

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 $a \in AP$ eine CTL-Formel. Außerdem sind \top und \perp 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 $EX \varphi_1$ und $AX \varphi_1$ CTL-Formeln.
4. Wenn φ_1 und φ_2 CTL-Formeln sind, dann sind $E[\varphi_1 U \varphi_2]$ und $A[\varphi_1 U \varphi_2]$ 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 $\mathcal{L} : V \rightarrow 2^{AP}$ ist, die jedem Zustand eine Menge an atomaren Aussagen zuweist [CE82, CES86].

Weiter bezeichnen wir einen Pfad als $\sigma = s_0 \dots s_n$ mit $s_i \in V$ für alle $i \in \{0, \dots, n\}$ 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. $v \models \top \Leftrightarrow \mathcal{K} \models \forall x(x = x)$ und $v \models \perp \Leftrightarrow \mathcal{K} \models \exists x(x \neq x)$.
2. Für $a \in AP$ 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 $v \models EX \varphi \Leftrightarrow$ es ex. ein $w \in V$ mit $(v, w) \in E$ und $w \models \varphi$ und analog:
 $v \models AX \varphi \Leftrightarrow$ für alle $w \in V$ mit $(v, w) \in E$ gilt $w \models \varphi$.
5. Es gilt $v \models E[\varphi_1 U \varphi_2] \Leftrightarrow$ 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 $\sigma[j] \models \varphi_1$. Analog ist die Definition für $A[\varphi_1 U \varphi_2]$, 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 $EX \varphi$ also, dass φ in einem beliebigen Nachfolgezustand gelten muss und $AX \varphi$, dass φ von allen Nachfolgezuständen erfüllt wird.

$E[\varphi_1 U \varphi_2]$ bzw. $A[\varphi_1 U \varphi_2]$ 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]:

- $EF \varphi \equiv E[\top \cup \varphi]$ und $AF \varphi \equiv A[\top \cup \varphi]$ bedeuten intuitiv, dass es einen Pfad gibt bzw. für alle Pfade irgendwann φ gilt.
- $EG \varphi \equiv \neg EF \neg \varphi$ und $AG \varphi \equiv \neg AF \neg \varphi$ 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 $AP = \{\text{idle}, \text{error}\}$

- $AG AF \text{idle}$
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.
- $EF AG \text{error}$
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 $\top \in \text{PCTL}$ und $\perp \in \text{PCTL}$.
2. Wenn AP die Menge atomarer Aussagen ist, dann ist jedes $a \in \text{AP}$ 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 $\varphi_1 U_{\geq p}^{\leq t} \varphi_2$ und $\varphi_1 U_{> p}^{\leq t} \varphi_2$ auch PCTL-Formeln.

Mit diesen Regeln können wir einige PCTL-Formeln aufstellen.

Beispiel 2 (Korrekte und inkorrekte PCTL-Formeln)

Sei $\text{AP} = \{A, B, X, Y\}$. Dann wären

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

korrekte PCTL-Formeln.

Inkorrekt gebildete Formeln wären zum Beispiel

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

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 $\mathcal{L} : S \rightarrow 2^{\text{AP}}$ sowie $\mathcal{T} : S \times S \rightarrow [0, 1]$ 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 $\text{paths}_{s_0}^{\mathfrak{S}}$ die Menge der Pfade in \mathfrak{S} , die in s_0 beginnen. Ein $\sigma \in \text{paths}_{s_0}^{\mathfrak{S}}$ ist dann von der Form $\sigma = s_0 s_1 s_2 \dots$ und wir definieren $\sigma[n] := s_n$ als den n -ten Zustand des Pfads und $\sigma \uparrow n := s_0 \dots s_n$ 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 $\sigma \in \text{paths}_{s_0}^{\mathfrak{S}}$ 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 $S = \{s_0, s_1, s_2\}$, $\mathcal{L} = \{s_0 \mapsto \{A\}, s_1 \mapsto \{A, B\}, s_2 \mapsto \{C\}\}$ und \mathcal{T} durch Tabelle 1 definiert. Dann ist $\mathfrak{S} = (S, s_0, \mathcal{T}, \mathcal{L})$ die in Abbildung 1 graphisch dargestellte Markov-Kette, wobei Transitionen mit einer Wahrscheinlichkeit von 0 nicht eingezeichnet werden.

\mathcal{T}	s_0	s_1	s_2
s_0	0	0.7	0.3
s_1	0.8	0	0.2
s_2	0	0	1

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

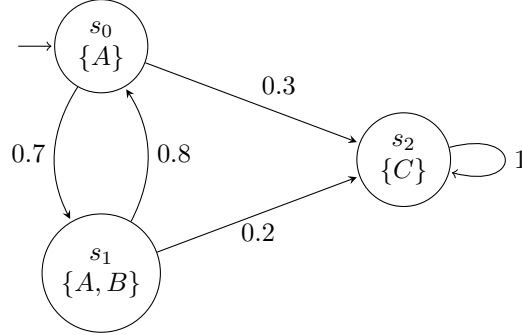


Abbildung 1: Graph für die Markov-Kette \mathfrak{S}

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 $X = \{\sigma \in \text{paths}_{s_0}^{\mathfrak{S}} : \sigma \upharpoonright n = \pi\}$ die Menge aller (unendlichen) Pfade, die mit dieser Folge beginnen. Wir definieren dann das Wahrscheinlichkeitsmaß

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

als das Produkt der Kanten zwischen den Zuständen der Folge. [HJ94]

Weiter ergeben sich einige Gleichheiten [HJ94]:

- Wählt man eine leere Folge mit Länge ergibt sich

$$\mu_{s_0}^{\mathfrak{S}}(\{\sigma \in \text{paths}_{s_0}^{\mathfrak{S}} : \sigma \upharpoonright 0 = s_0\}) = 1. \quad (1)$$

- Für eine abzählbare Menge $(X_i)_{i \in I}$ mit paarweise disjunkten Mengen an Pfaden gilt

$$\mu_{s_0}^{\mathfrak{S}}\left(\bigcup_{i \in I} X_i\right) = \sum_{i \in I} \mu_{s_0}^{\mathfrak{S}}(X_i). \quad (2)$$

- Sei $X \subseteq \text{paths}_{s_0}^{\mathfrak{S}}$ dann folgt

$$\mu_{s_0}^{\mathfrak{S}}(\text{paths}_{s_0}^{\mathfrak{S}} \setminus X) = 1 - \mu_{s_0}^{\mathfrak{S}}(X). \quad (3)$$

Betrachten wir zum verdeutlichen ein paar Beispiele.

Beispiel 4 (Wahrscheinlichkeitsmaße)

Der Einfachheit halber verwenden wir wieder die Markov-Kette aus Abbildung 1. Sei

$\pi = s_0 s_1 s_0 s_2$. Dann ist $X = \{\sigma \in \text{paths}_{s_1}^{\mathfrak{S}} : \sigma \uparrow 3 = \pi\}$ die Menge der Pfade, die durch folgenden Ausdruck dargestellt werden: $s_0 s_1 s_0 (s_2)^\omega$ und es folgt

$$\mu_{s_0}^{\mathfrak{S}}(X) = \mathcal{T}(s_0, s_1) \cdot \mathcal{T}(s_1, s_0) \cdot \mathcal{T}(s_0, s_2) = 0.7 \cdot 0.8 \cdot 0.3 = 0.168$$

Als zweites Beispiel wollen wir alle Pfade betrachten, die dem Ausdruck $s_0 (s_1 s_0)^* s_2$ entsprechen. Also alle Pfade, die endlich oft zwischen s_0 nach s_1 und wieder zurück wechseln und dann in s_2 enden. Es gilt:

$$\begin{aligned} & \mu_{s_0}^{\mathfrak{S}}(\{\sigma \in \text{paths}_{s_0}^{\mathfrak{S}} : \exists i \in \mathbb{N} (\sigma \uparrow 2i + 1 = s_0 (s_1 s_0)^i s_2)\}) \\ &= \mu_{s_0}^{\mathfrak{S}} \left(\bigcup_{i \in \mathbb{N}} \{\sigma \in \text{paths}_{s_0}^{\mathfrak{S}} : \sigma \uparrow 2i + 1 = s_0 (s_1 s_0)^i s_2\} \right) \\ &= \sum_{i \in \mathbb{N}} \mu_{s_0}^{\mathfrak{S}}(\{\sigma \in \text{paths}_{s_0}^{\mathfrak{S}} : \sigma \uparrow 2i + 1 = s_0 (s_1 s_0)^i s_2\}) \\ &= \sum_{i \in \mathbb{N}} (\mathcal{T}(s_0, s_1) \cdot \mathcal{T}(s_1, s_0))^i \cdot \mathcal{T}(s_0, s_2) \\ &= \sum_{i \in \mathbb{N}} 0.56^i \cdot 0.3 = 0.3 \cdot \sum_{i \in \mathbb{N}} 0.56^i \\ &= 0.3 \cdot \frac{1}{1 - 0.56} \approx 0,6818 \end{aligned}$$

Zuletzt soll ausgerechnet werden wie hoch die Wahrscheinlichkeit ist, niemals s_2 zu erreichen. Dies sind die genau die Pfade, die nur zwischen s_0 und s_1 hin-und-her wechseln. Ein alternativer Weg ist es, das Wahrscheinlichkeitsmaß der Pfade zu berechnen, die endlich von zwischen s_0 und s_1 wechseln und dann von s_1 aus in s_2 enden, also dem Ausdruck $s_0 s_1 (s_0 s_1)^* s_2$ entsprechen. Die Berechnung verläuft analog zu dem eben vorgeführten Beispiel. Das Ergebnis ist dann $0.14 \cdot \frac{1}{1 - 0.56} \approx 0.3182$. Es fällt auf, dass die Summe der letzten beiden Werte 1 ist. Fasst man beide Berechnungen zusammen erhält man demnach, dass die Wahrscheinlichkeit endlich oft zwischen s_0 und s_1 zu wechseln und irgendwie in s_2 zu enden gleich 1 ist. Mit Gleichung 3 ist die Wahrscheinlichkeit nie in s_2 zu enden also 0.

Nun können wir die Semantik von PCTL definieren. Genauer definieren wir eine Relation $\models_{\mathfrak{S}}$ für eine Markov-Kette $\mathfrak{S} = (S, s_i, \mathcal{T}, \mathcal{L})$, ein $s \in S$ und eine PCTL-Formel φ . Zur Vereinfachung definieren wir noch die Relation $\models_{\mathfrak{S}} \subseteq \text{paths}_s^{\mathfrak{S}} \times X$, wobei X die Menge der Ausdrücke von der Form $\varphi_1 U^{\leq t} \varphi_2$ mit $\varphi_1, \varphi_2 \in \text{PCTL}$, $t \in \mathbb{N}$ ist.

Definition 3.4 (Semantik von PCTL auf Markov-Ketten)

Sei $\mathfrak{S} = (S, s_i, \mathcal{T}, \mathcal{L})$ eine Markov-Kette und $s_0 \in S$. Dann lässt sich die Semantik für PCTL-Formeln induktiv definieren [HJ94]:

1. $s_0 \models_{\mathfrak{S}} \top \Leftrightarrow \mathfrak{S} \models \forall x (x = x)$ und $s_0 \models_{\mathfrak{S}} \perp \Leftrightarrow \mathfrak{S} \models \exists x (x \neq x)$
2. Für $a \in \text{AP}$ gilt $s_0 \models_{\mathfrak{S}} a \Leftrightarrow a \in \mathcal{L}(s_0)$

3. $s_0 \models_{\mathcal{E}} \neg \varphi \Leftrightarrow \text{nichts } s_0 \models_{\mathcal{E}} \varphi$
4. $s \models_{\mathcal{E}} \varphi_1 \wedge \varphi_2 \Leftrightarrow s_0 \models_{\mathcal{E}} \varphi \text{ und } s_0 \models_{\mathcal{E}} \varphi_2$
5. $\sigma \models \varphi_1 U^{\leq t} \varphi_2 \Leftrightarrow \exists i \leq t (\sigma[i] \models_{\mathcal{E}} \varphi_2 \text{ und } \sigma[j] \models_{\mathcal{E}} \varphi_1 \text{ für alle } 0 \leq j < i)$
6. $s_0 \models_{\mathcal{E}} \varphi_1 U_{\geq p}^{\leq t} \varphi_2 \Leftrightarrow \mu_{s_0}^{\mathcal{E}}(\{\sigma \in \text{paths}_{s_0}^{\mathcal{E}} : \sigma \models \varphi_1 U^{\leq t} \varphi_2\}) \geq p$
7. $s_0 \models_{\mathcal{E}} \varphi_1 U_{> p}^{\leq t} \varphi_2 \Leftrightarrow \mu_{s_0}^{\mathcal{E}}(\{\sigma \in \text{paths}_{s_0}^{\mathcal{E}} : \sigma \models \varphi_1 U^{\leq t} \varphi_2\}) > p$

Es fällt auf, dass im Operator $U_{\geq p}^{\leq t}$, der Parameter t Zeiteinheiten beschreibt, wobei eine Zeiteinheit genau eine Transition in der Markov-Kette ist. Analog beschreibt p die Wahrscheinlichkeit, dass ein gültiger Pfad „genommen“ wird.

Zusätzlich können wir weitere Operatoren definieren, um Formeln zu vereinfachen. Wie bekannt sind \neg und \wedge funktional vollständig, weshalb wir die bekannten booleschen Operatoren wie folgt definieren können:

- $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$
- $\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$.

Zusätzlich definieren wir den *außer* Operator \mathcal{U} [HJ94]:

- $\varphi_1 \mathcal{U}_{\geq p}^{\leq t} \varphi_2 \equiv \neg(\neg\varphi_1 U_{\geq 1-p}^{\leq t} \neg(\varphi_1 \vee \varphi_2))$
- $\varphi_1 \mathcal{U}_{> p}^{\leq t} \varphi_2 \equiv \neg(\neg\varphi_1 U_{\geq 1-p}^{\leq t} \neg(\varphi_1 \vee \varphi_2))$.

Intuitiv bedeutet $\varphi_1 \mathcal{U}_{\geq p}^{\leq t} \varphi_2$, dass mit einer Wahrscheinlichkeit von mindestens p entweder φ_1 für t Zeiteinheiten gilt, oder innerhalb von t Zeiteinheiten φ_2 erfüllt wird und bis dahin φ_1 gilt. Die Bedeutung von $\varphi_1 \mathcal{U}_{> p}^{\leq t} \varphi_2$ ist analog. [HJ94]

Bevor wir nun Beispiele betrachten wollen, soll noch ein Vergleich mit CTL stattfinden, welcher uns zusätzliche, sehr nützliche Operatoren liefern wird. Wie in Kapitel 2 erläutert wurde, besitzt CTL weder Aussagen über Zeit, noch Wahrscheinlichkeiten. Demnach können wir a priori einschränken, dass $t = \infty$ gelten muss und $p \in \{0, 1\}$, abhängig davon, ob ein A oder E verwendet wird. Man kann folgende Äquivalenzen finden [HJ94]:

- $A[\varphi_1 U \varphi_2] \equiv \varphi_1 U_{\geq 1}^{\leq \infty} \varphi_2$
- $E[\varphi_1 U \varphi_2] \equiv \varphi_1 U_{> 0}^{\leq \infty} \varphi_2$
- $AF \varphi \equiv \top U_{\geq 1}^{\leq \infty} \varphi$
- $EF \varphi \equiv \top U_{> 0}^{\leq \infty} \varphi$
- $AG \varphi \equiv \varphi U_{\geq 1}^{\leq \infty} \perp$
- $EG \varphi \equiv \varphi U_{> 0}^{\leq \infty} \perp$

Intuitiv haben die Operatoren die aus Kapitel 2 bekannten Bedeutungen. Durch die Fähigkeiten von PCTL lassen sich die vier unären Operatoren verallgemeinern. Betrachtet man die Formeln $G_{\geq p}^{\leq t} \varphi \equiv \varphi \mathcal{U}_{\geq p}^{\leq t} \perp$ und $F_{\geq p}^{\leq t} \varphi \equiv \top \mathcal{U}_{\geq p}^{\leq t} \varphi$ fällt einem auf, dass diese eben die Bedeutung von EG / AG und EF / AF für arbiträre $t \in \mathbb{N} \cup \{\infty\}$ und $p \in [0, 1]$ ausweiten. $G_{\geq p}^{\leq t} \varphi$ gilt also genau dann, wenn die Wahrscheinlichkeit, dass φ für t Zeiteinheiten mindestens p ist und $F_{\geq p}^{\leq t} \varphi$ bedeutet, dass mit einer Wahrscheinlichkeit von p , φ innerhalb von t Zeiteinheiten gelten wird. [HJ94]

In der Beschreibung von Systemen sind bedingte Aussagen wie zum Beispiel „Wenn A passiert, wird B passieren“ sehr hilfreich. Daher soll ein Operator, der solch ein Verhalten, verbunden mit Zeit- und Wahrscheinlichkeitsparametern, für PCTL definiert werden:

$$\varphi_1 \rightsquigarrow_{\geq p}^{\leq t} \varphi_2 \equiv \text{AG}(\varphi_1 \rightarrow F_{\text{seqp}}^{\leq t} \varphi_2),$$

mit der intuitiven Bedeutung, dass wenn φ_1 gilt, dann wird mit einer Wahrscheinlichkeit von mindestens p , in t Schritten φ_2 erfüllt. [HJ94]

Nun können wir Ausdrücke formulieren, die interessante Eigenschaften von Systemen beschreiben. Beginnen wir zuerst mit einigen abstrakten Beispielen.

Beispiel 5 (Abstrakte Beispiele für PCTL-Formeln)

Wir betrachten wieder die Markov-Kette aus Abbildung 1.

Die wir nun vertraut mit der Funktionsweise von PCTL sind, wollen wir das algorithmische Lösen des Model-Checking Spiels für PCTL-Formeln und Markov-Ketten betrachten.

3.2 Das Model-Checking Spiel für PCTL

Ein wichtiges Anwendungsgebiet für temporale Logiken wie CTL und PCTL ist die Verifikation von Systemen. Bei diesem erhält man eine Struktur \mathfrak{S} (im Fall von PCTL eine Markov-Kette) und eine PCTL-Formel φ , welche das gewünschte Verhalten eines Systems beschreibt. Nun überprüft man ob $\mathfrak{S} \models \varphi$ gilt und falls dies der Fall ist, wurde somit gezeigt, dass das durch \mathfrak{S} modellierte System sich korrekt verhält. Da komplexe Systeme aber nur durch sehr große Strukturen modelliert werden können und sich kompliziertes Verhalten auch nicht mit kurzen Formeln dargestellt werden kann, ist es sinnvoll dieses Modellierungsproblem zu automatisieren, indem man Model Checking Algorithmen entwickelt. Die Algorithmen aus [HJ94] sollen demnach in diesem Kapitel vorgestellt werden.

Sei $\mathfrak{S} = (S, s_i, \mathcal{T}, \mathcal{L})$ also eine Markov-Kette und φ eine PCTL-Formel. Der Algorithmus wird induktiv über den Formelaufbau definiert, indem für jede Formel zuerst ihre Subformeln ausgewertet werden. Dafür definieren wir eine Funktion $\text{label} : S \rightarrow 2^{\text{PCTL}}$, welche jedem Zustand aus \mathfrak{S} eine Menge an Formeln zuweist, die in diesem Zustand gelten. Am Anfang des Algorithmus wird $\text{label}(s) := \mathcal{L}(s) \cup \{\top\}$ für alle $s \in S$ gesetzt. Damit haben wir den Induktionsanfang bestimmt. Sei φ nun ein PCTL-Formel, die entweder der Regel 3 oder Regel 4 aus Definition 3.1 entspricht.

- Falls $\varphi = \neg\psi$ für eine PCTL-Formel ψ , dann können wir nach Induktionsvoraussetzung annehmen, dass ψ bereits für \mathfrak{S} ausgewertet wurde. Wir legen für alle $s \in S$ fest, dass $\text{label}(s) := \text{label}(s) \cup \{\varphi\}$, wenn $\psi \notin \text{label}(s)$.
- Falls $\varphi = \varphi_1 \wedge \varphi_2$, dann aktualisieren wir $\text{label}(s) := \text{label}(s) \cup \{\varphi\}$, wenn $\varphi_1 \in \text{label}(s)$ und $\varphi_2 \in \text{label}(s)$.
- Die Fälle $\varphi = \varphi_1 \bigcup_{\geq p}^{\leq t} \varphi_2$ und $\varphi = \varphi_1 \bigcup_{> p}^{\leq t} \varphi_2$ sind sehr viel komplexer und werden daher explizit im nächsten Abschnitt betrachtet.

Da nach Definition 3.1 zu jeder PCTL-Formel eine äquivalente Formel gebildet werden kann, die nur die obigen Operatoren verwendet, fehlt nur noch zu zeigen, wann $\text{label}(s)$ angepasst wird, im Fall der temporalen Operatoren. [HJ94]

Zur Vereinfachung teilen wir den Fall der temporalen Operatoren in drei Fälle auf, abhängig vom Parameter t . Die drei Fälle sind $t = 0$, $0 < t < \infty$ und $t = \infty$. Der Fall $t = 0$ ist trivial, da sich dann $s \models \varphi_1 \bigcup_{\geq p}^{\leq 0} \varphi_2$ offensichtlich zu $s \models \varphi_2$ vereinfachen lässt (und analog für $\bigcup_{> p}^{\leq 0}$). Die anderen beiden Fälle sollen nun aber weiter betrachtet werden, beginnend mit $0 < t < \infty$.

Sei $\varphi := \varphi_1 \bigcup_{\geq p}^{\leq t} \varphi_2$, $0 < t < \infty$ und die Zustände in denen φ_1 und φ_2 gelten wurden bereits bestimmt. Betrachtet man die Definition der Semantik für $\bigcup_{\geq p}^{\leq t}$ fällt einem auf, dass eine gesonderte Relation für Pfade und Formeln der Form $\varphi_1 \bigcup^{\leq t} \varphi_2$ definiert wurde. Daher definieren wir auch hier eine neue Funktion. Sei $\mathcal{P} : \mathbb{N} \times S \rightarrow [0, 1]$. Dann ist $\mathcal{P}(t, s)$ das Wahrscheinlichkeitsmaß für die Menge der Pfade σ , die in s beginnen und für die $\sigma \models \varphi_1 \bigcup^{\leq t} \varphi_2$ gilt. Wie in [HJ94] bewiesen erfüllt \mathcal{P} folgende Rekursionsgleichung:

$$\mathcal{P}(t, s) = \begin{cases} 1 & \text{wenn } \varphi_2 \in \text{label}(s) \\ 0 & \text{wenn } \varphi_1 \notin \text{label}(s) \\ \sum_{s' \in S} \mathcal{T}(s, s') \cdot \mathcal{P}(t-1, s') & \text{sonst} \end{cases} \quad (4)$$

Dies liefert unmittelbar Algorithmus 1 zum Berechnen von \mathcal{P} .

Ein anderer Ansatz zum berechnen von \mathcal{P} lässt sich mithilfe von Matrizen finden. Dafür partitionieren wir die Zustandsmenge $S = \{s_1, \dots, s_n\}$ in drei Teilmengen S_s , S_f und S_i auf mit den Bedeutungen, dass in S_s die Erfolgzzustände sind, anfangs also die $s \in S$ mit $\varphi_2 \in \text{label}(s)$, S_f sind die abgelehnten Zustände, also die $s \in S$ mit $\varphi_1, \varphi_2 \notin \text{label}(s)$. S_i sind die restlichen Zustände über die noch keine Aussage getroffen werden kann, genauer also die $s \in S$ mit $\varphi_1 \in \text{label}(s)$ und $\varphi_2 \notin \text{label}(s)$. [HJ94]

Nun definieren wir eine $|S| \times |S|$ -Matrix M

$$M[s_k, s_l] = \begin{cases} \mathcal{T}(s_k, s_l) & \text{wenn } s_k \in S_i \\ 1 & \text{wenn } s_k \notin S_i \wedge k = l \\ 0 & \text{sonst} \end{cases}$$

und für jedes $t \in \mathbb{N}$ einen Spaltenvektor $\overline{\mathcal{P}}(t)$ der Größe $|S|$ so, dass für das i -te Element $\overline{\mathcal{P}}(t)_i = \mathcal{P}(t, s_i)$ gilt. Für $t = 0$ gilt insbesondere also, dass $\overline{\mathcal{P}}(0)_i = 1$ wenn $s_i \in S_s$ und

Algorithmus 1 Übersetzung der Rekursionsgleichung 4 in Pseudo-Code [HJ94]

```
for  $i = 0, \dots, t$  do
  for all  $s \in S$  do
    if  $\varphi_2 \in \text{label}(s)$  then
       $\mathcal{P}(i, s) \leftarrow 1$ 
    else if  $\varphi_1 \notin \text{label}(s)$  then
       $\mathcal{P}(i, s) \leftarrow 0$ 
    else
       $\mathcal{P}(i, s) \leftarrow 0$ 
      for all  $s' \in S$  do
         $\mathcal{P}(i, s) \leftarrow \mathcal{P}(i, s) + \mathcal{T}(s, s') \cdot \mathcal{P}(i-1, s')$ 
      end for
    end if
  end for
end for
```

0 sonst. Nach dem Beweis in [HJ94] gilt dann, dass

$$\overline{\mathcal{P}}(t) = \overbrace{M \times M \times \dots \times M}^t \times \overline{\mathcal{P}}(0) = M^t \times \overline{\mathcal{P}}(0). \quad (5)$$

Diese Gleichung liefert direkt Algorithmus 2 zum Berechnen von \mathcal{P} . [HJ94]

Algorithmus 2 Algorithmus zum Berechnen von \mathcal{P} mithilfe der Gleichung 5 [HJ94]

```
for all  $s_i \in S$  do
  if  $\varphi_2 \in \text{label}(s)$  then
     $\overline{\mathcal{P}}(0)_i \leftarrow 1$ 
  else
     $\overline{\mathcal{P}}(0)_i \leftarrow 0$ 
  end if
end for
 $\overline{\mathcal{P}}(t) = M^t \times \overline{\mathcal{P}}(0)$ 
```

Damit haben wir nun zwei Algorithmen für das Berechnen von Wahrscheinlichkeiten in Bezug auf $U_{\geq p}^{\leq t}$ (bzw. $U_{> p}^{\leq t}$). Betrachten wir arithmetische Operationen fällt auf, dass Algorithmus 1 eine Zeitkomplexität von $\mathcal{O}(t \cdot |S|^2)$ bzw. falls $|E|$ die Anzahl der Transitionen in \mathcal{T} mit einem Wert größer 0 sind von $\mathcal{O}(t \cdot (|S| + |E|))$. Da zum Berechnen von M^t insgesamt $\log t$ Matrixmultiplikationen benötigt werden, welche jeweils in $\mathcal{O}(|S|^3)$ sind, ergibt sich für Algorithmus 2 eine Zeitkomplexität von $\mathcal{O}(\log t \cdot |S|^3)$. Es folgt, dass bei großen Werten für t , Algorithmus 2 sehr viel effizienter ist, während bei kleinen t aber sehr vielen Zuständen 1 schneller ist. [HJ94]

Falls $t = \infty$ ergeben sich für beide bisher vorgestellten Algorithmen einige Probleme, da Algorithmus 1 nicht terminieren würde und M^t im Algorithmus 2 nicht definiert ist. Daher müssen wir die Algorithmen anpassen. Betrachten wir die Partitionierung von S ,

welche wir zur Definition der Matrix M verwendet haben. Sei nun R die Menge der Erfolgszustände, also die, in denen φ_2 gilt. Um die abgelehnten Zustände zu identifizieren reicht es nun nicht mehr nur die betrachten, in denen φ_1 nicht gilt, da ein noch unentschiedener Zustand in S_i , von dem kein Erfolgszustand erreichbar ist auch abgelehnt werden muss. Dies liefert uns die Definition der Menge Q als die Knoten, in denen φ_1 nicht gilt oder denen, in denen φ_1 gilt, φ_2 nicht gilt und von denen kein Knoten in R erreichbar ist. Die restlichen Knoten, in denen φ_1 gilt, φ_2 nicht gilt und von denen ein Knoten in R erreichbar ist befinden sich wieder in der Menge S . [HJ94]

Da R einfach zu bestimmen ist und auch S_i direkt aus den Definitionen von R und Q folgt benötigen wir einen Algorithmus zum identifizieren von Q . Algorithmus 3 erreicht genau das, wobei er die Knotenmengen S_s und S_i aus Algorithmus 2 verwendet.

Algorithmus 3 Algorithmus zum Bestimmen von Q [HJ94]

```

ungesehen  $\leftarrow S_i \cup S_s$ 
rand  $\leftarrow S_s$ 
markiert  $\leftarrow \emptyset$ 
for  $i = 0, \dots, |S_i|$  do
    markiert  $\leftarrow$  markiert  $\cup$  rand
    ungesehen  $\leftarrow$  ungesehen  $\setminus$  rand
    rand  $\leftarrow \{s \mid s \in \text{ungesehen} \wedge \exists s' \in \text{rand} : \mathcal{T}(s, s') > 0\}$ 
end for
 $Q := S \setminus \text{markiert}$ 

```

Der Algorithmus markiert alle Knoten, von denen ein Knoten in S_s erreichbar ist, wobei zu Beginn des i -ten Durchlaufs der Schleife in der Menge rand genau die Knoten sind, deren kürzester Pfad zu einem Knoten in S_s die Länge i hat. Die Menge Q ist dann genau das Komplement. Analog lässt sich ein Algorithmus aufstellen um R zu der Menge zu erweitern die nicht nur die Knoten enthält die φ_2 erfüllen, sondern auch die, dessen Wahrscheinlichkeitsmaß, einen Knoten in S_s zu erreichen ohne über einen Knoten in S_f zu gehen gleich 1 ist. Nun können wir, ähnlich zur Rekursionsgleichung 4, eine Rekursionsgleichung angeben:

$$\mathcal{P}(\infty, s) = \begin{cases} 1 & \text{wenn } s \in R \\ 0 & \text{wenn } s \in Q \\ \sum_{s' \in S} \mathcal{T}(s, s') \cdot \mathcal{P}(\infty, s') & \text{sonst} \end{cases} \quad (6)$$

Betrachtet man $\mathcal{P}(\infty, s)$ bildet sich ein Gleichungssystem mit $|S|$ unbekannten. Dieses lässt sich mithilfe des Gaußschen Eliminierungsverfahren mit einem Zeitaufwand von $\mathcal{O}((|S| - |Q| - |R|)^2 \cdot 81)$ berechnen. [HJ94]

Somit haben wir für die drei Fälle ein Verfahren, um die Modellbeziehung zu berechnen. Sei $\varphi = \varphi_1 \cup_{\geq p}^t \varphi_2$ (bzw. $\varphi = \varphi_1 \cup_{> p}^t \varphi_2$) und φ_1 , sowie φ_2 wurden bereits betrachtet. Nun befinden wir uns in einem der drei Fälle. Gehe für jedes $s \in S$ diese durch und aktualisiere $\text{label}(s)$:

- Falls $t = 0$: Setze $\text{label}(s) = \text{label}(s) \cup \{\varphi\}$, wenn $\varphi_2 \in \text{label}(s)$.

- Falls $0 < t < \infty$: Sei $s = s_i$ für ein $i \in \mathbb{N}$. Berechne mithilfe von Algorithmus 1 die Funktion \mathcal{P} oder mithilfe von Algorithmus 2 den Spaltenvektor $\bar{\mathcal{P}}(t)$. Setze $\text{label}(s) = \text{label}(s) \cup \{\varphi\}$, wenn $\mathcal{P}(t, s) = \bar{\mathcal{P}}(t)_i \geq p$ (bzw. $\mathcal{P}(t, s) = \bar{\mathcal{P}}(t)_i > p$).
- Falls $t = \infty$: Löse das durch $\{\mathcal{P}(\infty, s) : s \in S\}$ definierte Gleichungssystem und setze $\text{label}(s) = \text{label}(s) \cup \{\varphi\}$, wenn $\mathcal{P}(\infty, s) \geq p$ (bzw. $\mathcal{P}(\infty, s) > p$).

Somit haben wir die Induktion vom Anfang des Kapitels beendet und wir sind in der Lage, für beliebige PCTL-Formeln φ , welche die elementaren Operatoren verwenden, in polynomieller Zeit zu überprüfen, ob für ein gegebenes s aus einer Markov-Kette \mathcal{S} $s \models_{\mathcal{S}} \varphi$ gilt.

Es lassen sich aber noch einige Verbesserungen finden. Zum Einen haben wir die erweiterten Operatoren wie $\mathcal{U}_{\geq p}^{\leq t}$, AF, EG usw. nicht betrachtet, da sich diese ja auf die behandelten, grundlegenden Operatoren vereinfachen lassen. Vor allem im Fall von $\mathcal{U}_{\geq p}^{\leq t}$ und $\mathcal{U}_{> p}^{\leq t}$ erhöht dies aber die Länge der Formel drastisch, weshalb in [HJ94] für diese Fälle weitere Algorithmen gefunden werden können.

Auch gibt es noch weitere Sonderfälle für die Operatoren $\mathcal{U}_{\geq p}^{\leq t}$ und $\mathcal{U}_{> p}^{\leq t}$. Ist $p \in \{0, 1\}$, dann gibt es deutlich effizientere Algorithmen als die hier vorgestellten. Für endliche t muss entweder ein Pfad mit Länge höchstens t gefunden werden, der eben die *Until*-Eigenschaften erfüllt oder einer, der diese nicht erfüllt, diese lassen sich ebenfalls in [HJ94] finden. Für $t = \infty$ lassen sich normale CTL Model-Checking Algorithmen verwenden. Beispiele dafür gibt es in [BK08] oder [CE82].

3.3 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 \leadsto 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.