Diagnosis, Synthesis and Analysis of Probabilistic Models

Tingting Han

Lehrstuhl für Informatik 2, RWTH Aachen
tingting.han@cs.rwth-aachen.de

1 Einleitung

Der zunehmende Einsatz von komplexen Rechnersystemen in vielen kritischen Bereichen wie der Betriebswirtschaft, der Logistik oder der Medizin hat einen hohen Bedarf an formalen Nachweisen der Korrektheit dieser Systeme hervorgerufen. *Formale Verifikation* ist eine bekannte Disziplin der Informatik, die sich zum einen damit beschäftigt, die Fehlerfreiheit von Systemen nachzuweisen, zum anderen aber, vorhandene Fehler zu diagnostizieren. Dabei wird mathematisch nachgewiesen, dass der, als formales Modell gegebene, Designentwurf eines Systems bezüglich einer formalen Spezifikation (in)korrekt ist. *Model checking* [CE81][QS82] ist ein automatisiertes, Modell-basiertes Verfahren, das durch eine in der Regel vollständige Überprüfung des Zustandsraumes ermittelt, ob das Modell eine bestimmte wünschenswerte Eigenschaft erfüllt (*system correctness*). Clarke, Emerson and Sifakis sind für ihre richtungsweisenden Arbeiten im Bereich Model Checking 2007 mit dem Turing Award ausgezeichnet worden.

Probabilistisches Model Checking basiert auf konventionellem Model Checking, erlaubt jedoch, Systemmodelle um probabilistische oder stochastische Information zu erweitern. Bekannte Modelle sind hier Markov-Ketten, mit diskreter oder oder kontinuierlicher Zeit (DTMCs bzw. CTMCs), und Varianten, die auch die Beschreibung von Nichtdeterminismus zulassen. Verfahren zum probabilistischem Model Checking verbinden Methoden aus Numerik und Operations Research mit der klassischen Erreichbarkeitsanalyse. Das ermöglicht, Eigenschaften zu überprüfen wie z.B. „die (maximale) Wahrscheinlichkeit, eine Menge von Zielzuständen zu erreichen, ohne dabei eine ander Menge von Zuständen zu durchlaufen, ist höchstens 0.6“. In dieser Dissertation betrachten wir drei unterschiedliche Aspekte der probabilistischen Verifikation.

Erzeugung von Gegenbeispielen. Eine der Stärken des klassischen Model Checkings ist es, diagnostische Gegenbeispiele zu generieren, wenn eine Eigenschaft vom Modell nicht erfüllt wird. Clarke bringt die Bedeutung dessen auf den Punkt [Cla08]: *„It is impossible to overestimate the importance of the counterexample feature. The counterexamples are invaluable in debugging complex systems. Some people use model checking just for this feature.“* Gegenbeispiele liefern diagnostisches Feedback; sie sind auch der Schlüssel zu effektiven Verfahren zur Modell-Abstraktion und -Verfeinerung, außerdem sind sie essentiell, um Ablaufpläne mit Hilfe von Echtzeit-Model Checking zu generieren.

Gegenbeispiele haben im *probabilistischen* Model Checking allerdings bisher nur geringe Aufmerksamkeit gefunden. Durch die stochastische Natur probabilistischer Modelle können Gegenbeispiele in der Regel nicht durch einzelne Pfade ausgedrückt werden. Stattdessen muß man Mengen von Pfaden betrachten, deren kumulierte Wahrscheinlichkeit die Verletzung einer Eigenschaft aufzeigt. Für verschiedene Kombinationen von probabilistischen Modellen und Logiken untersuchen wir Definitionen von Gegenbeispielen, deren kompakte Repräsentation, sowie Algorithmen zu deren Erzeugung. Die Ergebnisse erweitern bestehende Resultate aus [HK07a][HK07b][DHK08], und dem neueren Artikel [HKD09], und werden in Abschnitt 2 detailiert behandelt.

Parameter-Synthese. Beim traditionellen Model Checking können Eigenschaften nur dann verifiziert werden, wenn sämtliche Parameter des Modells bekannt sind. In frühen Designphasen sind detailierte Informationen bezüglich der Modellparameter, wie beispielsweise zeitliche Aspekte, Kosten, Wahrscheinlichkeiten, etc., oft nicht genau bestimmbar und können nur auf einen Bereich eingegrenzt werden; möglicherweise kann noch deren Beziehung zueinander angegeben werden. Gerade wenn zufällige Einflüsse von außen eine Rolle spielen ist es meist schwierig, präzise Parameterwerte zu erhalten, ohne umfassende Experimente und Messungen durchzuführen. Daher ist es für die Praxis wichtig, bestimmen zu können, für welche Wertebereiche von Parametern des Modells die betrachtete Spezifikation erfüllt wird. Mit anderen Worten, man möchte die Menge der Raten synthetisieren, für welche eine Eigenschaft gültig ist.

Dazu untersuchen wir CTMCs mit parametrisierten Raten. Für eine gegebene zeitbeschränkte Erreichbarkeitseigenschaft beantworten wir die Frage: „Für welche Parameterwerte kann garantiert werden, dass die Spezifikation von der CTMC erfüllt wird?“ Wir stellen ein neues Verfahren vor, das, im Vergleich zur Verwendung von Model Checking, in der Modellierungsphase einen „konstruktiveren“ Weg ermöglicht. Die Resultate basieren auf [HKM08] und werden in Abschnitt 3 eingehend erläutert.

Model Checking von Markovketten. Für DTMCs und CTMCs existieren vielfältige Arbeiten zur Verifikation von *branching-time*-(Echtzeit-)Eigenschaften und *linear-time*-Eigenschaften. Ein bisher ungelöstes, aber praktisch relevantes Problem ist die Verifikation von CTMCs bezüglich linear-time-Echtzeit-Eigenschaften. Für dieses Problem können wir, zumindest teilweise, eine Lösung präsentieren: In Abschnitt 4 untersuchen wir dazu die Verifikation von CTMC s bezüglich deterministischen Zeitautomaten (DTAs). Die vorgestellten Resultate basieren auf [CHKM09] und der kürzlich eingereichten Journal-Version [CHKM10].

2 Erzeugung von Gegenbeispielen

In diesem Abschnitt betrachten wir die Erzeugung von Gegenbeispielen für probabilistisches Model Checking. Eine DTMC $\mathcal{D} = (S, \mathbf{P}, L)$ ist ein *labelled transition system* (hier mit einer *endlichen* Menge von Zuständen S und einer Funktion L zur Zustandsbeschrif-

tung) wobei jede Transition von s nach s' mit Wahrscheinlichkeit $\mathbf{P}(s, s')$ beschriftet ist, mit der Bedingung, dass $\sum_{s' \in S} \mathbf{P}(s, s') = 1$ für alle $s \in S$. Eine der wichtigsten Arten von Eigenschaften, die im probabilistischen Model Checking überprüft werden, ist die Beschränktheit von *Erreichbarkeitswahrscheinlichkeiten*, d.h. der Wahrscheinlichkeit, einen bestimmten Zielzustand t von einem Startzustand s_0 zu erreichen. Für eine gegebene Schranke p ist diese Erreichbarkeitseigenschaft verletzt, wenn die Wahrscheinlichkeit, t zu erreichen, größer als p ist.

2.1 Indizien und Gegenbeispiele

Ist die Erreichbarkeitseigenschaft verletzt, bedeutet das, dass die kumulierte Wahrscheinlichkeit der Menge von Pfaden von s_0 nach t größer als p ist. Da jeder dieser Pfade zu dieser Eigenschaftsverletzung beiträgt, nennen wir sie *Indizien* (*evidence*), die mit der höchsten Wahrscheinlickeit *stärkste Indizien* (*strongest evidence*). Ein *Gegenbeispiel* ist dann eine Menge von Indizien, deren kumulierte Wahrscheinlichkeit p überschreitet. Ein *kleinstes* Gegenbeispiel ist ein Gegenbeispiel mit minimaler Anzahl von Indizien bei maximaler kumulierter Wahrscheinlichkeit (unter all den Gegenbeispielen mit derselben Anzahl von Indizien).

Beispiel 1 *Wir betrachten die* DTMC *in Abb. 1, wobei* s *und* t *Start- bzw. Zielzustand sind. Sei* 0.34 *die Wahrscheinlickeitsschranke. Die Erreichbarkeitseigenschaft ist verletzt, und die Indizien sind, unter anderen,* $\sigma_1 = s_0{\cdot}s_1{\cdot}t_1$, $\sigma_2 = s_0{\cdot}s_1{\cdot}s_2{\cdot}t_1$, *sowie* $\sigma_3 = s_0{\cdot}s_2{\cdot}t_1$. *Die jeweiligen Pfadwahrscheinlichkeiten sind* 0.2, 0.2 *und* 0.15. *Stärkste (und unterschiedliche) Indizien sind* σ_1 *und* σ_2. *Die Menge* $C_1 = \{\sigma_1, \sigma_2, \sigma_3\}$ *mit* $|C_1| = 3$ *und* $\mathbb{P}(C_1) = 0.55 > 0.34$ *ist ein Gegenbeispiel; ebenso* $C_2 = \{\sigma_1, \sigma_3\}$ *mit* $|C_2| = 2$. $C_3 = \{\sigma_1, \sigma_2\}$ *ist ein kleinstes Gegenbeispiel, da* $\mathbb{P}(C_3) = 0.4 > 0.35 = \mathbb{P}(C_2)$.

Um die stärksten Indizien und kleinsten Gegenbeispiele zu berechnen, transformieren wir die DTMC $\mathcal{D}$ in einen strukturell identischen gerichteten Graphen $\mathcal{G}_\mathcal{D}$, wobei wir für alle Transitionen s, s' in $\mathcal{D}$ mit Wahrscheinlichkeit $\mathbf{P}(s, s')$ Gewicht $\log \frac{1}{\mathbf{P}(s,s')}$ an die korrespondierende Kante in $\mathcal{G}_\mathcal{D}$ schreiben. Die Multiplikation von Wahrscheinlichkeiten in der DTMC reduziert sich damit zur Addition der Kantengewichte im Graphen. Die Korrektheit dieser Transformation ist garantiert, da **der kürzeste Pfad in $\mathcal{G}_\mathcal{D}$ mit dem wahrscheinlichsten Pfad in $\mathcal{D}$ korrespondiert**. Dies kann zum **k-kürzesten (-wahrscheinlichsten) Pfad** verallgemeinert werden.

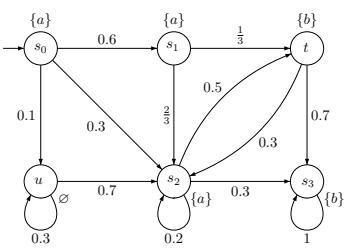


Abbildung 1: DTMC Beispiel 1

Dieses Resultat erlaubt uns, die Berechnung von stärksten Indizien ($k = 1$) bzw. kleinsten Gegenbeispielen (wenn k minimal ist, so dass k Pfade die Wahrscheinlichkeitsschranke überschreiten) als k-kürzestes Pfad-Problem in $\mathcal{G}_\mathcal{D}$ aufzufassen. So kann man das stärkste Indiz zum Beispiel mit *Dijkstras Algorithmus* für kürzeste Pfade finden; bei einer Beschränkung der Pfadlänge läßt sich der *Bellman-Ford-Algorithmus* einsetzen. Kleinste Gegenbeispiele lassen sich mit Eppsteins Algorithmus oder dem *recursive enumeration al-*

gorithm (REA) [JM99] finden. **Wir haben an diesen klassischen Algorithmen mehrere nicht-triviale Modifikationen vorgenommen, um auch mit komplizierteren Längenbeschränkungen umgehen zu können**, beispielsweise mit Pfaden mit Länge h aus dem Intervall $[l, b]$, oder mit Pfaden mit einer Mindestlänge h.

Beispiel 2 (Fortsetzung von Beispiel 1) *Die stärksten Indizien σ_1 and σ_2 sind auch die kürzesten Pfade in $\mathcal{G}_\mathcal{D}$ (nicht abgebildet), mit den Pfadgewichten $\log \frac{1}{0.6} + \log \frac{1}{\frac{1}{3}} = \log 5$ für σ_1, bzw. $\log \frac{1}{0.6} + \log \frac{1}{\frac{2}{3}} + \log \frac{1}{0.5} = \log 5$ für σ_2. Beide Pfade haben minimales Gewicht, verglichen mit anderen Pfaden. Das kleinste Gegenbeispiel $C_3 = \{\sigma_1, \sigma_2\}$ enthält somit die beiden kürzesten Pfade von $\mathcal{G}_\mathcal{D}$.*

2.2 Kompakte Repräsentation von Gegenbeispielen

Da die Größe eines Gegenbeispieles (gemessen an der Zahl der Indizien) exponential in der Anzahl der DTMC-Zustände sein kann, werden wir nun Gegenbeispiele mittels *regulärer Ausdrücke* kompakter darstellen. Reguläre Ausdrücke sind wohlbekannt, verständlich und kurz. Der Ansatz besteht darin, eine DTMC als deterministischen endlichen Automaten (DEA) aufzufassen, und reguläre Ausdrücke durch sukzessive Zustandselimination zu erhalten. Die Reihenfolge der Elimination wird zuvor heuristisch ermittelt (siehe, z.B., [HW07]). Die Berechnung der Wahrscheinlichkeit eines regulären Ausdrucks stützt sich dabei auf einen Ansatz von [Daw04] für parametrisches Model Checking von DTMCs. Dies führt letztendlich zu einer rekursiven Auswertung, die exakt ist, solange die Transitionswahrscheinlichkeiten rational bleiben. Für beschränkte *Until*-Formeln entwickeln wir eine neue rekursive Auswertung von beschränkten regulären Ausdrücken. Wir entwerfen einen Algorithmus, der die Minimalität des durch den regulären Ausdrücke repräsentierten Gegenbeispiels garantiert.

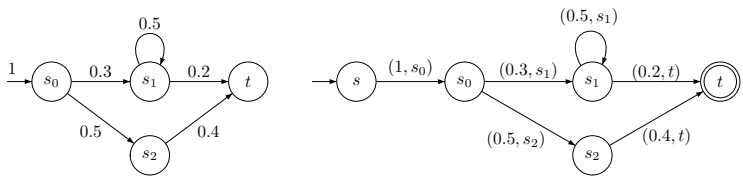


Abbildung 2: Eine DTMC und der zugehörige DEA

Beispiel 3 *In Abb. 2 ist auf der linken Seite eine* DTMC *abgebildet (die Transitionen, die nicht zu t führen sind ausgelassen), auf der rechten Seite der zugehörige DEA. Der reguläre Ausdruck ist $s_0 s_1^* t + s_0 s_2 t$ mit zugehöriger Wahrscheinlichkeit $0.3 \times \frac{1}{1-0.5} \times 0.2 + 0.5 \times 0.4 = 0.23$. Für Wahrscheinlickeitsschranke $p = 0.18$ terminiert der Algorithmus u.U. schon, sobald er den Pfad $s_0 s_2 t$ als regulären Ausdruck generiert, dessen Wahrscheinlichkeit bereits $0.2 > 0.18$ ist.*

Wir haben die Resultate für die Darstellung von Gegenbeispielen als explizite Aufzählung und als reguläre Ausdrücke so verallgemeinert, dass sie auch für andere probabilistische Modelle (MDPs und CTMCs) und andere Logiken gültig sind.

3 Parameter-Synthese

In der Designphase von Systemen ist man weniger an der Verifikation einer konkreten Probleminstanz interessiert, sondern an der Bestimmung von Parametergrenzen innerhalb welcher die gegebenen Spezifikationen erfüllt werden. Beispiele hierfür sind verlässliche Mehrprozessorsysteme und verteilte Datensicherungssysteme, bei denen der Ausfall von Komponenten (wie Speicher- und Prozessoreinheiten) berücksichtigt wird und wobei für die Reparaturzeiten lediglich untere und obere zeitliche Schranken bekannt sind. Solche Systeme möchte man nicht nur bezüglich verschiedener Parameter verifizieren, stattdessen ist man an der Synthetisierung einer Menge von zulässigen Parametern interessiert.

Das Syntheseproblem ist mit dem Model Checking-Verfahren verwandt: Gegeben ist ein (parametrisiertes) Modell und eine logische Beschreibung der Spezifikationen, wir haben eine formale Erfüllbarkeitsbeziehung, und so weiter. Die algorithmische Berechnung der Menge aller zulässigen Parameter ist jedoch deutlich schwieriger und macht die Entwicklung neuer Herangehensweisen erforderlich. In diesem Abschnitt stellen wir eine solche Herangehensweise vor.

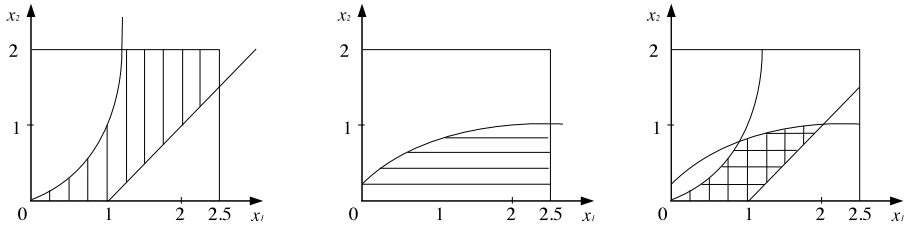
3.1 Das Syntheseproblem

In einer CTMC $\mathcal{C} = (S, \mathbf{R}, L)$ haben wir, statt einer Wahrscheinlichkeitsmatrix $\mathbf{P}$ wie für DTMCs, eine Ratenmatrix $\mathbf{R}$, wobei $\mathbf{R}(s, s')$ die Rate der Transition von s nach s' ist. Vereinfacht ausgedrückt geben Raten die durchschnittliche Geschwindigkeit für Zustandsübergänge an. Wir betrachten eine parametrisierte Version von CTMCs (pCTMC) $\mathcal{C}^{(\mathcal{X})} = (S, \mathbf{R}^{(\mathcal{X})}, L)$, wobei $\mathcal{X}$ die Menge der möglichen Parameter ist und jeder Parameter in einem geschlossenen Interval liegt (z.B. die rechteckige Fläche in Abb. 3(a) mit $\mathcal{X} = \{x_1, x_2\}$). Wir bezeichnen eine solche Fläche als *range region* ζ_{range}. Die Raten werden als Polynome über die Parameter angegeben. Da die Raten einer CTMC nichtnegativ sein müssen, fordern wir dies auch von den parametrisierten Raten (in Abb. 3(a) erhalten wir somit die Fläche zwischen den Kurven). Aus dieser Fläche und ζ_{range} ergibt sich die *initial region* ζ_{init} (die schraffierte Fläche in Abb. 3(a)).

Für eine gegebene zeitbeschränkte Erreichbarkeitseigenschaft mit einem Grenzbereich für die Wahrscheinlichkeit ist die Hauptaufgabe bei der Synthese das Bestimmen der *synthesis region* ζ^*_{syn} (die schraffierte Fläche in Abb. 3(c)) für die jeder Punkt in ζ^*_{syn} eine CTMC induziert, welche die gegebene Eigenschaft erfüllt.

Um dieses Ziel zu erreichen, führen wir zwei Schritte durch: (1) die Bestimmung der *synthesis region* innerhalb des rechteckigen Wertebereichs (ζ_{syn}, siehe Abb. 3(b)) und (2) die Berechnung des Schnitts $\zeta^*_{syn} = \zeta_{range} \cap \zeta_{syn}$ (siehe Abb. 3(c)). Wir werden uns im Folgenden hauptsächlich auf den ersten Schritt konzentrieren.

In diesem Zusammenhang haben wir einen *symbolischen* und eine *nicht-symbolischen* Ansatz zur Approximation der *synthesis region* untersucht. Aufgrund der besseren praktischen Resultate werden wir hier nur auf den nicht-symbolischen Ansatz eingehen.



(a) initial region ζ_{init} (b) ζ_{syn} (c) $\zeta'_{syn} = \zeta_{init} \cap \zeta_{syn}$

Abbildung 3: pCTMCs und verwandte Konzepte

3.2 Nicht-symbolische Approximation der *synthesis region*

Der nicht-symbolische Ansatz besteht darin, ζ_{syn} im Bereich von ζ_{range} zu finden. Im Prinzip muss jeder Punkt in ζ_{range} auf Erfüllbarkeit der Erreichbarkeitseigenschaft hin überprüft werden. Dies ist allerdings nicht praktikabel, da ζ_{range} eine überabzählbare Menge ist. Ein verbreiteter Ansatz ist es, den "kontinuierlichen" Bereich ζ_{range} zu diskretisieren. Dadurch erhält man eine endliche Menge von "Proben" anhand welcher die tatsächliche *synthesis region* approximiert werden kann.

Für eine gegebene Gitterbreite kann ζ_{range} wie in Abb. 4 diskretisiert werden. Für jeden Gitterpunkt (eine Probe) kann die zugehörige CTMC durch Instanziierung der Parameter erzeugt werden. Die Berechnung der transienten Wahrscheinlichkeiten wird dann bezüglich dieser CTMC durchgeführt, woraus sich direkt ergibt ob der Gitterpunkt im Bereich ζ_{syn} liegt oder nicht. Die Hauptschwierigkeit dieses Ansatzes besteht darin, die Kurven von ζ_{syn} möglichst gut zu approximieren, ohne zu viele Gitterpunkte zu erzeugen, die analysiert werden müssen. Hierfür wählen wir eine Verfeinerungstechnik, die die Zellen des Gitters nur dann feiner auflöst, wenn sie auf einer Kurve liegen.

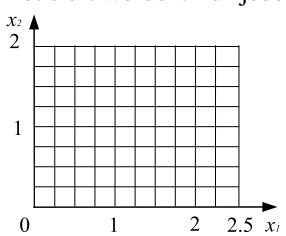
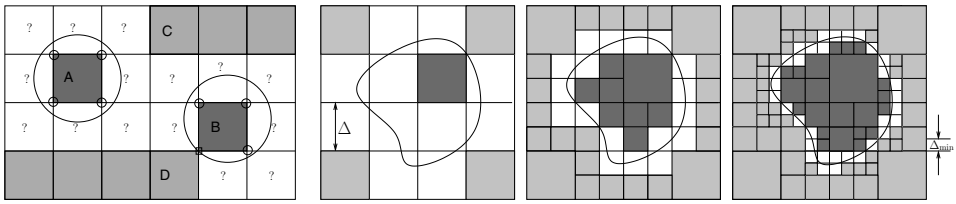


Abbildung 4: Discretization

3.3 Markierung und Verfeinerung

Markierung. Für den Markierungsschritt geben wir ein Kriterium an, mit dem grob entschieden werden kann, ob eine Gitterzellen innerhalb oder außerhalb einer *synthesis region* liegt. Beispielsweise sind die dunklen Bereiche in Abb. 5(a) *vollständig innerhalb* der *synthesis region* ζ_{range} (den beiden Kreisflächen) wohingegen die hellgrauen Zellen *vollständig außerhalb* liegen. Die mit "?" markierten weißen Zellen sind teilweise innerhalb und teilweise außerhalb der *synthesis region*, was bedeutet, dass sie auf einer der Kurven liegen.

Verfeinerung. Die ?-markierten Gitterzellen sind diejenigen, für die ohne weitere Verfeinerung keine Aussage bezüglich der Zugehörigkeit zur *synthesis region* getroffen wer-



(a) Marking (b) Refinement

Abbildung 5: Markierung und Verfeinerung

den kann. Daher halbieren wir für eine solche Zelle gc die Gitterbreite in jeder Dimension und erhalten dadurch (für zwei Parameter) vier kleinere Gitterzellen. Für jede dieser Gitterzellen werden nun wiederum die Markierung und die Verfeinerung durchgeführt, bis eine vordefinierte minimale Gitterbreite erreicht wurde (siehe Abb. 5(b)). Wir konnten zeigen, dass die Differenz zwischen tatsächlicher und approximierter Lösung durch die minimale Gitterbreite beschränkt ist.

Die Komplexität der Markierungs- und Verfeinerungsverfahrens ist abhängig von der Anzahl der Gitterzellen, der Anzahl der Verfeinerungsschritte, dem prozentualen Anteil der zu verfeinernden Zellen, etc.

4 Model checking CTMCs für DTA Spezifikationen

CTMC Model Checking-Algorithmen wurden bisher nur für branching-time-Spezifikationen entwickelt. Dadurch war es beispielsweise nicht möglich, die folgende praktisch relevante Eigenschaft zu verifizieren, die mittels des Zeitautomaten (TA) in Abb. 6 formalisiert werden kann: *Prozess A versendet eine Nachricht mit einem maximalen Abstand von* 2 *Zeiteinheiten zwischen aufeinanderfolgenden Versuchen; sollte die Übermittlung der Nachricht nach maximal* 10 *Zeiteinheiten immer noch erfolglos sein, wird die Prozedur mit einem Timeout abgebrochen.*

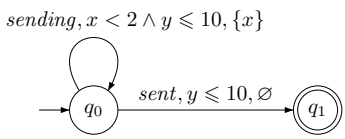


Abbildung 6: DTA Spezifikation

Wir untersuchen das Verifikationsproblem für CTMCs bezüglich *linearen* Echtzeit-Spezifikationen, die in Form von *deterministischen* TA (DTA) [AD94] gegeben sind. Formal betrachten wir das folgende Problem: *Gegeben eine* CTMC $\mathcal{C}$ *und einen* DTA $\mathcal{A}$*, wie groß ist die Wahrscheinlichkeit, dass* $\mathcal{C}$ *die Spezifikation* $\mathcal{A}$ *erfüllt?* Dieses Problem ist äquivalent zur Frage: *Wie groß ist die Wahrscheinlichkeit der Menge aller Pfade in* $\mathcal{C}$*, welche von* $\mathcal{A}$ *akzeptiert werden?*

4.1 Verifikation von DTA Spezifikationen

Ein DTA $\mathcal{A}$ akzeptiert zeitbehaftete Pfade. Zusätzlich zu klassischen Zeitautomaten mit Locations, Guards, Clocks, Resets und Startlocations sowie akzeptierenden Locations, for-

dern wir die *Determiniertheit* des Zeitautomaten, d.h., die Startlocation muss eindeutig sein und die Guards der Transitionen, die mit derselben Aktion aus einem Zustand herausführen, müssen disjunkt sein. Das Model Checking-Problem reduziert sich auf die Berechnung der Wahrscheinlichkeiten der Menge von Pfaden in einer gegebenen CTMC $\mathcal{C}$, die von einem DTA akzeptiert werden. Wir zeigen, dass dieses Problem wohldefiniert ist, d.h., **die Menge der Pfade ist *messbar***.

Dazu nehmen wir das Produkt $\mathcal{C} \otimes \mathcal{A}$ von CTMC $\mathcal{C}$ und DTA $\mathcal{A}$. Daraus ergibt sich ein neues Modell, die *deterministischen Markov Zeitautomaten* (DMTA), eine Variante der DTA, in der neben den üblichen Komponenten der Zeitautomaten wie Guards, Clocks und Resets zusätzlich die Verweildauer in einer Location als exponentiell verteilt angenommen wird. **Die Wahrscheinlichkeit, mit der $\mathcal{C}$ von DTA $\mathcal{A}$ akzeptiert wird, ist gleich der Erreichbarkeitswahrscheinlichkeit für akzeptierte Pfade in $\mathcal{C} \otimes \mathcal{A}$.**

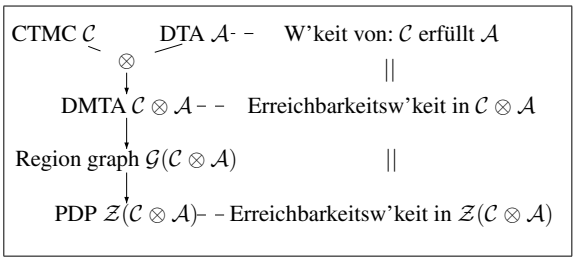


Abbildung 7: Berechnung der W'keit von: $\mathcal{C}$ erfüllt $\mathcal{A}$

Für einen DMTA $\mathcal{C} \otimes \mathcal{A}$ erhalten wir den *region graph* mit einer einfachen Adaption der Standard-Konstruktion. Wir zeigen, dass der ***region graph* von $\mathcal{C} \otimes \mathcal{A}$ ein *Piecewise Deterministic Markov Process* (PDP) [Dav93]** ist. Dieses Modell ist beispielsweise aus der stochastischen Kontrolltheorie und der Finanzmathematik bekannt.

Wir zeigen weiterhin: **Die Berechnung der Erreichbarkeitswahrscheinlichkeiten in $\mathcal{C} \otimes \mathcal{A}$ kann auf die Berechnung der Erreichbarkeitswahrscheinlichkeiten im PDP $\mathcal{Z}(\mathcal{C} \otimes \mathcal{A})$ reduziert werden.**

Die entsprechende Erreichbarkeitswahrscheinlichkeit ist charakterisiert durch **die kleinste Lösung eines Systems von Integralgleichungen das durch den PDP gegeben ist.** Schlussendlich zeigen wir, dass diese Wahrscheinlichkeit durch **die Lösung eines Systems von *partiellen Differentialgleichungen* (PDEs)** approximiert werden kann.

4.2 Spezialfälle

Für einige Spezialfälle sind effizientere Lösungsverfahren bekannt, über die im Folgenden ein kurzer Überblick gegeben wird.

$\star$ **Single Clock DTA Spezifikationen.** Für DTAs mit nur einer Uhr zeigen wir, dass das System von Integralgleichungen in ein **System von *linearen Gleichungen*** umgewandelt werden kann, wobei die Koeffizienten die Lösungen von *einfachen Differentialgleichungen* (ODEs) sind, die entweder analytisch berechnet (für kleine Zustandsräume) oder beliebig genau und effizient approximiert werden können.

⋆ **Qualitative Spezifikationen.** Bei qualitativen Spezifikationen wird überprüft, ob die Wahrscheinlichkeit größer als 0 oder gleich 1 ist, dass $\mathcal{A}$ von $\mathcal{C}$ erfüllt wird. Wir zeigen, dass solche Eigenschaften unter Verwendung von Techniken aus der Graphanalyse verifiziert werden können, indem lediglich der endliche, gerichtete Graph betrachtet wird, der dem PDP $\mathcal{Z}(\mathcal{C} \otimes \mathcal{A})$ zugrunde liegt und die Transitionswahrscheinlichkeiten ignoriert werden. Um zu bestimmen, ob die Wahrscheinlichkeit größer als 0 ist, muss überprüft werden ob ein Endzustand im Graph erreichbar ist. Um zu bestimmen, ob die Wahrscheinlichkeit gleich 1 ist, muss überprüft werden, ob ein Pfad existiert, mit dem niemals ein Endzustand erreicht werden kann. In diesem Fall muss die Wahrscheinlichkeit kleiner als 1 sein.

⋆ **DTA$^\omega$ Specificationen.** Für DTAs, die unendliche Pfade akzeptieren (Pfade, die bestimmte Locations unendlich oft besuchen) zeigen wir, dass diese Spezifikationen auf den Fall reduziert werden können, in dem endliche Pfade akzeptiert werden. Dazu müssen die Mengen der akzeptierenden *bottom strongly connected components* (BSCCs) im *region graph* bestimmt werden.

5 Zusammenfassung

In dieser Dissertation wurden drei Bereiche im Forschungsgebiet des probabilistischen Model Checkings behandelt: *Diagnose* — das Erzeugen von Gegenbeispielen, *Synthese* — das Bestimmen von gültigen Parameterwerten und *Analyse* — die Verifikation von linearen Echtzeiteigenschaften. Während diese drei Bereiche vergleichsweise unabhängig voneinander zu betrachten sind, werden doch in allen von ihnen signifikante Beiträge zur Entwicklung neuer Theorien und Algorithmen geleistet, die bisher nur wenig Beachtung fanden. Im Folgenden werden die wichtigsten Neuerungen kurz zusammengefasst.

⋆ Wir haben eine formale Definition und ein algorithmisches Rahmenwerk für die Erzeugung von Gegenbeispielen im probabilitischen Bereich eingeführt. Diese Arbeit hat weitere Beiträge zu diesem Thema inspiriert.

⋆ Im Bereich der Parametersynthese für probabilistische Modelle haben wir einen ersten, wichtigen Schritt getan.

⋆ Wir haben approximative Algorithmen für die Verifikation von linearen Echtzeiteigenschaften für CTMCs vorgestellt, die im Gebiet der Systembiologie Anwendung finden dürften.

In jedem dieser drei Bereiche erwarten wir noch viele weitere Herausforderungen, zu deren Lösung zukünftige Forschungsarbeiten auf die Resultate dieser Dissertation zurückgreifen können.

Danksagung. I sincerely thank my supervisor Joost-Pieter Katoen, thank my colleagues Henrik Bohnenkamp and Daniel Klink for the German translation, and thank everyone who made this happen!

Literatur

[AD94] Rajeev Alur und David L. Dill. A Theory of Timed Automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[CE81] Edmund M. Clarke und E. Allen Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *Logic of Programs*, LNCS 131, Seiten 52–71, 1981.

[CHKM09] Taolue Chen, Tingting Han, Joost-Pieter Katoen und Alexandru Mereacre. Quantitative model checking of continuous-time Markov chains against timed automata specification. In *LICS*, Seiten 309–318, 2009.

[CHKM10] Taolue Chen, Tingting Han, Joost-Pieter Katoen und Alexandru Mereacre. Model Checking of Continuous-Time Markov Chains
Against Timed Automata Specifications. *Logic Methods in Computer Science*, submitted, 2010.

[Cla08] Edmund M. Clarke. The Birth of Model Checking. In *25 Years of Model Checking*, Seiten 1–26, 2008.

[Dav93] Mark H. A. Davis. *Markov Models and Optimization*. Chapman and Hall, 1993.

[Daw04] Conrado Daws. Symbolic and Parametric Model Checking of Discrete-Time Markov Chains. In *ICTAC*, LNCS 3407, Seiten 280–294, 2004.

[DHK08] Berteun Damman, Tingting Han und Joost-Pieter Katoen. Regular expressions for PCTL counterexamples. In *QEST*, Seiten 179–188, 2008.

[HK07a] Tingting Han und Joost-Pieter Katoen. Counterexamples in Probabilistic Model Checking. In *TACAS*, LNCS 4424, Seiten 72–86, 2007.

[HK07b] Tingting Han und Joost-Pieter Katoen. Providing Evidence of Likely Being on Time: Counterexample Generation for CTMC Model Checking. In *ATVA*, LNCS 4762, Seiten 331–346, 2007.

[HKD09] Tingting Han, Joost-Pieter Katoen und Berteun Damman. Counterexample Generation in Probabilistic Model Checking. *IEEE Trans. Software Eng.*, 35(2):241–257, 2009.

[HKM08] Tingting Han, Joost-Pieter Katoen und Alexandru Mereacre. Approximate parameter synthesis for probabilistic time-bounded reachability. In *RTSS*, Seiten 173–182, 2008.

[HW07] Yo-Sub Han und Derick Wood. Obtaining shorter regular expressions from finite-state automata. *Theor. Comput. Sci.*, 370(1-3):110–120, 2007.

[JM99] Víctor M. Jiménez und Andrés Marzal. Computing the K shortest paths: A new algorithm and an experimental comparison. In *WAE*, LNCS 1668, Seiten 15–29, 1999.

[QS82] Jean-Pierre Queille und Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In *Symposium on Programming*, LNCS 137, Seiten 337–351, 1982.