Seminar

Eine Logik für das Schlussfolgern über Zeit und Zuverlässigkeit

Theodor Teslia

Informatik 11 – Embedded Software
RWTH Aachen University
Aachen, Germany
teslia@embedded.rwth-aachen.de

Betreuer
Robin Mross

Abstract: Eine prägnante Zusammenfassung des Kerninhaltes ohne thematische Einleitung und Fazit. Was ist PCTL und welche Probleme kann es lösen, die andere Logiken bisher nicht erfüllen konnten.

1 Einführung

Hier führe ich in die Logik PCTL ein, die Probleme die diese löst und wie dies circa gemacht wird. Falls es funktioniert, führe ich außerdem in Halbringsemantik ein und erkläre kurz, in welchem Punkt sich die Ansätze ähneln und unterscheiden.

2 Grundlagen

Um Erweiterungen einer Logik zu verstehen ist es ratsam die Grundlegende auch zu kennen. Aus diesem Grund soll in diesem Kapitel in die Logik *Computation Tree Logic* (CTL) und naheliegende, wichtige Themen eingeleitet werden.

CTL ist eine temporale Logik um Aussagen in nicht-deterministischen Systemen zu treffen. Dafür betrachtet man Kripkestrukturen, welche eine Art Graph darstellen, da diese viele Eigenschaften liefern, die für CTL-Anwendungsfälle interessant sind.

Definition 2.1 (Syntax von CTL)
Die Menge der CTL-Formeln lässt sich induktiv mithilfe folgender Regeln definieren [CE82, BK08]:

1. Wenn AP eine Menge an atomaren Aussagen ist, dann ist jedes $a \in \mathsf{AP}$ eine CTL-Formel. Außerdem sind $\top$ und $\bot$ CTL-Formeln.
2. Wenn φ_1 und φ_2 CTL-Formeln sind, dann sind $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$ und $\varphi_1 \vee \varphi_2$ ebenfalls CTL-Formeln.
3. Wenn φ_1 eine CTL-Formel ist, dann sind auch $\mathrm{EX}\,\varphi_1$ und $\mathrm{AX}\,\varphi_1$ CTL-Formeln.
4. Wenn φ_1 und φ_2 CTL-Formeln sind, dass sind $\mathrm{E}[\varphi_1\,\mathrm{U}\,\varphi_2]$ und $\mathrm{A}[\varphi_1\,\mathrm{U}\,\varphi_2]$ auch CTL-Formeln.

Bevor die Semantik erläutert wird, sollen zuerst die Strukturen gezeigt werden, die man typischerweise mit CTL zusammen verwendet.

Definition 2.2 (Kripkestrukturen)
Eine *Kripkestruktur* oder *Transitionssystem* ist ein Graph von der Form $\mathcal{K} = (V, E, \mathcal{L})$, wobei E eine zweistellige Kantenrelation über V ist und $\mathcal{L}$ eine Funktion $\mathcal{L} : V \to 2^{\mathsf{AP}}$ ist, die jedem Zustand eine Menge an atomaren Aussagen zuweist [CE82, CES86].

Weiter bezeichnen wir einen Pfad als $\sigma = s_0 \ldots s_n$ mit $s_i \in V$ für alle $i \in \{0, \ldots, n\}$ und $\sigma[i] = s_i$ für $0 \leq i \leq n$ [BK08].

Definition 2.3 (Semantik von CTL)
Für ein Transitionssystem $\mathcal{K} = (V, E, \mathcal{L})$ können wir damit induktiv die Modellbeziehung $\models$ zwischen einem Knoten $v \in V$ und CTL-Formeln definieren [BK08]:

1. $v \models \top \Leftrightarrow \forall x(x = x)$ und $v \models \bot \Leftrightarrow \exists x(x \neq x)$.
2. Für $a \in \mathrm{AP}$ gilt $v \models a \Leftrightarrow a \in \mathcal{L}(v)$.
3. Es gilt $v \models \neg\varphi \Leftrightarrow v \not\models \varphi$,
   $v \models \varphi_1 \wedge \varphi_2 \Leftrightarrow v \models \varphi_1$ und $v \models \varphi_2$,
   $v \models \varphi_1 \vee \varphi_2 \Leftrightarrow v \models \varphi_1$ oder $v \models \varphi_2$,
4. Es gilt $v \models \mathrm{EX}\,\varphi \Leftrightarrow$ es ex. ein $w \in V$ mit $(v, w) \in E$ und $w \models \varphi$ und analog:
   $v \models \mathrm{AX}\,\varphi \Leftrightarrow$ für alle $w \in V$ mit $(v, w) \in E$ gilt $w \models \varphi$.
5. Es gilt $v \models \mathrm{E}[\varphi_1\,\mathrm{U}\,\varphi_2] \Leftrightarrow$ es existiert ein Pfad σ in $\mathcal{K}$, der in v beginnt und ein $i \in \mathbb{N}$ so, dass $\sigma[i] \models \varphi_2$ und für alle $0 \leq j < i$ gilt $\sigma[i] \models \varphi_1$. Analog ist die Definition für $\mathrm{A}[\varphi_1\,\mathrm{U}\,\varphi_2]$, es muss dann aber für jeden in v beginnenden Pfad ein $i \in \mathbb{N}$ geben so, dass der i-te Zustand φ_2 und alle Zustände davor φ_1 erfüllen.

Intuitiv bedeutet $\mathrm{EX}\,\varphi$ also, dass φ in einem beliebigen Nachfolgezustand gelten muss und $\mathrm{AX}\,\varphi$, dass φ von allen Nachfolgezuständen erfüllt wird.

$\mathrm{E}[\varphi_1\,\mathrm{U}\,\varphi_2]$ bzw. $\mathrm{A}[\varphi_1\,\mathrm{U}\,\varphi_2]$ sagt aus, dass es einen Pfad σ gibt bzw. auf allen Pfaden σ gilt, dass zuerst φ_1 gilt, bis von einem Zustand φ_2 erfüllt wird. Die anderen Operatoren haben die bekannte Bedeutung.

Um einige Eigenschaften einfacher auszudrücken werden zusätzlich noch weitere Operatoren definiert [CE82]:

- $\mathrm{EF}\,\varphi \equiv \mathrm{E}[\top\,\mathrm{U}\,\varphi]$ und $\mathrm{AF}\,\varphi \equiv \mathrm{A}[\top\,\mathrm{U}\,\varphi]$ bedeuten intuitiv, dass es einen Pfad gibt bzw. für alle Pfade irgendwann φ gilt.
- $\mathrm{EG}\,\varphi \equiv \neg\,\mathrm{EF}\,\neg\varphi$ und $\mathrm{AG}\,\varphi \equiv \neg\,\mathrm{AF}\,\neg\varphi$ bedeuten, dass es einen Pfad gibt bzw. für alle Pfade gilt, dass in jedem Zustand φ gilt.

Mithilfe dieser Syntax und zugehöriger Semantik auf Transitionssystemen lassen sich nun einige interessante Aussagen formulieren:

Beispiel 1 (Beispiele für CTL-Formeln und deren Bedeutung)
Sei die Menge der atomaren Aussagen $\mathsf{AP} = \{\mathsf{idle}, \mathsf{error}\}$

- $\mathrm{AG}\,\mathrm{AF}\,\mathsf{idle}$
  Die Formel gilt für einen Zustand v, wenn auf jedem in v beginnenden Pfad, für jeden Zustand auf diesem, irgendwann idle erfüllt. Das heißt, idle gilt unendlich oft.
- $\mathrm{EF}\,\mathrm{AG}\,\mathsf{error}$
  Diese CTL-Formel besagt, dass es einen Pfad gibt, auf dem ab irgendeinem Zustand alle Zustände die atomare Aussage error erfüllen. Das heißt, es gibt einen Pfad mit einem irreversiblen Fehler.

Zusätzlich soll noch angemerkt werden, dass $0 \in \mathbb{N}$ gilt.

3 Eine Logik für Zeit und Zuverlässigkeit

Wie in Kapitel 2 gezeigt, lassen sich mithilfe von CTL viele interessante Eigenschaften von nicht-deterministischen System beschreiben. Jedoch gibt es auch Anwendungsfälle, in denen mehr Ausdruckskraft benötigt wird, als ein All- und Existenzquantor liefern können. Ein Gebiet in dem dies stark auffällt sind Soft-Realtime Systeme. In diesen existieren für Prozesse bestimmte Zeitschranken (*Deadlines*), im Gegensatz zu Hard-Realtime Systemen führt das einhalten einer Zeitschranke aber nicht zu einem Systemabbruch oder katastrophalem Ereignis, sondern stellt beispielsweise nur eine Verschlechterung der Effizienz dar. [HJ94]. Um eben solche Systeme gut beschreiben zu können benötigt man zwei weitere Aspekte:

1. Um die Zeitschranken zu formulieren wird ein Konzept von Zeit benötigt. Dieses soll aussagen können, dass zwei Ereignisse eine bestimmte Zeitspanne t voneinander entfernt sind.
2. Da aber das Verfehlen einer Zeitschranke nicht unbedingt zum Verwerfen einer Formel führen soll, werden zusätzlich Wahrscheinlichkeiten benötigt. Da in Soft-Realtime Systemen das Überschreiten einer Deadline zwar nicht direkt verboten ist, aber im Allgemeinen vermieden werden sollte, ist es sinnvoll über die Wahrscheinlichkeit eines Ereignisses Aussagen zu treffen.

Kombiniert man diese Aspekte lassen sich Eigenschaften wie „Nach Ereignis X passiert innerhalb von 15 Zeiteinheiten mit Wahrscheinlichkeit 90% Ereignis Y“ oder „Ereignis A tritt mit einer Wahrscheinlichkeit von 90% in 10 und mit 95% in 20 Zeitschritten auf“. Eine Logik, die eben diese Erweiterungen von CTL sinnvoll implementiert ist die Logik Probabilistic Computation Tree Logic (PCTL). Sinnvoll bedeutet hier, dass es einen Model Checking Algorithmus mit polynomieller Laufzeit gibt. In diesem Kapitel soll zuerst die Syntax von PCTL erläutert werden, danach die dazugehörige Semantik aufgezeigt werden, um dann zwei verschiedene Ansätze für das Model Checking von PCTL mit Transitionssystemen zu zeigen. Im Anschluss sollen die kennengelernten Konzepte der Logik sowie des Model Checkings an einem Beispiel erläutert werden.

3.1 Syntax und Semantik von PCTL

Um *Probabilistic Computation Tree Logic* (PCTL) besser zu verstehen, soll hier die Syntax, die Modelle welche wir zum Auswerten verwenden, sowie die Semantik der Logik erläutert werden.

Wie auch für CTL können wir die Syntax von PCTL mithilfe folgender rekursiver Regeln definieren:

Definition 3.1 (Syntax von PCTL)
Die Menge der PCTL-Formeln lässt sich induktiv wie folgt definieren [HJ94]:

1. Es gilt $\top \in \mathsf{PCTL}$ und $\bot \in \mathsf{PCTL}$.
2. Wenn AP die Menge atomarer Aussagen ist, dann ist jedes $a \in \mathsf{AP}$ eine PCTL Formel.
3. Wenn φ_1 und φ_2 PCTL-Formeln sind, dann sind $\neg\varphi_1$ und $(\varphi_1 \wedge \varphi_2)$ auch PCTL-Formeln.
4. Für zwei PCTL-Formeln φ_1 und φ_2, $t \in \mathbb{N} \cup \{\infty\}$ und $p \in [0,1] \subseteq \mathbb{R}$, sind $\varphi_1 \, \mathrm{U}^{\leq t}_{\geq p} \, \varphi_2$ und $\varphi_1 \, \mathrm{U}^{\leq t}_{> p} \, \varphi_2$ auch PCTL-Formeln.

Mit diesen Regeln können wir einige PCTL-Formeln aufstellen.

Beispiel 2 (Korrekte und inkorrekte PCTL-Formeln)
Sei $\mathsf{AP} = \{A, B, X, Y\}$. Dann wären

$$\neg(X \wedge \neg(\top \, \mathrm{U}^{\leq 15}_{\geq 90\%} \, Y)) \text{ und } (A \, \mathrm{U}^{\leq 10}_{\geq 90\%} \, B) \wedge (A \, \mathrm{U}^{\leq 20}_{> 95\%} \, B)$$

korrekte PCTL-Formeln.

Inkorrekt gebildete Formeln wären zum Beispiel

$$\neg(X \wedge \neg(\top \, \mathrm{U}^{\leq 15}_{\geq 90\%})) \text{ und } (A \, \mathrm{U}^{\leq 10}_{\geq 90\%} \, B)(A \, \mathrm{U}^{\leq 20}_{> 95\%} \, B).$$

Ähnlich, wie wir Transitionssysteme definiert haben, um diese als Modelle von CTL-Formeln zu verwenden, wollen wir nun sogenannte Markov-Ketten definieren, um Eigenschaften von diesen mithilfe von PCTL zu formulieren.

Definition 3.2 (Markov-Ketten)
Sei S eine endliche Menge, $s_i \in S$ und $\mathcal{L} : S \to 2^{\mathsf{AP}}$ sowie $\mathcal{T} : S \times S \to [0, 1]$ Funktionen so, dass für alle $s \in S$ gilt: $\sum_{s' \in S} \mathcal{T}(s, s') = 1$.

Dann nennen wir $\mathfrak{S} = (S, s_i, \mathcal{T}, \mathcal{L})$ eine Markov-Kette, wobei S eine Menge an Zuständen ist, s_i der Anfangszustand, $\mathcal{T}$ die Transitions-Wahrscheinlichkeits-Funktion und $\mathcal{L}$ die Bezeichnungsfunktion, die jedem Zustand eine Menge an atomaren Aussagen zuweist.

Weiter bezeichnen wir mit $\mathrm{paths}_{s_0}^{\mathfrak{S}}$ die Menge der Pfade in $\mathfrak{S}$, die in s_0 beginnen. Ein $\sigma \in \mathrm{paths}_{s_0}^{\mathfrak{S}}$ ist dann von der Form $\sigma = s_0 s_1 s_2 \dots$ und wir definieren $\sigma[n] \coloneqq s_n$ als den n-ten Zustand des Pfads und $\sigma \uparrow n \coloneqq s_0 \dots s_n$ als den $n + 1$ langen Präfix des Pfads. [HJ94]

Zur Einfachheit sagen wir, dass zwischen vom Knoten s zum Knoten s' genau dann eine Kante existiert, wenn $\mathcal{T}(s, s') \neq 0$. Man erkennt, dass im Allgemeinen ein Pfad $\sigma \in \mathrm{paths}_{s_0}^{\mathfrak{S}}$ unendlich lang ist. Dies ist wohldefiniert, da jeder Zustand eine ausgehende Kante haben muss. Andernfalls gibt es ein $\hat{s} \in S$ mit $\{s' \in S : \mathcal{T}(\hat{s}, s') = 0\} = S$. Aber dann ist $\sum_{s' \in S} \mathcal{T}(\hat{s}, s') = 0$. Widerspruch! Es gibt für jeden Zustand also mindestens einen Nachfolgezustand, es gibt also immer unendliche Pfade.

Eine Markov Kette ist also ein gerichteter, gewichteter Graph, wobei die Gewichtung der Kanten angibt, wie wahrscheinlich es ist, eine bestimmte Kante auszuwählen. Zusätzlich soll die Summe aller Gewichte der ausgehenden Kanten eines Knotens immer gleich eins sein. Damit ist auch gewährleistet, dass es keine isolierten Knoten gibt.

Im Kontext von Systemen sollen die Zustände des Graphens Zustände des Systems beschreiben. Die Kanten stellen die Möglichen Folgezustände des Systems dar, wobei der Wechsel in einen Folgezustand mit der Wahrscheinlichkeit durchgeführt wird, mit der die Kante annotiert ist. In jedem Zustand gelten atomare Aussagen, welche die Zustände beschreiben, diese werden von der Funktion $\mathcal{L}$ zugewiesen.

Betrachten wir eine Markov-Kette als Beispiel:

Beispiel 3 (Beispiel einer Markov-Kette)
Sei $S = \{s_1, s_2, s_3, s_4\}$, $\mathcal{L} = \{s_1 \mapsto A, s_2 \mapsto B, s_3 \mapsto C, x_4 \mapsto D\}$ und $\mathcal{T}$ durch Tabelle 1 definiert. Dann ist $\mathfrak{S} = (S, s_1, \mathcal{T}, \mathcal{L})$ die in Abbildung 1 graphisch dargestellte Markov-Kette, wobei Transitionen mit einer Wahrscheinlichkeit von 0 nicht eingezeichnet werden.

Bevor wir die Semantik von PCTL-Formeln für Markov-Ketten formal definieren können benötigen wir noch den Begriff des Wahrscheinlichkeitsmaßes.

Definition 3.3 (Wahrscheinlichkeitsmaß)
Sei $\pi = s_0 \dots s_n$ eine Folge an Zuständen einer Markov-Kette $\mathfrak{S}$ und $X = \{\sigma \in \mathrm{paths}_{s_0}^{\mathfrak{s}} : \sigma \uparrow n = \pi\}$ die Menge aller (unendlichen) Pfade, die mit dieser Folge be-

$\mathcal{T}$	s_1	s_2	s_3	s_4
s_1	0	1	0	0
s_2	0.15	0	0.7	0.15
s_3	0	0.7	0	0.3
s_4	0	0	0	1

Tabelle 1: Tabelle zur Definition der Funktion $\mathcal{T} : S \times S \to [0, 1]$

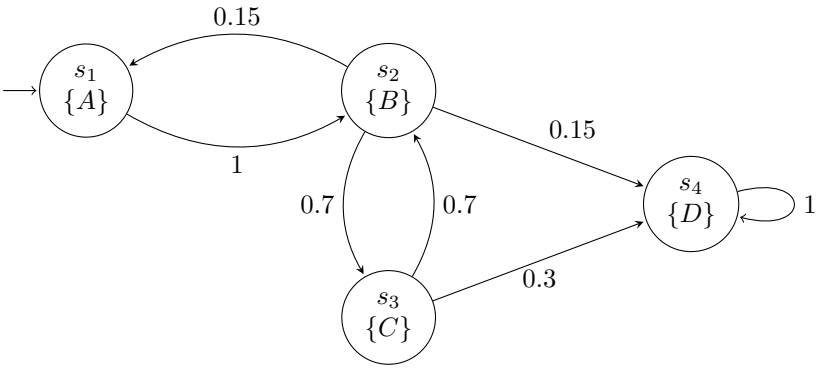

Abbildung 1: Graph für die Markov-Kette $\mathfrak{S}$

ginnen. Wir definieren dann das Wahrscheinlichkeitsmaß

$$\mu_{s_0}^{\mathfrak{S}}(X) \coloneqq \mathcal{T}(s_0, s_1) \cdot \dots \cdot \mathcal{T}(s_{n-1}, s_n)$$

Definition 3.4 (Semantik von PCTL auf Markov-Ketten)
Sei $\mathfrak{S} = (S, s_i, \mathcal{T}, \mathcal{L})$ eine Markov-Kette. Dann lässt sich die Semantik für PCTL-Formeln induktiv definieren:

1.

3.2 Semantik

Hier wird die Semantik erklärt und das verwendete Transitionssystem, die Markov Kette eingeführt. Einige simple, abstrakte Beispiele ebenfalls zum Erklären verwendet werden. Ein Vergleich mit CTL soll passieren, wobei auf die in [HJ94] definierte Einbettung verwiesen, aber auch der Fehler bzgl. $\mathrm{EG}\,\varphi$ gezeigt wird.

3.3 Model-Checking Algorithmen

Die in [HJ94] genannten Model-Checking Algorithmen werden erklärt und anhand von Tabellen und unterschiedlichen Formeln erläutert. Wie im Paper soll eine Aufteilung in die unterschiedlichen Parameter-*Arten* (also $0 < t < \infty$, $t = 0$, $t = \infty$ und analog für p) stattfinden.

3.4 Angewandtes Beispiel

Was genau als Beispiel verwendet wird, muss noch entschieden werden, im Zweifelsfall ein etwas anderes Übertragungsprotokoll als im Paper. Einige unterschiedliche Formeln sollen übersetzt werden so, dass auch die Bedeutung anderer Formeln als nur $\rightsquigarrow$ klarer wird.

4 Vergleich mit Halbringsemantik

Ob dieses Kapitel umgesetzt wird hängt von der Vergleichbarkeit von Viterbi-Halbring + CTL mit PCTL ab. Der Platz im restlichen Teil der Arbeit, der durch die Existenz des Kapitels wegfällt, soll von Kapitel 5 gepuffert werden. Bei Platzproblemen kann über das Zusammenlegen der Kapitel 4.1 und 4.2 nachgedacht werden. Hier soll kurz erklärt werden, was genau Halbringsemantik ist und wofür sie im Allgemeinen verwendet wird. In dieser Einleitung und Kapitel 4.1 wird vor allem auf [GT17] verwiesen.

4.1 Halbringsemantik zum Auswerten von Wahrscheinlichkeiten

Verwendung des Viterbi-Halbrings zum Auswerten von FO auf Graphen. Dieses Kapitel wird bei Platzproblemen weggelassen und der Übergang von FO auf CTL wird kurz in Kapitel 4.2 passieren. Falls ich kein veröffentlichtes Paper mit den Informationen finde, benutze ich [Grä22].

4.2 Halbringsemantik für CTL

Erweiterung der Halbringsemantik für FO auf CTL. Informationen sollen aus [DGNT19] und [LLM05] stammen.

4.3 Vergleich von Halbringsemantik für CTL mit PCTL

Unterschiede im Ansatz der beiden Varianten. Falls sich diese einfach beheben lassen, Vergleich in der Nutzung, der Ausdruckskraft etc.

5 Verwandte Arbeiten

Andere Ansätze die entweder nur Zeit oder Wahrscheinlichkeiten zu CTL bzw. Fixpunktlogiken hinzufügen sollen hier erläutert werden.

5.1 Erweiterung von CTL durch Echtzeit

Ein anderer Ansatz zum Ergänzen durch Zeit ist die in [ACD90] definierte Logik, welche erläutert werden soll. Interessant ist auch die Logik aus [JM86], welche eine Fixpunkt-Logik, also ausdrucksstärker als CTL ist. Ein Vergleich mit PCTL bzgl. der Ausdruckskraft soll folgen, evtl. mit Beschränkung von p auf extreme Werte, also $p \in \{0, 1\}$ für PCTL.

5.2 Erweiterung von CTL durch Wahrscheinlichkeiten

Es gibt auch Logiken die nur Wahrscheinlichkeiten hinzufügen. Lassen sich unterschiedliche Ansätze finden? Lösen diese andere Probleme? Paper mit anderen probabilistischen Logiken (die CTL erweitern): [HS84], [LS82] und [CC92].

6 Konklusion

Hier fasse ich noch einmal kurz die Ergebnisse zusammen. Also was PCTL ist, wofür es benutzt werden kann und evtl. wie Halbringsemantik ein ähnliches Ergebnis erzielen kann.

Literatur

[ACD90] Rajeev Alur, Costas Courcoubetis und David Dill. Model-checking for real-time systems. In *[1990] Proceedings. Fifth Annual IEEE Symposium on Logic in Computer Science*, Seiten 414–425. IEEE, 1990.

[BK08] Christel Baier und Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[CC92] Linda Christoff und Ivan Christoff. Reasoning about safety and liveness properties for probabilistic processes. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, Seiten 342–355. Springer, 1992.

[CE82] Edmund M Clarke und E Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Logics of Programs: Workshop, Yorktown Heights, New York, May 1981*, Seiten 52–71. Springer, 1982.

[CES86] Edmund M Clarke, E Allen Emerson und A Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263, 1986.

[DGNT19] Katrin M Dannert, Erich Grädel, Matthias Naaf und Val Tannen. Generalized absorptive polynomials and provenance semantics for fixed-point logic. *arXiv preprint arXiv:1910.07910*, 2019.

[Grä22] Erich Grädel. Provenance Analysis and Semiring Semantics for Logic and Games. 2022.

[GT17] Erich Grädel und Val Tannen. Semiring provenance for first-order model checking. *arXiv preprint arXiv:1712.01980*, 2017.

[HJ94] Hans Hansson und Bengt Jonsson. A logic for reasoning about time and reliability. *Formal aspects of computing*, 6:512–535, 1994.

[HS84] Sergiu Hart und Micha Sharir. Probabilistic temporal logics for finite and bounded models. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, Seiten 1–13, 1984.

[JM86] Farnam Jahanian und Aloysius Ka-Lau Mok. Safety analysis of timing properties in real-time systems. *IEEE Transactions on software engineering*, (9):890–904, 1986.

[LLM05] Alberto Lluch-Lafuente und Ugo Montanari. Quantitative μ-calculus and CTL defined over constraint semirings. *Theoretical Computer Science*, 346(1):135–160, 2005.

[LS82] Daniel Lehmann und Saharon Shelah. Reasoning with time and chance. *Information and Control*, 53(3):165–198, 1982.