, F, T, F, M, M, M);
-T-> i=4: (T, T, F, F, F, M, M, M); -T-> i=5: (T, F, F, F, T, M, M, M);
```

Dieser Präfix beschreibt den folgenden Programmablauf:

- Prozess 0 prüft die Schleifenbedingung $\neg lock \wedge (x > 0)$, diese ist erfüllt.
- Prozess 0 dekrementiert x auf 1.
- Prozess 0 prüft die Schleifenbedingung $\neg lock \wedge (x > 0)$, diese ist erfüllt.
- Prozess 0 dekrementiert x auf 0.
- Prozess 0 prüft die Schleifenbedingung $\neg lock \wedge (x > 0)$, diese ist *nicht* erfüllt.
- Prozess 0 terminiert.

Aus dem Quelltext in Abbildung 4.3.1 wird ersichtlich, dass die while-Schleife auch dann terminiert, wenn *lock* wahr ist, beziehungsweise wenn Prozess 1 ausgeführt wurde. Somit existiert offensichtlich auch ein kürzerer Zeugenpfad für $\mathbf{F}(pc\_0 = END)$. Prozess 1 ist jedoch nicht Gegenstand der Abstraktion. Für $k = 2$ ergibt sich:

```
LTL formula evaluates MAYBE for the given program and bound k = 2

Propositions:
!(!lock && (x>0)), pc_0=0, pc_0=1, pc_0=2, pc_0=END, pc_1=0, pc_1=1, pc_1=END

Witness:
i=0: (F, T, F, F, F, M, M, M); -T-> i=1: (M, T, F, F, F, M, M, M);
-M-> i=2: (M, F, F, F, T, M, M, M);
```

An Position $i = 1$ des Präfix ist der Wahrheitswert der negierten Schleifenbedingung $!(!lock \;\&\&\; (x > 0))$ unbekannt. Möglicherweise wurde dieser Wert durch Prozess 1 verändert. Somit ist auch nicht bekannt, ob ein direkter Nachfolgezustand existiert in dem Prozess 0 terminiert.

Wird jedoch eine verfeinerte Abstraktion betrachtet, welche auch den Prozess 1 berücksichtigt, so liefert der Bounded Model Checker einen echten Zeugenpfad der Länge 2 für $\mathbf{F}(pc\_0 = END)$:

```
LTL formula evaluates TRUE for the given program and bound k = 2

Propositions:
!(!lock && (x>0)), pc_0=0, pc_0=1, pc_0=2, pc_0=END, pc_1=0, pc_1=1, pc_1=END

Witness:
i=0: (F, T, F, F, F, T, F, F); -T-> i=1: (T, T, F, F, F, F, F, T);
-T-> i=2: (T, F, F, F, T, F, F, T);
```

Nachdem Prozess 1 terminiert ist und die negierte Schleifenbedingung $!(!lock \;\&\&\; (x > 0))$ wahr gemacht hat, kann Prozess 0 terminieren, ohne den Rumpf der while-Schleife auszuführen.

Der implementierte Bounded Model Checker für partielle Kripke Strukturen wird also hier zur Verifikation von Systemen eingesetzt, die aufgrund einer abstrakten und somit reduzierten Darstellung nur teilweise bekannt sind. Wird ein Präfix bestimmt, dessen Zeugeneigenschaft für die betrachtete temporallogische Formel unbekannt ist, so kann dieser als Grundlage für weitere Verfeinerungsschritte dienen. Das Finden von Gegenbeispielen beziehungsweise Zeugenpfaden minimaler Länge ist eine Eigenschaft, die alle inkrementell vorgehenden BMC-Verfahren miteinander verbindet. Hier zeigt sich, dass die Länge des gefundenen Gegenbeispiels vom Grad der Abstraktion abhängig ist. Für verfeinerte Systembeschreibungen existieren gegebenenfalls kürzere Pfadpräfixe, welche definite Zeugen für die zu überprüfende LTL-Formel sind. Die Kombination von Abstraktionstechniken und Bounded Model Checking gestattet es also, sowohl durch die Verfeinerung des Modells als auch durch die Anpassung der Schranke k, die Komplexität der Probleminstanz schrittweise zu verändern. Somit sind erweiterte Möglichkeiten gegeben, den Zustandsraum zielgerichtet zu durchsuchen und effizient temporallogische Eigenschaften nachzuweisen.

Abstraktionen bilden jedoch nicht das einzig denkbare Einsatzgebiet des entwickelten Bounded Model Checkers. Selbstadaptive Systeme, inkonsistente Spezifikationen und andere Systeme mit nur teilweise bekannten Zustandsräumen sind ebenfalls potentielle Quellen für dreiwertige Kripke Strukturen. Durch entsprechende Anpassungen des Ein-

gabeformats der implementierten Anwendung lassen sich auch solche Systeme mit der konzeptionierten BMC-Technik verifizieren.

Kapitel 5.

Schluss

In diesem Kapitel werden noch einmal die wichtigsten Erkenntnisse und Ergebnisse dieser Arbeit zusammengefasst. Zudem erfolgt ein Ausblick auf sinnvolle weiterführende Forschungen im Bereich Bounded Model Checking für partielle Kripke Strukturen.

5.1. Zusammenfassung

Im Rahmen dieser Arbeit wurde ein Bounded Model Checking Verfahren für partielle Kripke Strukturen konzeptioniert. Das Verfahren stellt eine Weiterentwicklung der Technik aus [Weh08] dar. Während sich die ursprüngliche Technik auf Kripke Strukturen mit dreiwertiger Beschriftungsfunktion beschränkt, ist das erweiterte Verfahren für Kripke Strukturen mit dreiwertiger Beschriftungs- *und* Transitionsfunktion ausgelegt. Sowohl atomare Propositionen als auch Transitionen können auf der Modellebene den zusätzlichen Wahrheitswert *unbekannt* annehmen.

Dazu wurde zunächst das Bounded Model Checking Problem für die erweiterte Form partieller Systembeschreibungen definiert und eine Übersetzung des BMC-Problems in eine aussagenlogische Formel mit zusätzlicher Konstanten *unbekannt* entwickelt. Zudem wurde bewiesen, dass diese Übersetzung einer korrekten Reduktion auf das Erfüllbarkeitsproblem sat_3 [Weh08] entspricht und das Bounded Model Checking somit durch zwei herkömmliche Erfüllbarkeitstests realisiert werden kann.

Das entwickelte Verfahren ermöglicht die Verifikation von Systemen, deren Zustandsräume nur teilweise bekannt sind. Quellen unbekannter Eigenschaften sind beispielsweise inkonsistente Spezifikationen oder Abstraktionen, die das Systemverhalten nicht vollständig wiedergeben. Die Widersprüchlichkeit oder Unvollständigkeit solcher Beschreibungen kann sehr natürlich und präzise durch Kripke Strukturen mit dreiwertiger Beschriftungs- und Transitionsfunktion modelliert werden. Für diese partiellen Modelle lassen sich mit der neu konzeptionierten Bounded Model Checking Technik temporallogische Eigenschaften nachweisen.

Als weiterer Bestandteil dieser Arbeit erfolgte die praktische Umsetzung der Technik. Zunächst fand eine Konkretisierung des entwickelten Konzepts statt. Es wurde eine geeignete Datenstruktur zur Repräsentation der aussagenlogischen Kodierung bestimmt, welche eine effiziente Konvertierung in das SAT-Solver Eingabeformat DIMACS ermöglicht.

Zudem wurde ein algorithmisierbares Transformationsverfahren der Form *partielle Systembeschreibung → aussagenlogische Kodierung → konjunktive Normalform* entwickelt. Darauf aufbauend wurde die Bounded Model Checking Technik in Java implementiert und in die Anwendung 3Spot integriert. 3Spot ist ein Abstraktionsverfahren für parallele Systeme. Erzeugte Abstraktionen entsprechen partiellen Beschreibungen des betrachteten Systems. Die Ergänzung von 3Spot um einen Bounded Model Checker ermöglicht die Überprüfung der Gültigkeit temporallogischer Formeln anhand der abstrahierten Modelle. Dazu wird für die dreiwertigen Modelle sowie für eine LTL-Formel eine zweiwertige aussagenlogische Kodierung mit der zusätzlichen Konstanten *unbekannt* erzeugt und in die konjunktive Normalform transformiert. Der Wahrheitswert des kodierten Bounded Model Checking Problems wird nun durch zwei Erfüllbarkeitstests bestimmt, wobei alle Vorkommen der Konstanten jeweils einmal durch *true* und einmal durch *false* substituiert werden.

Es wurde also sowohl konzeptionell als auch praktisch gezeigt, dass sich die BMC-Technik aus [Weh08] auf Kripke Strukturen mit dreiwertiger Transitionsfunktion ausdehnen lässt und dabei ebenfalls mit zwei SAT-Instanzen auskommt. Die Erweiterung ermöglicht es, beim Bounded Model Checking präzisere Modelle partiell bekannter Systeme zu betrachten und somit auch genauere Aussagen bezüglich der Gültigkeit temporallogischer Eigenschaften zu erhalten.

5.2. Ausblick

Die Bounded Model Checking Technik für partielle Kripke Strukturen bietet zudem eine Reihe von Ansatzpunkten für weitergehende Forschungen. So wäre eine Verallgemeinerung des Verfahrens auf beliebige Mehrwertigkeit von Interesse. Dies würde die Möglichkeit schaffen, verschiedene Grade von Ungewissheit auf der Modellebene zu differenzieren. Die Definition eines solchen mehrwertigen Model Checking Problems findet sich beispielsweise in [CED01]. Für die in dieser Arbeit betrachtete Technik wäre dafür die Einführung weiterer Konstanten und folglich die Reduktion des BMC-Problems auf eine größere Anzahl von Erfüllbarkeitsproblemen notwendig. Ebenso bietet es sich an, das Verfahren für weitere Temporallogiken wie CTL* oder dem modalen μ-Kalkül zu definieren und somit an Ausdrucksstärke zu gewinnen.

Darüber hinaus ist das Potential vorhanden, die Technik bezüglich der Leistungsfähigkeit im Umgang mit großen Zustandsräumen zu verbessern. [CPR+02] beschäftigt sich mit platzsparenden Kodierungen von BMC-Problemen. Die dort vorgestellten Alternativen zu der ursprünglichen Kodierung aus [BCC+99] sind somit auch für eine Weiterentwicklung der Technik aus dieser Arbeit interessant. Der Erfolg des Bounded Model Checking ist insbesondere auf die Fortschritte der Solver Technologie zurückzuführen. So gilt es, die aktuellen und zukünftigen Entwicklungen in diesem angrenzenden Forschungsbereich zu verfolgen. Es existiert beispielsweise ein Ansatz, das BMC-

Problem als Quantifizierte Boolesche Formel (QBF) zu kodieren und auf spezielle QBF-Entscheidungsprozeduren zurückzugreifen [JB07]. Dies ermöglicht eine exponentiell kürzere Repräsentation der Kodierung. QBF-Solver stellen allerdings bezüglich der Effizienz, (bisher) keine echte Alternative zu SAT-Solvern dar.

Auch bietet es sich an, die Implementierung weiter zu optimieren. Die partiellen Systembeschreibungen liegen als mehrwertige Entscheidungsdiagramme vor. Diese werden zunächst aussagenlogisch kodiert und die Kodierung als reduzierter boolescher Schaltkreis (RBC) dargestellt. Es folgt eine strukturerhaltende Transformation in die konjunktive Normalform. Hierbei wird 'Top-Down' für jeden Knoten des RBC eine neue boolesche Variable eingeführt und es werden jeweils drei Klauseln generiert. Es existieren darüber hinaus eine Reihe alternativer KNF-Konvertierungsverfahren für RBCs. Die Boy de la Tour Konvertierung [Boy92] führt beispielsweise nicht für jeden Knoten eine neue Variable ein, sondern nur, falls dies die Anzahl neu produzierter Klauseln minimieren würde. Das Verfahren ist somit klauselreduziert, besitzt allerdings eine höhere Laufzeit. Ein weiterer Ansatz ist die Compact Konvertierung von [JS04], ein zu Boy de la Tour ähnliches Verfahren, welches jedoch 'Bottom-Up' vorgeht und dadurch bessere Laufzeiten erzielt. Es bleibt festzustellen, inwieweit sich durch die Umsetzung eines dieser erweiterten Konvertierungsverfahren eine höhere Effizienz beim Bounded Model Checking erreichen lässt.

Anhang A.

Beweise

Lemma 3.2.1:
Sei π ein Pfad einer partiellen Kripke Struktur M, φ eine LTL$^+$-Formel und $k \in \mathbb{N}$. Zudem sei $[\pi \models_k^i \varphi]$ definiert gemäß den Bounded LTL$^+$-Semantiken für Pfade ohne *k-Schleife. Dann gilt*

$$[\pi \models_k^i \varphi] \leq [\pi \models_{k+1}^i \varphi]$$

Beweis:
Induktion über die Struktur von φ.

- Atomare Propositionen: $\varphi = p$ (Analog für Komplemente)

$$\begin{aligned} [\pi \models_k^i p] &= L(\pi(i), p) \\ &= [\pi \models_{k+1}^i p] \end{aligned}$$

- Temporaloperator *AND*: $\varphi = \psi_1 \wedge \psi_2$

$$\begin{aligned} [\pi \models_k^i \psi_1 \wedge \psi_2] &= [\pi \models_k^i \psi_1] \wedge [\pi \models_k^i \psi_2] \\ &\leq [\pi \models_{k+1}^i \psi_1] \wedge [\pi \models_{k+1}^i \psi_2] \\ &= [\pi \models_{k+1}^i \psi_1 \wedge \psi_2] \end{aligned}$$

- Temporaloperator *OR*: $\varphi = \psi_1 \vee \psi_2$

$$\begin{aligned} [\pi \models_k^i \psi_1 \vee \psi_2] &= [\pi \models_k^i \psi_1] \vee [\pi \models_k^i \psi_2] \\ &\leq [\pi \models_{k+1}^i \psi_1] \vee [\pi \models_{k+1}^i \psi_2] \\ &= [\pi \models_{k+1}^i \psi_1 \vee \psi_2] \end{aligned}$$

- Temporaloperator *NEXT*: $\varphi = \mathbf{X}\psi$

$$\begin{aligned}
[\pi \models_k^i \mathbf{X}\psi] &= R(\pi(i), \pi(i+1)) \wedge [\pi \models_k^{i+1} \psi] \\
&\leq R(\pi(i), \pi(i+1)) \wedge [\pi \models_{k+1}^{i+1} \psi] \\
&= [\pi \models_{k+1}^{i} \mathbf{X}\psi]
\end{aligned}$$

- Temporaloperator *GLOBALLY*: $\varphi = \mathbf{G}\psi$

$$\begin{aligned}
[\pi \models_k^i \mathbf{G}\psi] &= false \\
&= [\pi \models_{k+1}^{i} \mathbf{G}\psi]
\end{aligned}$$

- Temporaloperator *EVENTUALLY*: $\varphi = \mathbf{F}\psi$

$$\begin{aligned}
[\pi \models_k^i \mathbf{F}\psi] &= \bigvee_{j=i}^{k} [\bigwedge_{n=i}^{j-1} R(\pi(n), \pi(n+1)) \wedge [\pi \models_k^j \psi]] \\
&\leq \bigvee_{j=i}^{k+1} [\bigwedge_{n=i}^{j-1} R(\pi(n), \pi(n+1)) \wedge [\pi \models_{k+1}^j \psi]] \\
&= [\pi \models_{k+1}^{i} \mathbf{F}\psi]
\end{aligned}$$

- Temporaloperator *UNTIL*: $\varphi = \psi_1 \mathbf{U} \psi_2$

$$\begin{aligned}
[\pi \models_k^i \psi_1 \mathbf{U} \psi_2] &= \bigvee_{j=i}^{k} [\bigwedge_{n=i}^{j-1} ([\pi \models_k^n \psi_1] \wedge R(\pi(n), \pi(n+1))) \wedge [\pi \models_k^j \psi_2]] \\
&\leq \bigvee_{j=i}^{k+1} [\bigwedge_{n=i}^{j-1} ([\pi \models_{k+1}^n \psi_1] \wedge R(\pi(n), \pi(n+1))) \wedge [\pi \models_{k+1}^j \psi_2]] \\
&= [\pi \models_{k+1}^{i} \psi_1 \mathbf{U} \psi_2]
\end{aligned}$$

□

Lemma 3.3.1:
Sei π ein Pfad einer partiellen Kripke Struktur M, φ eine LTL$^+$-Formel und $k \in \mathbb{N}$. Zudem sei $[\pi \models_k^i \varphi]$ definiert gemäß den Bounded LTL$^+$-Semantiken für Pfade ohne *k-Schleife. Dann gilt*

$$[\pi \models_k^i \varphi] \geq I_\pi([\![M]\!]_k \wedge [\![\varphi]\!]_k^i)$$

Beweis:
Induktion über die Struktur von φ.

- Atomare Propositionen: $\varphi = p$ (Analog für Komplemente)

$$
\begin{aligned}
[\pi \models_k^i p] &= L(\pi(i), p) \\
I_\pi([\![M]\!]_k \wedge [\![p]\!]_k^i) &= I_\pi([\![M]\!]_k \wedge \bigvee_{s \in S} c(s)_i \wedge L(s, p)) \\
&= I_\pi([\![M]\!]_k) \wedge L(s_i, p) \\
&= I_\pi([\![M]\!]_k) \wedge L(\pi(i), p) \\
&\leq L(\pi(i), p)
\end{aligned}
$$

- Temporaloperator AND: $\varphi = \psi_1 \wedge \psi_2$

$$
\begin{aligned}
[\pi \models_k^i \psi_1 \wedge \psi_2] &= [\pi \models_k^i \psi_1] \wedge [\pi \models_k^i \psi_2] \\
&\geq I_\pi([\![M]\!]_k \wedge [\![\psi_1]\!]_k^i) \wedge I_\pi([\![M]\!]_k \wedge [\![\psi_2]\!]_k^i) \\
&= I_\pi([\![M]\!]_k \wedge [\![\psi_1 \wedge \psi_2]\!]_k^i)
\end{aligned}
$$

- Temporaloperator OR: $\varphi = \psi_1 \vee \psi_2$

$$
\begin{aligned}
[\pi \models_k^i \psi_1 \vee \psi_2] &= [\pi \models_k^i \psi_1] \vee [\pi \models_k^i \psi_2] \\
&\geq I_\pi([\![M]\!]_k \wedge [\![\psi_1]\!]_k^i) \vee I_\pi([\![M]\!]_k \wedge [\![\psi_2]\!]_k^i) \\
&= I_\pi([\![M]\!]_k \wedge [\![\psi_1 \vee \psi_2]\!]_k^i)
\end{aligned}
$$

- Temporaloperator $NEXT$: $\varphi = \mathbf{X}\psi$

$$
\begin{aligned}
[\pi \models_k^i \mathbf{X}\psi] &= R(\pi(i), \pi(i+1)) \wedge [\pi \models_k^{i+1} \psi] \\
&\geq R(\pi(i), \pi(i+1)) \wedge I_\pi([\![M]\!]_k \wedge [\![\psi]\!]_k^{i+1}) \\
&= I_\pi(T_{i,i+1} \wedge [\![M]\!]_k \wedge [\![\mathbf{X}\psi]\!]_k^i) \\
&= I_\pi([\![M]\!]_k \wedge [\![\mathbf{X}\psi]\!]_k^i)
\end{aligned}
$$

- Temporaloperator $GLOBALLY$: $\varphi = \mathbf{G}\psi$

$$
\begin{aligned}
[\pi \models_k^i \mathbf{G}\psi] &= false \\
&= I_\pi([\![M]\!]_k \wedge [\![\mathbf{G}\psi]\!]_k^i)
\end{aligned}
$$

- Temporaloperator $EVENTUALLY$: $\varphi = \mathbf{F}\psi$

$$
\begin{aligned}
[\pi \models_k^i \mathbf{F}\psi] &= \bigvee_{j=i}^{k} [\bigwedge_{n=i}^{j-1} R(\pi(n), \pi(n+1)) \wedge [\pi \models_k^j \psi]] \\
&\geq \bigvee_{j=i}^{k} [\bigwedge_{n=i}^{j-1} R(\pi(n), \pi(n+1)) \wedge I_\pi([\![M]\!]_k \wedge [\![\psi]\!]_k^j)]
\end{aligned}
$$

$$
\begin{aligned}
&= I_\pi(\bigvee_{j=i}^{k}[\bigwedge_{n=i}^{j-1} T_{n,n+1} \wedge [\![M]\!]_k \wedge [\![\psi]\!]_k^j]) \\
&= I_\pi(\bigvee_{j=i}^{k}[[\![M]\!]_k \wedge [\![\psi]\!]_k^j]) \\
&= I_\pi([\![M]\!]_k \wedge \bigvee_{j=i}^{k}[\![\psi]\!]_k^j) \\
&= I_\pi([\![M]\!]_k \wedge [\![\mathbf{F}\psi]\!]_k^i)
\end{aligned}
$$

- Temporaloperator *UNTIL*: $\varphi = \psi_1 \mathbf{U} \psi_2$

$$
\begin{aligned}
[\pi \models_k^i \psi_1 \mathbf{U} \psi_2] &= \bigvee_{j=i}^{k}[\bigwedge_{n=i}^{j-1}([\pi \models_k^n \psi_1] \wedge R(\pi(n), \pi(n+1))) \wedge [\pi \models_k^j \psi_2]] \\
&\geq \bigvee_{j=i}^{k}[\bigwedge_{n=i}^{j-1}(I_\pi([\![M]\!]_k \wedge [\![\psi_1]\!]_k^n) \wedge R(\pi(n), \pi(n+1))) \wedge I_\pi([\![M]\!]_k \wedge [\![\psi_2]\!]_k^j)] \\
&= I_\pi(\bigvee_{j=i}^{k}[\bigwedge_{n=i}^{j-1}([\![M]\!]_k \wedge [\![\psi_1]\!]_k^n \wedge T_{n,n+1}) \wedge [\![M]\!]_k \wedge [\![\psi_2]\!]_k^j]) \\
&= I_\pi(\bigvee_{j=i}^{k}[\bigwedge_{n=i}^{j-1}([\![M]\!]_k \wedge [\![\psi_1]\!]_k^n) \wedge [\![M]\!]_k \wedge [\![\psi_2]\!]_k^j]) \\
&= I_\pi([\![M]\!]_k \wedge \bigvee_{j=i}^{k}[\bigwedge_{n=i}^{j-1}[\![\psi_1]\!]_k^n \wedge [\![\psi_2]\!]_k^j]) \\
&= I_\pi([\![M]\!]_k \wedge [\![\psi_1 U \psi_2]\!]_k^i)
\end{aligned}
$$

□

Lemma 3.3.2:
Sei π ein Pfad einer partiellen Kripke Struktur M, φ eine LTL$^+$-Formel, $v \in \{true, \bot, false\}$ und $k \in \mathbb{N}$. Zudem sei $[\pi \models_k^i \varphi]$ definiert gemäß den Bounded LTL$^+$-Semantiken für Pfade ohne *k-Schleife. Dann gilt*

$$[\pi \models_k^i \varphi] \geq v \quad \Rightarrow \quad \exists j, i \leq j \leq k : [\pi \models_j^i \varphi] \geq v \wedge \forall n, i \leq n < j : R(\pi(n), \pi(n+1)) \geq v$$

Beweis:
Induktion über die Struktur von φ.

- Atomare Propositionen: $\varphi = p$ (Analog für Komplemente)

$$
\begin{aligned}
[\pi \models_k^i p] \geq v \quad &\Rightarrow \quad L(\pi(i), p) \geq v \\
&\Rightarrow \quad [\pi \models_i^i p] \geq v
\end{aligned}
$$

- Temporaloperator *AND*: $\varphi = \psi_1 \wedge \psi_2$

$$[\pi \models_k^i \psi_1 \wedge \psi_2] \geq v \quad \Rightarrow \quad [\pi \models_k^i \psi_1] \geq v \wedge [\pi \models_k^i \psi_2] \geq v$$

- Temporaloperator *OR*: $\varphi = \psi_1 \vee \psi_2$

$$[\pi \models_k^i \psi_1 \vee \psi_2] \geq v \quad \Rightarrow \quad [\pi \models_k^i \psi_1] \geq v \vee [\pi \models_k^i \psi_2] \geq v$$

- Temporaloperator *NEXT*: $\varphi = \mathbf{X}\psi$

$$[\pi \models_k^i \mathbf{X}\psi] \geq v \quad \Rightarrow \quad [\pi \models_k^{i+1} \psi] \geq v \wedge R(\pi(i), \pi(i+1)) \geq v$$

- Temporaloperator *GLOBALLY*: $\varphi = \mathbf{G}\psi$

$$\begin{aligned} [\pi \models_k^i \mathbf{G}\psi] \geq v \quad &\Rightarrow \quad v = false \\ &\Rightarrow \quad [\pi \models_i^i \mathbf{G}\psi] \geq v \end{aligned}$$

- Temporaloperator *EVENTUALLY*: $\varphi = \mathbf{F}\psi$

$$\begin{aligned} [\pi \models_k^i \mathbf{F}\psi] \geq v \quad &\Rightarrow \quad \exists j, i \leq j \leq k : [\pi \models_k^j \psi] \geq v \\ &\quad \wedge \forall n, i \leq n < j : R(\pi(n), \pi(n+1)) \geq v \end{aligned}$$

- Temporaloperator *UNTIL*: $\varphi = \psi_1 \mathbf{U} \psi_2$

$$\begin{aligned} [\pi \models_k^i \psi_1 \mathbf{U} \psi_2] \geq v \quad &\Rightarrow \quad \exists j, i \leq j \leq k : [\pi \models_k^j \psi_2] \geq v \\ &\quad \wedge \forall n, i \leq n < j : R(\pi(n), \pi(n+1)) \geq v \wedge [\pi \models_k^n \psi_1] \geq v \end{aligned}$$

□

Lemma 3.3.3:
Sei π ein Pfad einer partiellen Kripke Struktur M, φ eine LTL$^+$-Formel und $k \in \mathbb{N}$. Zudem sei $[\pi \models_k^i \varphi]$ definiert gemäß den Bounded LTL$^+$-Semantiken für Pfade ohne *k-Schleife. Dann gilt*

$$[\pi \models_k^i \varphi] \leq I_\pi([\![\varphi]\!]_k^i)$$

Beweis:
Induktion über die Struktur von φ.

- Atomare Propositionen: $\varphi = p$ (Analog für Komplemente)

$$
\begin{array}{lcl}
[\pi \models_k^i p] & = & L(\pi(i), p) \\
I_\pi([\![p]\!]_k^i) & = & I_\pi(\bigvee_{s \in S} c(s)_i \wedge L(s, p)) \\
 & = & L(s_i, p) \\
 & = & L(\pi(i), p)
\end{array}
$$

- Temporaloperator AND: $\varphi = \psi_1 \wedge \psi_2$

$$
\begin{array}{lcl}
[\pi \models_k^i \psi_1 \wedge \psi_2] & = & [\pi \models_k^i \psi_1] \wedge [\pi \models_k^i \psi_2] \\
 & \leq & I_\pi([\![\psi_1]\!]_k^i \wedge [\![\psi_2]\!]_k^i) \\
 & = & I_\pi([\![\psi_1 \wedge \psi_2]\!]_k^i)
\end{array}
$$

- Temporaloperator OR: $\varphi = \psi_1 \vee \psi_2$

$$
\begin{array}{lcl}
[\pi \models_k^i \psi_1 \vee \psi_2] & = & [\pi \models_k^i \psi_1] \vee [\pi \models_k^i \psi_2] \\
 & \leq & I_\pi([\![\psi_1]\!]_k^i \vee [\![\psi_2]\!]_k^i) \\
 & = & I_\pi([\![\psi_1 \vee \psi_2]\!]_k^i)
\end{array}
$$

- Temporaloperator $NEXT$: $\varphi = \mathbf{X}\psi$

$$
\begin{array}{lcl}
[\pi \models_k^i \mathbf{X}\psi] & = & R(\pi(i), \pi(i+1)) \wedge [\pi \models_k^{i+1} \psi] \\
 & \leq & R(\pi(i), \pi(i+1)) \wedge I_\pi([\![\psi]\!]_k^{i+1}) \\
 & = & R(\pi(i), \pi(i+1)) \wedge I_\pi([\![\mathbf{X}\psi]\!]_k^i) \\
 & \leq & I_\pi([\![\mathbf{X}\psi]\!]_k^i)
\end{array}
$$

- Temporaloperator $GLOBALLY$: $\varphi = \mathbf{G}\psi$

$$
\begin{array}{lcl}
[\pi \models_k^i \mathbf{G}\psi] & = & false \\
 & = & I_\pi([\![\mathbf{G}\psi]\!]_k^i)
\end{array}
$$

- Temporaloperator $EVENTUALLY$: $\varphi = \mathbf{F}\psi$

$$
\begin{array}{lcl}
[\pi \models_k^i \mathbf{F}\psi] & = & \bigvee_{j=i}^{k} [\bigwedge_{n=i}^{j-1} R(\pi(n), \pi(n+1)) \wedge [\pi \models_k^j \psi]] \\
 & \leq & I_\pi(\bigvee_{j=i}^{k} [\bigwedge_{n=i}^{j-1} R(\pi(n), \pi(n+1)) \wedge [\![\psi]\!]_k^j]) \\
 & \leq & I_\pi(\bigvee_{j=i}^{k} [\![\psi]\!]_k^j) \\
 & = & I_\pi([\![\mathbf{F}\psi]\!]_k^i)
\end{array}
$$

- Temporaloperator *UNTIL*: $\varphi = \psi_1 \mathbf{U} \psi_2$

$$\begin{aligned}
[\pi \models_k^i \psi_1 \mathbf{U} \psi_2] &= \bigvee_{j=i}^{k} [\bigwedge_{n=i}^{j-1} ([\pi \models_k^n \psi_1] \wedge R(\pi(n), \pi(n+1))) \wedge [\pi \models_k^j \psi_2]] \\
&\leq I_\pi(\bigvee_{j=i}^{k} [\bigwedge_{n=i}^{j-1} ([\![\psi_1]\!]_k^n \wedge R(\pi(n), \pi(n+1))) \wedge [\![\psi_2]\!]_k^j]) \\
&\leq I_\pi(\bigvee_{j=i}^{k} [\bigwedge_{n=i}^{j-1} [\![\psi_1]\!]_k^n \wedge [\![\psi_2]\!]_k^j]) \\
&= I_\pi([\![\psi_1 U \psi_2]\!]_k^i)
\end{aligned}$$

□

Lemma 3.3.4:
Sei π ein Pfad einer partiellen Kripke Struktur M, φ eine LTL$^+$-Formel und $k \in \mathbb{N}$. Dann gilt

$$[\pi \models_k^i \varphi] \geq \bigvee_{l=0}^{k} I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\varphi]\!]_k^i)$$

Beweis:
Es sind zwei Fälle zu unterscheiden.

1. π besitzt *keine* k-Schleife.
   $\Rightarrow \bigvee_{l=0}^{k} I_\pi({}_lL_k) = false$
   $\Rightarrow \bigvee_{l=0}^{k} I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\varphi]\!]_k^i) = false$
   $\Rightarrow [\pi \models_k^i \varphi] \geq \bigvee_{l=0}^{k} I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\varphi]\!]_k^i)$

2. π besitzt eine k-Schleife.
   $\Rightarrow \exists l, 0 \leq l \leq k : I_\pi({}_lL_k) \in \{true, \bot\} \wedge \forall l', l' \neq l : I_\pi({}_{l'}L_k) = false$
   Sei $l \in \mathbb{N}$ der Index für den π eine (k,l)-Schleife besitzt. Es bleibt noch zu zeigen

$$[\pi \models_k^i \varphi] \geq I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\varphi]\!]_k^i)$$

   Induktion über die Struktur von φ. Da π eine (k,l)-Schleife besitzt gilt $\forall j \geq l$: $[\pi \models_k^j \varphi] = [\pi \models_k^{j+(k-l)+1} \varphi]$ und $\pi(j) = \pi(j + (k-l) + 1)$.

   - Atomare Propositionen: $\varphi = p$ (Analog für Komplemente)

$$\begin{aligned}
[\pi \models_k^i p] &= L(\pi(i), p) \\
I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![p]\!]_k^i) &= I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge \bigvee_{s \in S} c(s)_i \wedge L(s,p)) \\
&= I_\pi({}_lL_k \wedge [\![M]\!]_k) \wedge L(\pi(i), p) \\
&\leq L(\pi(i), p)
\end{aligned}$$

- Temporaloperator *AND*: $\varphi = \psi_1 \wedge \psi_2$

$$\begin{aligned}
[\pi \models_k^i \psi_1 \wedge \psi_2] &= [\pi \models_k^i \psi_1] \wedge [\pi \models_k^i \psi_2] \\
&\geq I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi_1]\!]_k^i) \wedge I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi_2]\!]_k^i) \\
&= I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi_1 \wedge \psi_2]\!]_k^i)
\end{aligned}$$

- Temporaloperator *OR*: $\varphi = \psi_1 \vee \psi_2$

$$\begin{aligned}
[\pi \models_k^i \psi_1 \vee \psi_2] &= [\pi \models_k^i \psi_1] \vee [\pi \models_k^i \psi_2] \\
&\geq I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi_1]\!]_k^i) \vee I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi_2]\!]_k^i) \\
&= I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi_1 \vee \psi_2]\!]_k^i)
\end{aligned}$$

- Temporaloperator *NEXT*: $\varphi = \mathbf{X}\psi$

$$\begin{aligned}
[\pi \models_k^i \mathbf{X}\psi] &= R(\pi(i), \pi(i+1)) \wedge [\pi \models_k^{i+1} \psi] \\
&\geq R(\pi(i), \pi(i+1)) \wedge I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi]\!]_k^{succ(i)}) \\
&= I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi]\!]_k^{succ(i)}) \\
&= I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\mathbf{X}\psi]\!]_k^i)
\end{aligned}$$

- Temporaloperator *GLOBALLY*: $\varphi = \mathbf{G}\psi$

$$\begin{aligned}
[\pi \models_k^i \mathbf{G}\psi] &= \textstyle\bigwedge_{j \in \mathbb{N},\ j \geq i} ([\pi \models_k^j \psi] \wedge R(\pi(j), \pi(j+1))) \\
&\geq \textstyle\bigwedge_{j \in \mathbb{N},\ j \geq i} (I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi]\!]_k^j) \wedge R(\pi(j), \pi(j+1))) \\
&= \textstyle\bigwedge_{j \in \mathbb{N},\ j \geq i} I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi]\!]_k^j) \\
&= \textstyle\bigwedge_{j=min(i,l)}^{k} I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi]\!]_k^j) \\
&= I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge \textstyle\bigwedge_{j=min(i,l)}^{k} {}_l[\![\psi]\!]_k^j) \\
&= I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\mathbf{G}\psi]\!]_k^i)
\end{aligned}$$

- Temporaloperator *EVENTUALLY*: $\varphi = \mathbf{F}\psi$

$$\begin{aligned}
[\pi \models_k^i \mathbf{F}\psi] &= \textstyle\bigvee_{j \in \mathbb{N},\ j \geq i} ([\pi \models_k^j \psi] \wedge \bigwedge_{n=i}^{j-1} R(\pi(n), \pi(n+1))) \\
&\geq \textstyle\bigvee_{j \in \mathbb{N},\ j \geq i} (I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi]\!]_k^j) \wedge \bigwedge_{n=i}^{j-1} R(\pi(n), \pi(n+1))) \\
&= \textstyle\bigvee_{j \in \mathbb{N},\ j \geq i} I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi]\!]_k^j) \\
&= \textstyle\bigvee_{j=min(i,l)}^{k} I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\psi]\!]_k^j) \\
&= I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge \textstyle\bigvee_{j=min(i,l)}^{k} {}_l[\![\psi]\!]_k^j) \\
&= I_\pi({}_lL_k \wedge [\![M]\!]_k \wedge_l [\![\mathbf{F}\psi]\!]_k^i)
\end{aligned}$$

- Temporaloperator *UNTIL*: $\varphi = \psi_1 \mathbf{U} \psi_2$

$$
\begin{aligned}
[\pi \models_k^i \psi_1 \mathbf{U} \psi_2] &= \bigvee_{j\in\mathbb{N},\ j\geq i}([\pi \models_k^j \psi_2] \wedge \bigwedge_{n=i}^{j-1}([\pi \models_k^n \psi_1] \wedge R(\pi(n), \pi(n+1)))) \\
&\geq \bigvee_{j\in\mathbb{N},\ j\geq i}(I_\pi({}_l L_k \wedge [\![M]\!]_k \wedge_l [\![\psi_2]\!]_k^j) \\
&\quad \wedge \bigwedge_{n=i}^{j-1}(I_\pi({}_l L_k \wedge [\![M]\!]_k \wedge_l [\![\psi_1]\!]_k^n) \wedge R(\pi(n), \pi(n+1)))) \\
&= I_\pi({}_l L_k \wedge [\![M]\!]_k) \wedge \bigvee_{j\in\mathbb{N},\ j\geq i}(I_\pi({}_l [\![\psi_2]\!]_k^j) \wedge \bigwedge_{n=i}^{j-1} I_\pi({}_l [\![\psi_1]\!]_k^n)) \\
&= I_\pi({}_l L_k \wedge [\![M]\!]_k) \wedge \bigvee_{j=i}^{k}(I_\pi({}_l [\![\psi_2]\!]_k^j) \wedge \bigwedge_{n=i}^{j-1} I_\pi({}_l [\![\psi_1]\!]_k^n)) \\
&\quad \vee \bigvee_{j=l}^{i-1}(I_\pi({}_l [\![\psi_2]\!]_k^j) \wedge \bigwedge_{n=i}^{k} I_\pi({}_l [\![\psi_1]\!]_k^n) \wedge \bigwedge_{n=i}^{k} I_\pi({}_l [\![\psi_1]\!]_k^n)) \\
&= I_\pi({}_l L_k \wedge [\![M]\!]_k \wedge_l [\![\psi_1 \mathbf{U} \psi_2]\!]_k^i)
\end{aligned}
$$

□

Lemma 3.3.5:
Sei π ein Pfad einer partiellen Kripke Struktur M und φ eine LTL$^+$-Formel. Zudem seien $l, k \in \mathbb{N}$ mit $l \leq k$ und π besitzt eine (k,l)-Schleife. Dann gilt

$$[\pi \models_k^i \varphi] \leq I_\pi({}_l [\![\varphi]\!]_k^i)$$

Beweis:
Induktion über die Struktur von φ. Da π eine (k,l)-Schleife besitzt gilt $\forall j \geq l : [\pi \models_k^j \varphi] = [\pi \models_k^{j+(k-l)+1} \varphi]$ und $\pi(j) = \pi(j + (k-l) + 1)$.

- Atomare Propositionen: $\varphi = p$ (Analog für Komplemente)

$$
\begin{aligned}
[\pi \models_k^i p] &= L(\pi(i), p) \\
I_\pi({}_l [\![p]\!]_k^i) &= I_\pi(\bigvee_{s\in S} c(s)_i \wedge L(s,p)) \\
&= L(s_i, p) \\
&= L(\pi(i), p)
\end{aligned}
$$

- Temporaloperator *AND*: $\varphi = \psi_1 \wedge \psi_2$

$$
\begin{aligned}
[\pi \models_k^i \psi_1 \wedge \psi_2] &= [\pi \models_k^i \psi_1] \wedge [\pi \models_k^i \psi_2] \\
&\leq I_\pi({}_l [\![\psi_1]\!]_k^i \wedge_l [\![\psi_2]\!]_k^i) \\
&= I_\pi({}_l [\![\psi_1 \wedge \psi_2]\!]_k^i)
\end{aligned}
$$

- Temporaloperator OR: $\varphi = \psi_1 \vee \psi_2$

$$\begin{aligned}
[\pi \models_k^i \psi_1 \vee \psi_2] &= [\pi \models_k^i \psi_1] \vee [\pi \models_k^i \psi_2] \\
&\leq I_\pi({}_l[\![\psi_1]\!]_k^i \vee_l [\![\psi_2]\!]_k^i) \\
&= I_\pi({}_l[\![\psi_1 \vee \psi_2]\!]_k^i)
\end{aligned}$$

- Temporaloperator $NEXT$: $\varphi = \mathbf{X}\psi$

$$\begin{aligned}
[\pi \models_k^i \mathbf{X}\psi] &= R(\pi(i), \pi(i+1)) \wedge [\pi \models_k^{i+1} \psi] \\
&\leq [\pi \models_k^{i+1} \psi] \\
&\leq I_\pi({}_l[\![\psi]\!]_k^{succ(i)}) \\
&= I_\pi({}_l[\![\mathbf{X}\psi]\!]_k^i)
\end{aligned}$$

- Temporaloperator $GLOBALLY$: $\varphi = \mathbf{G}\psi$

$$\begin{aligned}
[\pi \models_k^i \mathbf{G}\psi] &= \textstyle\bigwedge_{j \in \mathbb{N},\ j \geq i} ([\pi \models_k^j \psi] \wedge R(\pi(j), \pi(j+1))) \\
&\leq \textstyle\bigwedge_{j \in \mathbb{N},\ j \geq i} [\pi \models_k^j \psi] \\
&= \textstyle\bigwedge_{j=min(i,l)}^{k} [\pi \models_k^j \psi] \\
&\leq \textstyle\bigwedge_{j=min(i,l)}^{k} I_\pi({}_l[\![\psi]\!]_k^j) \\
&= I_\pi(\textstyle\bigwedge_{j=min(i,l)}^{k} {}_l[\![\psi]\!]_k^j) \\
&= I_\pi({}_l[\![\mathbf{G}\psi]\!]_k^i)
\end{aligned}$$

- Temporaloperator $EVENTUALLY$: $\varphi = \mathbf{F}\psi$

$$\begin{aligned}
[\pi \models_k^i \mathbf{F}\psi] &= \textstyle\bigvee_{j \in \mathbb{N},\ j \geq i} ([\pi \models_k^j \psi] \wedge \bigwedge_{n=i}^{j-1} R(\pi(n), \pi(n+1))) \\
&\leq \textstyle\bigvee_{j \in \mathbb{N},\ j \geq i} [\pi \models_k^j \psi] \\
&= \textstyle\bigvee_{j=min(i,l)}^{k} [\pi \models_k^j \psi] \\
&\leq \textstyle\bigvee_{j=min(i,l)}^{k} I_\pi({}_l[\![\psi]\!]_k^j) \\
&= I_\pi(\textstyle\bigvee_{j=min(i,l)}^{k} {}_l[\![\psi]\!]_k^j) \\
&= I_\pi({}_l[\![\mathbf{F}\psi]\!]_k^i)
\end{aligned}$$

- Temporaloperator $UNTIL$: $\varphi = \psi_1 \mathbf{U} \psi_2$

$$
\begin{aligned}
[\pi \models^i_k \psi_1 \mathbf{U} \psi_2] &= \bigvee_{j\in\mathbb{N},\ j\geq i}([\pi \models^j_k \psi_2] \wedge \bigwedge_{n=i}^{j-1}([\pi \models^n_k \psi_1] \wedge R(\pi(n), \pi(n+1)))) \\
&\leq \bigvee_{j\in\mathbb{N},\ j\geq i}([\pi \models^j_k \psi_2] \wedge \bigwedge_{n=i}^{j-1}[\pi \models^n_k \psi_1]) \\
&= \bigvee_{j=i}^{k}([\pi \models^j_k \psi_2] \wedge \bigwedge_{n=i}^{j-1}[\pi \models^n_k \psi_1]) \\
&\quad \vee \bigvee_{j=l}^{i-1}([\pi \models^j_k \psi_2] \wedge \bigwedge_{n=i}^{k}[\pi \models^n_k \psi_1] \wedge \bigwedge_{n=i}^{k}[\pi \models^n_k \psi_1]) \\
&\leq \bigvee_{j=i}^{k}(I_\pi({}_l[\![\psi_2]\!]^j_k) \wedge \bigwedge_{n=i}^{j-1} I_\pi({}_l[\![\psi_1]\!]^n_k)) \\
&\quad \vee \bigvee_{j=l}^{i-1}(I_\pi({}_l[\![\psi_2]\!]^j_k) \wedge \bigwedge_{n=i}^{k} I_\pi({}_l[\![\psi_1]\!]^n_k) \wedge \bigwedge_{n=i}^{k} I_\pi({}_l[\![\psi_1]\!]^n_k)) \\
&= I_\pi({}_l[\![\psi_1 \mathbf{U} \psi_2]\!]^i_k)
\end{aligned}
$$

□

Anhang B.

Implementierung

Die Implementierung sowie eine Kurzanleitung sind auf CD beigefügt.

Literaturverzeichnis

[ABE00] Abdulla, P.A., Bjesse, P., Een, N.: *Symbolic Reachability Analysis Based on SAT-Solvers.* In: TACAS '00: Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems. LNCS, vol. 1785, pp. 411-425, Springer (2000).

[AY04] Andrade, J.O., Yonezawa, T.: *Multi-valued bounded model checking.* In: Technical report, Department of Computer Science, University of Tsukuba, Japan (2004).

[BCC+99] Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y.: *Symbolic Model Checking without BDDs.* In: Cleaveland,W.R. (ed.) TACAS 1999. LNCS, vol. 1579, pp. 193-207. Springer, Heidelberg (1999).

[BCC+03] Biere, A., Cimatti, A., Clarke, E.M., Strichman, O., Zhu, Y.: *Bounded Model Checking.* In: Advances in Computers, Vol. 58:118-149, Academic Press (2003).

[BCM+92] Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, L.J.: *Symbolic Model Checking: 1020 States and Beyond.* In: Inf. Comput. 98(2), 142-170 (1992).

[BG99] Bruns, G., Godefroid, P.: *Model checking partial state spaces with 3- valued temporal logics.* In: Halbwachs, N., Peled, D.A. (eds.) CAV 1999. LNCS, vol. 1633, pp. 274-287. Springer, Heidelberg (1999).

[BG00] Bruns, G., Godefroid, P.: *Generalized Model Checking: Reasoning about Partial State Spaces.* In: Palamidessi, C. (ed.) CONCUR 2000. LNCS, Vol. 1877, pp. 168-182. Springer, Heidelberg (2000).

[BHJ+07] Beyer, D., Henzinger, T.A., Jhala, R., Majumdar, R.: *The Software Model Checker Blast: Applications to Software Engineering.* In: International Journal on Software Tools for Technology Transfer (STTT), 9(5-6):505-525, (2007).

[Boy92] Boy de la Tour, T.: *An Optimality Result for Clause Form Translation.* In: Journal of Symbolic Computation, 14(4): 283-302, (1992).

[CCG+02] Cimatti, A., Clarke, E.M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: *NuSMV 2: An OpenSource Tool for Symbolic Model Checking.* In: E. Brinksma and K. G. Larsen, editors, Proceedings of Computer Aided Verification: 14th International Conference (CAV 2002), volume 2404 of Lecture Notes in Computer Science, pages 359-364. Springer (2002).

[CDE01] Chechik, M., Devereux, B., Easterbrook, S.M.: *Implementing a Multi-valued Symbolic Model Checker.* In: Proceedings of TACAS'01, pages 404-419. Springer (2001).

[CDE+03] Chechik, M., Devereux, B., Easterbrook, S.M., Gurfinkel, A.: *Multi-valued symbolic model-checking.* In: ACM Trans. Softw. Eng. Methodol. 12(4), 371-408 (2003).

[CED01] Chechik, M., Easterbrook, S.M., Devereux, B.: *Model checking with multi-valued temporal logics.* In: ISMVL, pp. 187-192 (2001).

[CGJ+01] Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: *Progress on the State Explosion Problem in Model Checking.* In: Informatics, 10 Years Back, 10 Years Ahead, volume 2000 of LNCS, pages 176-194 (2001).

[CGL92] Clarke, E., Grumberg, O., Long, D.E.: *Model Checking and abtraction.* In: 19th ACM POPL (1992).

[CGP99] Clarke, E., Grumberg, O., Peled, D.: *Model checking.* In: MIT Press, Cambridge (1999).

[CPR+02] Clarke, E., Grumberg, O., Peled, D.: *Improving the Encoding of LTL Model Checking into SAT.* In: VMCAI '02: Revised Papers from the Third International Workshop on Verification, Model Checking, and Abstract Interpretation, pp. 196-207. Springer (2002).

[Dim93] *DIMACS: Satisability Suggested Format.* ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/ (1993).

[DLL62] Davis, M., Logemann, G., Loveland, D.: *A machine program for theorem-proving.* In: Commun. ACM, vol. 5, pp. 394-397. ACM, New York (1962).

[ES05] Een, N., Sörensson, N.: *MiniSat – A SAT Solver with Conflict-Clause Minimization.* In: Proc. Theory and Applications of Satisfiability Testing (SAT'05) (2005).

[Fit94] Fitting, M.: *Kleene's three valued logics and their children.* In: Fundam. Inform. 20(1/2/3), 113-131 (1994).

[GC03] Gurfinkel, A., Chechik, M.: *Multi-valued model checking via classical model checking.* In: Amadio, R., Lugiez, D. (eds.) CONCUR 2003. LNCS, vol. 2761, pp. 263-277. Springer, Heidelberg (2003).

[GC06] Gurfinkel, A., Chechik, M.: *Why waste a perfectly good abstraction?* In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006. LNCS, vol. 3920, pp. 212-226. Springer, Heidelberg (2006).

[Hol97] Holzmann, G.J.: *The Model Checker SPIN.* In: IEEE Trans. Softw. Eng. vol. 23, pp. 279-295. IEEE Press, Piscataway, NJ, USA (1997).

[INH96] Iwashita, H., Nakata, T., Hirose, F.: *CTL model checking based on forward state traversal.* In: ICCAD '96: Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design. pp. 82-87. IEEE Computer Society, San Jose (1996).

[JB07] Jussila, T., Biere, A.: *Compressing BMC Encodings with QBF.* In: Electron. Notes Theor. Comput. Sci. vol. 174, pp. 45-56. Elsevier Science Publishers B. V., Amsterdam (2007).

[JS04] Jackson, P., Sheridan, D.: *The Optimality of a Fast CNF Conversion and its Use with SAT.* In: Technical Report APES-82-2002, APES Research Group, Mar. 2004. Available from http://www.dcs.st-and.ac.uk/ apes/apes-reports.html. (2004).

[Kai92] Kaivola, R.: *Compositional Model Checking for Linear-Time Temporal Logic.* In: CAV '92: Proceedings of the Fourth International Workshop on Computer Aided Verification. LNCS, vol. 663, pp. 248-259. Springer, London (1992).

[KM07] Kramer, J., Magee, J.: *Self-Managed Systems: an Architectural Challenge.* In: ICSE 2007 - Future of Software Engineering Track. ACM Press, New York (2007).

[KP02] Konikowska, B., Penczek, W.: *Reducing Model Checking from Multivalued CTL* to CTL*.* In: Brim, L., Jancar, P., Kretinsky, M., Kucera, A. (eds.) CONCUR 2002. LNCS, vol. 2421, pp. 226-239. Springer, Heidelberg (2002).

[LKM03] Liu, C., Kuehlmann, A., Moskewicz, M.: *CAMA: A Multi-Valued Satisfiability Solver.* In: International Conference on Computer Aided Design, pp. 326-333. IEEE/ACM (2003).

[MMZ+01] Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: *Chaff: Engineering an Efficient SAT Solver.* In: DAC, pp. 530-535. ACM, New York (2001).

[MP92] Manna, Z., Pnueli, A.: *The Temporal Logic of Reactive and Concurrent Systems: Specification.* In: Springer, Heidelberg (1992).

[PBG05] Prasad, M.R., Biere, A., Gupta, A.: *A survey of recent advances in SATbased formal verification.* In: STTT 7(2), 156-173 (2005).

[Rob65] Robinson, J. A.: *A Machine-Oriented Logic Based on the Resolution Principle.* In: J. ACM, vol. 12, pp. 23-41. ACM, New York (1965).

[She06] Sheridan, D.: *Temporal Logic Encodings for SAT-based Bounded Model Checking.* In: PhD thesis, University of Edinburgh, College of Science and Engineering, School of Informatics (2006).

[SWW09] Schrieb, J., Wehrheim, H., Wonisch, D.: *Three-valued spotlight abstractions.* Unpublished (2009).

[Weh08] Wehrheim, H.: *Bounded Model Checking for Partial Kripke Structures.* In: Proceedings International Conference on Theoretical Aspects of Computing (ICTAC 2008), LNCS 5160, pp.380-394, Springer (2008).

[WW07] Wachter, B., Westphal, B.: *The Spotlight Principle: On Process-Summarizing State Abstractions.* In: In Andreas Podelski and Byron Cook, editors, Verification, Model Checking and Abstract Interpretation, volume 4349 of Lecture Notes in Computer Science. Springer Verlag (2007).