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, dann sind $\mathrm{E}[\varphi_1\,\mathrm{U}\,\varphi_2]$, $\mathrm{A}[\varphi_1\,\mathrm{U}\,\varphi_2]$, $\mathrm{E}[\varphi_1\,\mathrm{W}\,\varphi_2]$ und $\mathrm{A}[\varphi_1\,\mathrm{W}\,\varphi_2]$ 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 $\sigma = s_0 s_1 \ldots$ mit $s_i \in V$ für alle $i \in \{0, 1, \ldots, \}$ und $\sigma[i] = s_i$ für $i \in \mathbb{N}$, bzw. für endliche Pfade $\sigma = s_0 \ldots s_n$ 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 $v \models \bot \Leftrightarrow \mathcal{K} \models \exists x(x \neq x)$.

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.

6. Es gilt $v \models \mathrm{E}[\varphi_1\,\mathrm{W}\,\varphi_2] \Leftrightarrow$ es existiert ein (unendlicher) Pfad σ so, dass entweder ein $i \in \mathbb{N}$ existiert mit $\sigma[i] \models \varphi_2$ und für alle $0 \leq j < i$ gilt $\sigma[j] \models \varphi_1$, oder es gilt $\sigma[i] \models \varphi_1$ für alle $i \in \mathbb{N}$. Für $\mathrm{A}[\varphi_1\,\mathrm{W}\,\varphi_2]$ ist die Definition analog, bloß mit einem All- anstatt Existenzquantor.

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

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

$\mathcal{T}$	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 $\mathcal{T} : S \times S \to [0, 1]$

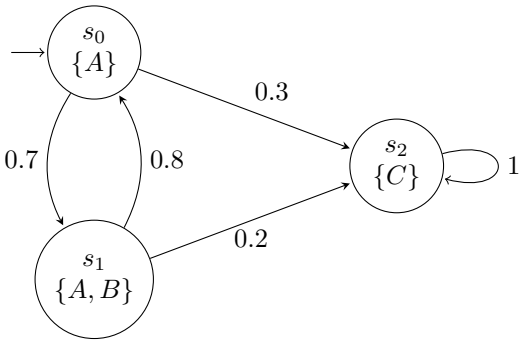

Abbildung 1: Graph für die Markov-Kette $\mathfrak{S}$

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

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

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

Weiter ergeben sich einige Gleichheiten [HJ94]:

- Wählt man eine leere Folge mit Länge ergibt sich

$$\mu^{\mathfrak{S}}_{s_0}(\{\sigma \in \text{paths}^{\mathfrak{S}}_{s_0} : \sigma \uparrow 0 = s_0\}) = 1. \tag{1}$$

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

$$\mu^{\mathfrak{S}}_{s_0}(\bigcup_{i \in I} X_i) = \sum_{i \in I} \mu^{\mathfrak{S}}_{s_0}(X_i). \tag{2}$$

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

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

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_4 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 meine 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 $\models\!\!\models_{\mathfrak{S}} \subseteq \text{paths}_s^{\mathfrak{S}} \times X$, wobei X die Menge der Ausdrücke von der Form $\varphi_1 \, \mathrm{U}^{\leq t} \, \varphi_2$ 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 $s_0 \models_{\mathfrak{S}} \bot \Leftrightarrow \mathfrak{S} \models \exists x(x \neq x)$
2. Für $a \in \mathsf{AP}$ gilt $s_0 \models_{\mathfrak{S}} a \Leftrightarrow a \in \mathcal{L}(s_0)$
3. $s_0 \models_{\mathfrak{S}} \neg\varphi \Leftrightarrow$ nicht$s_0 \models_{\mathfrak{S}} \varphi$
4. $s \models_{\mathfrak{S}} \varphi_1 \wedge \varphi_2 \Leftrightarrow s_0 \models_{\mathfrak{S}} \varphi$ und $s_0 \models_{\mathfrak{S}} \varphi_2$
5. $\sigma \not\models \varphi_1 \,\mathrm{U}^{\leq t}\, \varphi_2 \Leftrightarrow \exists i \leq t(\sigma[i] \models_{\mathfrak{S}} \varphi_2$ und $\sigma[j] \models_{\mathfrak{S}} \varphi_1$ für alle $0 \leq j < i)$
6. $s_0 \models_{\mathfrak{S}} \varphi_1 \,\mathrm{U}^{\leq t}_{\geq p}\, \varphi_2 \Leftrightarrow \mu^{\mathfrak{S}}_{s_0}(\{\sigma \in \mathrm{paths}^{\mathfrak{S}}_{s_0} : \sigma \not\models \varphi_1 \,\mathrm{U}^{\leq t}\, \varphi_2\}) \geq p$
7. $s_0 \models_{\mathfrak{S}} \varphi_1 \,\mathrm{U}^{\leq t}_{> p}\, \varphi_2 \Leftrightarrow \mu^{\mathfrak{S}}_{s_0}(\{\sigma \in \mathrm{paths}^{\mathfrak{S}}_{s_0} : \sigma \not\models \varphi_1 \,\mathrm{U}^{\leq t}\, \varphi_2\}) > p$

Es fällt auf, dass im Operator $\mathrm{U}^{\leq t}_{\geq p}$, 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]:

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

Intuitiv bedeutet $\varphi_1 \mathcal{U}^{\leq t}_{\geq p} \varphi_2$, 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 $\varphi_1 \mathcal{U}^{\leq t}_{> p} \varphi_2$ 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]:

- $\mathrm{A}[\varphi_1 \,\mathrm{U}\, \varphi_2] \equiv \varphi_1 \,\mathrm{U}^{\leq \infty}_{\geq 1}\, \varphi_2$
- $\mathrm{E}[\varphi_1 \,\mathrm{U}\, \varphi_2] \equiv \varphi_1 \,\mathrm{U}^{\leq \infty}_{> 0}\, \varphi_2$
- $\mathrm{AF}\, \varphi \equiv \top \,\mathrm{U}^{\leq \infty}_{\geq 1}\, \varphi$
- $\mathrm{EF}\, \varphi \equiv \top \,\mathrm{U}^{\leq \infty}_{> 0}\, \varphi$

- $\mathrm{AG}\,\varphi \equiv \varphi \mathcal{U}^{\leq\infty}_{\geq 1} \bot$
- $\mathrm{EG}\,\varphi \equiv \varphi \mathcal{U}^{\leq\infty}_{>0} \bot$

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 $\mathrm{G}^{\leq t}_{\geq p}\,\varphi \equiv \varphi \mathcal{U}^{\leq t}_{\geq p} \bot$ und $\mathrm{F}^{\leq t}_{\geq p}\,\varphi \equiv \top \mathrm{U}^{\leq t}_{\geq p}\,\varphi$ 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. $\mathrm{G}^{\leq t}_{\geq p}\,\varphi$ gilt also genau dann, wenn die Wahrscheinlichkeit, dass φ für t Zeiteinheiten mindestens p ist und $\mathrm{F}^{\leq t}_{\geq p}\,\varphi$ 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:

$$\varphi_1 \rightsquigarrow^{\leq t}_{\geq p} \varphi_2 \equiv \mathrm{AG}(\varphi_1 \rightarrow \mathrm{F}^{\leq t}_{\geq p}\,\varphi_2),$$

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 (Beispiele für PCTL-Formeln und deren Syntax)
Betrachte die Formeln aus Beispiel 2. Die erste ist $\varphi = \neg(X \wedge \neg(\top \mathrm{U}^{\leq 15}_{\geq 90\%}\, Y))$. Es fällt schnell auf, dass $\neg(X \wedge \neg(\top \mathrm{U}^{\leq 15}_{\geq 90\%}\, Y)) \equiv X \rightarrow (\top \mathrm{U}^{\leq 15}_{\geq 90\%}\, Y) \equiv X \rightarrow \mathrm{F}^{\leq 15}_{\geq 90\%}\, Y$. Intuitiv bedeutet $s \models_{\mathfrak{S}} \varphi$ also, dass wenn $s \models_{\mathfrak{S}} X$, dann folgt, dass innerhalb von 15 Zeiteinheiten mit einer Wahrscheinlichkeit von 90%, Y gelten wird.

Die zweite Formel aus Beispiel 2 ist $\psi = (A \mathrm{U}^{\leq 10}_{\geq 90\%}\, B) \wedge (A \mathrm{U}^{\leq 20}_{>95\%}\, B)$. Die Bedeutung ist intuitiv hier, dass mit einer Wahrscheinlichkeit von mindestens 90% A für 10 Schritte und dann B gilt und mit einer Wahrscheinlichkeit von mehr als 95% gilt A für 20 Schritte und danach B.

Für ein paar weitere Beispiele und Erkenntnisse soll die Markov-Kette $\mathfrak{S} = (\{s_0, s_1, s_2\}, s_0, \mathcal{T}, \mathcal{L})$ aus Abbildung 1 betrachtet werden. Um Wahrscheinlichkeitsmaße zu erläutern, wurden zwei interessante Maße berechnet. Zum Einen die Wahrscheinlichkeit, dass von s_0 irgendwann zu s_2 gewechselt wird, und die analoge Wahrscheinlichkeit für den Wechsel zu s_2 von s_1 aus. Es fällt auf, dass die Summe der Wahrscheinlichkeiten 1, die Wahrscheinlichkeit, nie s_2 zu erreichen, ist also 0. Es folgt also $s_0 \models_{\mathfrak{S}} \neg \mathrm{EG}\, A$ bzw. $s_0 \models_{\mathfrak{S}} A \rightsquigarrow^{\leq\infty}_{\geq 1} C$.

Da 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 $\mathrm{label} : S \to 2^{\mathsf{PCTL}}$, welche jedem Zustand aus $\mathfrak{S}$ eine Menge an Formeln zuweist, die in diesem Zustand gelten. Am Anfang des Algorithmus wird $\mathrm{label}(s) \coloneqq \mathcal{L}(s) \cup \{\top\}$ 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 $\mathrm{label}(s) \coloneqq \mathrm{label}(s) \cup \{\varphi\}$, wenn $\psi \notin \mathrm{label}(s)$.
- Falls $\varphi = \varphi_1 \wedge \varphi_2$, dann aktualisieren wir $\mathrm{label}(s) \coloneqq \mathrm{label}(s) \cup \{\varphi\}$, wenn $\varphi_1 \in \mathrm{label}(s)$ und $\varphi_2 \in \mathrm{label}(s)$.
- Die Fälle $\varphi = \varphi_1 \,\mathrm{U}^{\leq t}_{\geq p}\, \varphi_2$ und $\varphi = \varphi_1 \,\mathrm{U}^{\leq t}_{> p}\, \varphi_2$ 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 $\mathrm{label}(s)$ 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 $s \models \varphi_1 \,\mathrm{U}^{\leq 0}_{\geq p}\, \varphi_2$ offensichtlich zu $s \models \varphi_2$ vereinfachen lässt (und analog für $\mathrm{U}^{\leq 0}_{> p}$). Die anderen beiden Fälle sollen nun aber weiter betrachtet werden, beginnend mit $0 < t < \infty$.

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

die $\sigma \models \varphi_1 \, \mathrm{U}^{\leq t} \, \varphi_2$ gilt. Wie in [HJ94] bewiesen erfüllt $\mathcal{P}$ folgende Rekursionsgleichung:

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

wobei $t > 0$. Für $t = 0$ definieren $\mathcal{P}(0,s)$ wie folgt:

$$\mathcal{P}(0,s) = \begin{cases} 1 & \text{wenn } \varphi_2 \in \mathrm{label}(s) \\ 0 & \text{sonst} \end{cases} \tag{5}$$

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

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

```
for i = 0, ..., t do
    for all s ∈ S do
        if φ2 ∈ label(s) then
            P(i, s) ← 1
        else if φ1 ∉ label(s) then
            P(i, s) ← 0
        else
            P(i, s) ← 0
            if i > 1 then
                for all s' ∈ S do
                    P(i, s) ← P(i, s) + T(s, s') · P(i − 1, s')
                end for
            end if
        end if
    end for
end for
```

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

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

$$\overline{\mathcal{P}}(t) = \overbrace{M \times M \times \cdots \times M}^{t} \times \overline{\mathcal{P}}(0) = M^t \times \overline{\mathcal{P}}(0). \tag{6}$$

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

Algorithmus 2 Algorithmus zum Berechnen von $\mathcal{P}$ mithilfe der Gleichung 6 [HJ94]

```
for all s_i ∈ S do
    if φ_2 ∈ label(s) then
        P̄(0)_i ← 1
    else
        P̄(0)_i ← 0
    end if
end for
P̄(t) = M^t × P̄(0)
```

Damit haben wir nun zwei Algorithmen für das Berechnen von Wahrscheinlichkeiten in Bezug auf $\mathrm{U}^{\leq t}_{\geq p}$ (bzw. $\mathrm{U}^{\leq t}_{>p}$). 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.

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

Algorithmus 3 Algorithmus zum Bestimmen von Q [HJ94]

ungesehen $\leftarrow S_i \cup S_s$
rand $\leftarrow S_s$
markiert $\leftarrow \emptyset$
for $i = 0, \ldots, |S_i|$ **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
$Q \coloneqq S \setminus$ markiert

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 Gleichung 4, eine Rekursionsgleichung angeben:

$$\mathcal{P}(\infty, s) = \begin{cases} 1 & \text{wenn } s \in R \\ 0 & \text{wenn } s \in Q \\ \sum_{s' \in S} \mathcal{T}(s, s') \cdot \mathcal{P}(\infty, s') & \text{sonst} \end{cases} \tag{7}$$

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

Somit haben wir für die drei Fälle ein Verfahren, um die Modellbeziehung zu berechnen. Sei $\varphi = \varphi_1 \, \mathrm{U}^{\leq t}_{\geq p} \, \varphi_2$ (bzw. $\varphi = \varphi_1 \, \mathrm{U}^{\leq t}_{> p} \, \varphi_2$) 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 $\mathrm{label}(s)$:

- Falls $t = 0$: Setze $\mathrm{label}(s) = \mathrm{label}(s) \cup \{\varphi\}$, wenn $\varphi_2 \in \mathrm{label}(s)$.
- 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 $\overline{\mathcal{P}}(t)$. Setze $\mathrm{label}(s) = \mathrm{label}(s) \cup \{\varphi\}$, wenn $\mathcal{P}(t, s) = \overline{\mathcal{P}}(t)_i \geq p$ (bzw. $\mathcal{P}(t, s) = \overline{\mathcal{P}}(t)_i > p$).
- Falls $t = \infty$: Löse das durch $\{\mathcal{P}(\infty, s) : s \in S\}$ definierte Gleichungssystem und setze $\mathrm{label}(s) = \mathrm{label}(s) \cup \{\varphi\}$, 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 $\mathfrak{S}$ $s \models_{\mathfrak{S}} \varphi$ gilt.

Es lassen sich aber noch einige Verbesserungen finden. Zum Einen haben wir die erweiterten Operatoren wie $\mathcal{U}^{\leq t}_{\geq p}$, AF, EG usw. nicht betrachtet, da sich diese ja auf die behandelten, grundlegenden Operatoren vereinfachen lassen. Vor allem im Fall von $\mathcal{U}^{\leq t}_{\geq p}$ und $\mathcal{U}^{\leq t}_{> p}$ 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 $\mathrm{U}_{\geq p}^{\leq t}$ und $\mathrm{U}_{>p}^{\leq t}$. 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 Model-Checking Beispiel

Sei $\mathsf{AP} = \{\mathsf{running}, \mathsf{stopped}, \mathsf{warning}, \mathsf{error}\}$. Weiter definieren wir die Markov-Kette $\mathfrak{S} = (S = \{s_0, s_1, s_2\}, s_0, \mathcal{T}, \mathcal{L})$, wobei $\mathcal{T}$ die durch Tabelle 2 definierte Funktion ist und $\mathcal{L} = \{s_0 \mapsto \{\mathsf{running}\}, s_1 \mapsto \{\mathsf{stopped}, \mathsf{warning}\}, s_2 \mapsto \{\mathsf{stopped}, \mathsf{error}\}\}$ gilt.

$\mathcal{T}$	s_0	s_1	s_2
s_0	0.95	0	0.05
s_1	0.4	0.5	0.1
s_2	0	0.4	0.6

Tabelle 2: Tabelle zur Definition der Funktion $\mathcal{T} : S \times S \to [0, 1]$ für das Model-Checking Beispiel

Dadurch ergibt sich dann der Graph aus Abbildung 2.

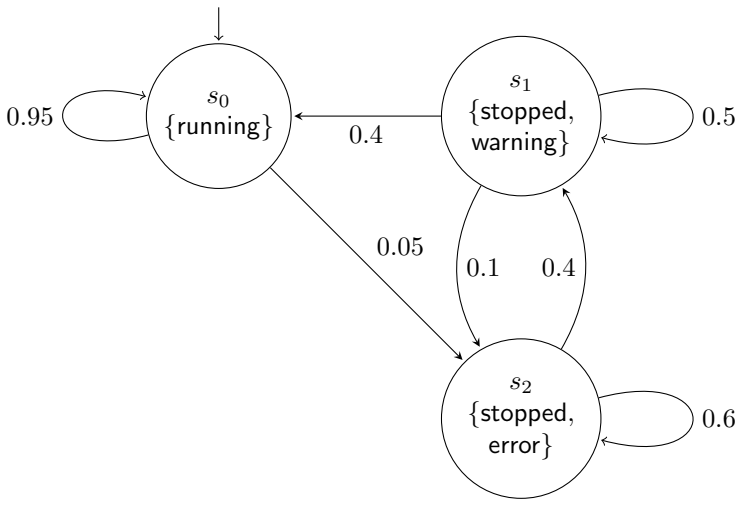

Abbildung 2: Graph für die Markov-Kette $\mathfrak{S}$ des Model-Checking Beispiels

Die Markov-Kette beschreibt ein beliebiges System. Beim Auftreten eines Fehlers, was in jedem Zeitschritt mit einer Wahrscheinlichkeit von 5% passieren kann, wird in den

Zustand s_2 gewechselt, welcher den Fehlerzustand darstellt, und das System stoppt den Betrieb. Mit einer Wahrscheinlichkeit von 40% wird das System repariert, wodurch es in den Zustand s_1 wechselt. In diesem hat das System noch immer nicht seinen Betrieb wiederaufgenommen, dies passiert erst, wenn der Fehler quittiert wurde, was jeden Zeitschritt mit einer Wahrscheinlichkeit von 40% passieren kann.

Betrachten wir zu Beginn die Formel $\varphi = \mathrm{F}^{\leq 2}_{\geq 60\%}\,\mathsf{running}$. Mithilfe der Model-Checking Algorithmen aus Kapitel 3.2 wollen wir nun überprüfen, von welchen Zuständen aus $\mathfrak{S}$ φ erfüllt wird. Nach den Definitionen aus Kapitel 3 gilt $\varphi \equiv \top\,\mathrm{U}^{\leq 2}_{\geq 60\%}\,\mathsf{running}$. Offensichtlich gilt $\top \in \mathrm{label}(s)$ für alle $s \in S$ und $\mathsf{running} \in \mathrm{label}(s_0)$. Benutzen wir zuerst Algorithmus 1. Wir erhalten dadurch die Funktion, die in Tabelle 3 gesehen werden kann.

$\mathcal{P}(i,s)$	$s = s_0$	$s = s_1$	$s = s_2$
$i = 0$	1	0	0
$i = 1$	1	0.4	0
$i = 2$	1	$0.4 \cdot 1 + 0.5 \cdot 0.4 + 0.1 \cdot 0 = 0.6$	0.16

Tabelle 3: Tabelle für die induktive Definition der Funktion $\mathcal{P} : \mathbb{N} \times S \to [0,1]$

Damit erhalten wir also, dass $s_0 \models_{\mathfrak{S}} \varphi$ und $s_1 \models_{\mathfrak{S}} \varphi$ gelten, da $P(2,s_0) > P(2,s_1) \geq 0.6$.

Selbiges wollen wir nun auch mit Algorithmus 2 berechnen. Es gilt $\overline{\mathcal{P}}(0) = (1,0,0)^{\mathrm{T}}$, da $\mathsf{running} \in \mathrm{label}(s_0)$, aber $\mathsf{running} \notin \mathrm{label}(s_1)$ bzw. $\mathsf{running} \notin \mathrm{label}(s_2)$. Nun müssen wir die Matrix M definieren. Es gilt

$$M \coloneqq \begin{pmatrix} 1 & 0 & 0 \\ 0.4 & 0.5 & 0.1 \\ 0 & 0.4 & 0.6 \end{pmatrix}$$

und dann $M^2 \cdot \overline{\mathcal{P}}(0) = \overline{\mathcal{P}}(2) = (1, 0.6, 0.16)^{\mathrm{T}}$. Wir erhalten also die gleichen Werte wie mit Algorithmus 1.

Als zweites Beispiel wollen wir die Formel $\psi = \mathrm{EG}\,\mathsf{running}$ betrachten. Wieder formulieren wir eine äquivalente Formel mit den Operatoren, für die wir Model-Checking Algorithmen definiert haben. Es gilt $\psi \equiv \mathsf{running}\,\mathcal{U}^{\leq \infty}_{>0}\,\bot \equiv \neg(\neg\mathsf{running}\,\mathrm{U}^{\leq \infty}_{\geq 1}\,\neg\mathsf{running})$. Definiere θ so, dass $\neg(\neg\mathsf{running}\,\mathrm{U}^{\leq \infty}_{\geq 1}\,\neg\mathsf{running}) = \neg(\theta)$ und werte zuerst θ für alle $s \in S$ aus. Die einzige Subformel in θ ist $\neg\mathsf{running}$ und es gilt offensichtlich $\neg\mathsf{running} \in \mathrm{label}(s)$ genau dann, wenn $s \neq s_0$. Offensichtlich gilt $R = \{s_1, s_2\}$ und $s_0 \notin Q$. Damit erhalten wir das Gleichungssystem

$$\begin{aligned}
\mathcal{P}(\infty, s_0) &= \mathcal{T}(s_0,s_0) \cdot \mathcal{P}(\infty,s_0) + \mathcal{T}(s_0,s_1) \cdot \mathcal{P}(\infty,s_1) + \mathcal{T}(s_0,s_2) \cdot \mathcal{P}(\infty,s_2) \\
&= 0.95 \cdot \mathcal{P}(\infty,s_0) + 0 \cdot \mathcal{P}(\infty,s_1) + 0.05 \cdot \mathcal{P}(\infty,s_2) \\
\mathcal{P}(\infty,s_1) &= 1 \\
\mathcal{P}(\infty,s_2) &= 1
\end{aligned}$$

bzw. in Matrixschreibweise

$$\left(\begin{array}{ccc|c} 0.95 \cdot \mathcal{P}(\infty, s_0) & 0 & 0.05 \cdot \mathcal{P}(\infty, s_2) & \mathcal{P}(\infty, s_0) \\ 0 & 1 \cdot \mathcal{P}(\infty, s_1) & 0 & 1 \\ 0 & 0 & 1 \cdot \mathcal{P}(\infty, s_2) & 1 \end{array}\right)$$

$$\rightsquigarrow \left(\begin{array}{ccc|c} -0.05 \cdot \mathcal{P}(\infty, s_0) & 0 & 0.05 \cdot \mathcal{P}(\infty, s_2) & 0 \\ 0 & 1 \cdot \mathcal{P}(\infty, s_1) & 0 & 1 \\ 0 & 0 & 1 \cdot \mathcal{P}(\infty, s_2) & 1 \end{array}\right)$$

$$\rightsquigarrow \begin{pmatrix} -0.05 & 0 & 0.05 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathcal{P}(\infty, s_0) \\ \mathcal{P}(\infty, s_1) \\ \mathcal{P}(\infty, s_2) \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}.$$

Löst man dieses Gleichungssystem erhält man $\mathcal{P}(\infty, s_0) = \mathcal{P}(\infty, s_1) = \mathcal{P}(\infty, s_2) = 1$. Und es ergibt sich, dass $s \models_{\mathfrak{S}} \theta$ für jedes $s \in S$ gilt. Nach Definition von θ folgt also, dass $s \not\models \psi$ für alle $s \in S$, insbesondere $s_0 \not\models \psi$. Damit haben wir gezeigt, dass es keinen Pfad gibt, der immer nur in s_0 bleibt. Über eine unbeschränkte Laufzeit ist es also unmöglich, dass das System immer läuft.

Mit diesen beiden Beispielen wurden die drei Algorithmen aus Kapitel 3.2 genauer erklärt und besprochen. Insgesamt ist nun also bekannt was PCTL ist, wofür es verwendet werden kann, die PCTL-Formeln gebildet werden und was diese bedeuten und wie man algorithmisch auswertet, ob eine gegebene Markov-Kette ein Modell einer PCTL-Formel ist. Im Rest dieser Arbeit soll PCTL mit anderen Logiken verglichen werden, um zu betrachten, wie sinnvoll und zielführend der in [HJ94] verwendete Ansatz ist.

4 Verwandte Arbeiten

Da nun die Logik PCTL bekannt ist sollen einige alternative Ansätze bzw. Logiken diskutiert werden, die entweder zeitliche Aspekte oder Wahrscheinlichkeiten hinzufügen. Zuerst sollen zwei Logiken betrachtet werden, die Zeit implementieren. Eine davon unterscheidet sich von PCTL dadurch, dass Zeit nicht mehr diskret durch Hochzählen von Transitionen behandelt wird, sondern reelle Werte dafür verwendet werden [ACD90]. Die Andere ist eine Verallgemeinerung über Propositionale Temporale Logik PTL und verwendet einen sogenannten *Einfrier-Operator* [AH94]. Nachdem diese Logiken miteinander und mit PCTL verglichen wurden, sollen noch Wahrscheinlichkeiten betrachtet werden. Dafür werden Logiken vorgeführt, welche es ermöglichen Wahrscheinlichkeiten auszudrücken.

4.1 Logiken mit Echtzeit

In diesem Kapitel werden die Logiken *Timed Computation Tree Logic* (TCTL) und *Timed Propositional Temporal Logic* (TPTL) anhand ihrer Syntax sowie Semantik erörtert und ihre Model-Checking Möglichkeit diskutiert. Zusätzlich werden diese dann jeweils mit PCTL verglichen.

4.1.1 Timed Computation Tree Logic

Die Logik TCTL stellt, ähnlich wie PCTL, eine Möglichkeit zur Verfügung, um zeitliche Zusammenhänge auszudrücken. Während dies in PCTL aber nur mit diskreten Werten möglich ist, verwendet TCTL reelle Werte, wodurch sich Systeme potentiell besser beschreiben lassen. Da dies aber offensichtlicher weise komplexer, als bloß bei jeder verwendeten Transition einen Zähler zu erhöhen, wird eine neue Struktur benötigt, welche Systeme modellieren soll und über welchen TCTL-Formeln ausgewertet werden. Dafür definieren wir eine feste Menge $\mathcal{N}$, welche uns Werte für zeitliche Vergleiche in Formeln gibt. Der Einfachheit halber definieren wir hier $\mathcal{N} \coloneqq \{0, 1, \dots\} = \mathbb{N}$. Da es aber offensichtlich eine Bijektion zwischen $\mathbb{N}$ und $\mathbb{Q}$ gibt, lässt sich auch $\mathcal{N} \coloneqq \mathbb{Q}$ festlegen, wodurch sich die Vorteile einer dichten Ordnung[1] im Vergleich zu PCTL ausnutzen lassen. [ACD90]

Definition 4.1 (Zeitliche Graphen)
Ein *zeitlicher Graph* ist ein Tupel $\mathfrak{S} = (S, s_0, E, C, \pi, \tau, \mu)$ mit folgenden Definitionen [ACD90]:

- S ist eine endliche Menge an Knoten.
- $s_0 \in S$ ist der Startzustand.
- $E \subseteq S \times S$ ist die Kantenrelation.
- C ist eine endliche Menge an Uhren.
- $\pi : E \to 2^C$ ist eine Abbildung, die jeder Kante eine Menge an Uhren zuweist.
- τ ist eine Abbildung, die jeder Kante eine Formel zuweist, die aus Booleschen Junktoren über den atomaren Formeln $x \leq c$ und $c \leq x$ besteht, wobei $c \in C$ und $x \in \mathcal{N}$ gilt.
- Für eine Menge an atomaren Aussagen AP ist $\mu : S \to 2^{\mathsf{AP}}$ eine Abbildung, die jedem Zustand eine Menge an atomaren Aussagen zuweist.

Ein zeitlicher Graph besitzt also eine Menge an sogenannten Uhren C. Diese zählen unabhängig voneinander hoch so, dass jede Uhr zu jedem „Zeitpunkt“ einen reellen Wert $x \in \mathbb{R}_{\geq 0}$ speichert. Mithilfe von π lassen sich Uhren zurücksetzen, für jede Transition lässt sich also eine Menge an Uhren auswählen, welche mit dem Wechsel über die Transition zurückgesetzt werden. Zusätzlich existiert für jede Kante eine Formel, welche bestimmte Voraussetzungen an die Transition stellt.

Semantisch soll ein zeitlicher Graph so interpretiert werden, dass eine Kante $e \in E$ nur dann genommen werden kann, wenn $\tau(e)$ zum aktuellen Zeitpunkt erfüllt wird. Alle Uhren zählen gleich schnell hoch, können aber natürlich unterschiedliche Werte besitzen, da das Zurücksetzen unabhängig voneinander passieren kann.

Um diese Definition besser zu verstehen, soll ein Beispiel angeführt werden.

[1] Eine dichte Ordnung ist eine lineare Ordnung so, dass zwischen je zwei Elementen ein drittes liegt.

Beispiel 6 (Beispiel für einen zeitlichen Graphen)
Sei $\mathsf{AP} = \{\mathsf{running}, \mathsf{stopped}, \mathsf{warning}, \mathsf{error}\}$ eine Menge an atomaren Aussagen und $\mathfrak{S} = (S, s_0, E, C, \pi, \tau, \mu)$ ein zeitlicher Graph mit

- $S \coloneqq \{s_0, s_1, s_2\}$
- $E \coloneqq \{(s_0, s_2), (s_1, s_0), (s_1, s_1), (s_1, s_2), (s_2, s_1)\}$
- $C \coloneqq \{x, y\}$
- $\pi(u, v) = \begin{cases} \{x\} & \text{falls } v = s_2 \\ \{y\} & \text{falls } (u, v) = (s_1, s_0) \\ \emptyset & \text{sonst} \end{cases}$
- $\tau(u, v) = \begin{cases} \top & \text{falls } v = s_2 \\ x \geq 2 & \text{falls } u = s_2 \\ x \geq 3 & \text{sonst} \end{cases}$
- $\mu(s) = \begin{cases} \{\mathsf{running}\} & \text{falls } s = s_0 \\ \{\mathsf{stopped}, \mathsf{warning}\} & \text{falls } s = s_1 \\ \{\mathsf{stopped}, \mathsf{error}\} & \text{falls } s = s_2 \end{cases}$

Dann lässt sich die graphische Darstellung von $\mathfrak{S}$ in Abbildung 3 sehen[2].

Es lässt sich also sehen, dass die Uhr x die zeit stoppt, die seit dem Letzten Fehler passiert ist, während die Uhr y die Zeit misst, die seit der Letzten Wiederaufnahme des Betriebs des Systems vergangen ist. Weiter gibt τ uns die Information, dass wenn ein Fehler auftritt, für zwei Zeiteinheiten der Fehler bestehen bleiben muss und die darauf folgende Fehlermeldung für mindestens drei Zeiteinheiten seit dem auftreten des Fehlers beibehalten wird.

Um nun TCTL zu definieren benötigen wir noch einige Begriffe und Strukturen. Für einen zeitlichen Graphen $\mathfrak{S}$ definiert $\Gamma(\mathfrak{S})$ die Menge aller möglichen Zuweisungen von Werten zu den Uhren, formal also $\Gamma(\mathfrak{S}) \coloneqq \{\nu \mid \nu : C \to \mathbb{R}_{\geq 0}\}$. Weiter schreiben wir für ein $\nu \in \Gamma(\mathfrak{S})$ und ein $t \in \mathbb{R}_{\geq 0}$, den Ausdruck $\nu + t$ für eine neue Zuweisung definiert als $(\nu + t)(x) \coloneqq \nu(x) + t$ und für ein $x \in C$ definieren wir die Schreibweise $[x \mapsto t]\nu$ als

$$([x \mapsto t]\nu)(y) \coloneqq \begin{cases} t & \text{falls } x = y \\ \nu(y) & \text{sonst} \end{cases}$$

Eine interessante Beobachtung ist die, dass die eine Konfiguration eines zeitlichen Graphens ein Paar aus Zustand und Uhr-Belegungen sind. Der aktuelle Stand eines zeitlichen Graphens $\mathfrak{S}$ lässt sich also eindeutig durch das Paar $\langle s, \nu \rangle$ charakterisieren, wobei s ein Zustand ist und $\nu \in \Gamma(\mathfrak{S})$ gilt. [ACD90]

Damit können wir Läufe durch zeitliche Graphen definieren, welche stetige Zustandswechsel erlauben. Es ist also möglich nicht nur zu diskreten Zeitpunkten einen Zustand zu

[2]Die Formel $\top$ wird als Platzhalter für eine beliebige Tautologie wie zum Beispiel $x \geq 2 \vee \neg(x \geq 2)$.

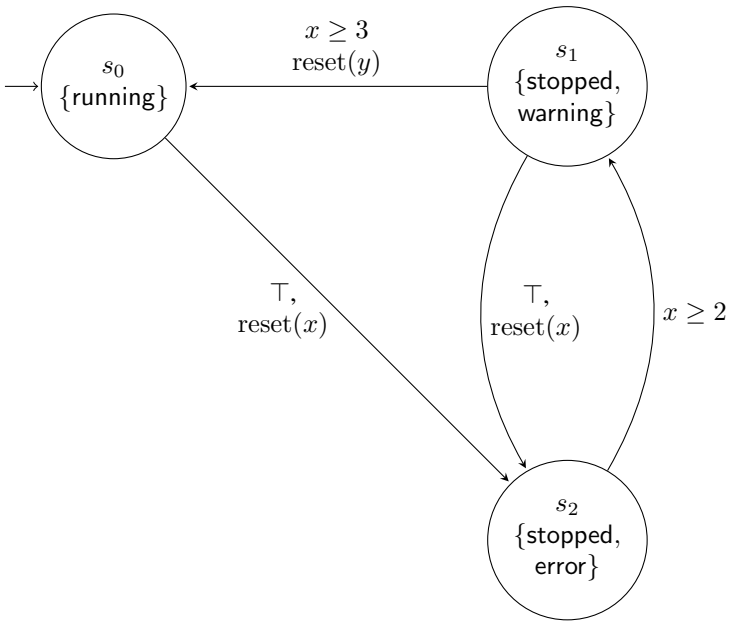

Abbildung 3: Die graphische Darstellung des zeitlichen Graphen $\mathfrak{S}$

wechseln, wodurch die in der Einleitung benannten Vorteile von TCTL verwendet werden können.

Definition 4.2 ($\langle s, \nu \rangle$-Lauf)
Sei $\langle s, \nu \rangle$ eine Konfiguration des zeitlichen Graphen $\mathfrak{S} = (S, s_0, E, C, \pi, \tau, \mu)$. Dann ist ein $\langle s, \nu \rangle$-Lauf eine unendliche Folge an Tripeln der Form

$$(\langle s, \nu, 0 \rangle = \langle s_1, \nu_1, t_1 \rangle, \langle s_2, \nu_2, t_2 \rangle, \dots) \in (S \times \Gamma(\mathfrak{S}) \times \mathbb{R}_{\geq 0})^\omega$$

so, dass folgende Regeln eingehalten werden [ACD90]:

1. Für alle $i \in \mathbb{N} \setminus \{0\}$ gilt $t_{i+1} > t_i$. Die Zeit-Werte steigen also streng monoton.
2. Für die Tripel $\langle s_i, \nu_i, t_i \rangle$ und $\langle s_{i+1}, \nu_{i+1}, t_{i+1} \rangle$ definieren wir $e_i = (s_i, s_{i+1})$ und es muss gelten $e_i \in E$.
3. Für alle $i \in \mathbb{N} \setminus \{0\}$ gilt $\nu_{i+1} = [\pi(e_i) \mapsto 0](\nu_i + t_{i+1} - t)$.
4. Für die boolesche Voraussetzung $\tau(e_i)$ ist $(\nu_i + t_{i+1} - t_i)$ eine erfüllende Belegung der Uhr-Werte.
5. Für jedes $t \in \mathbb{R}_{\geq 0}$ gibt es ein j so, dass $t_j \geq t$. Es gibt also keine obere Schranke für die Zeit-Werte.

Nun können wir TCTL definieren.

Definition 4.3 (Syntax von TCTL)
Sei AP eine Menge an atomaren Aussagen, $p \in \mathsf{AP}$ und $c \in \mathcal{N}$. Dann definieren die beiden folgenden Grammatiken die Logik TCTL:

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists\varphi \, \mathrm{U}_{\kappa c} \, \varphi \mid \forall\varphi \, \mathrm{U}_{\kappa c} \, \varphi$$
$$\kappa ::= < \mid \leq \mid = \mid \geq \mid >$$

Die Menge der TCTL-Formeln sind die ausdrücke, die von φ erzeugt werden. [ACD90]

Informell soll $\exists\varphi_1 \, \mathrm{U}_{\sim c} \, \varphi_2$ bedeuten, dass es einen Präfix gibt, der κc Zeiteinheiten lang ist und in dem φ_1 gilt, wobei im Zustand nach dem Präfix φ_2 gelten muss. Die Bedeutung von $\forall\varphi_1 \, \mathrm{U}_{\kappa c} \, \varphi_2$ ist analog.

Schließlich können wir die Semantik von TCTL definieren.

Definition 4.4 (Semantik von TCTL)
Sei $\mathfrak{S} = (S, s_0, E, C, \pi, \tau, \mu)$ ein zeitlicher Graph. Dann können wir die Modellrelation $\models$ zwischen Konfigurationen von zeitlichen Graphen und TCTL-Formeln induktiv definieren. Sei $s \in S$, $nu \in \Gamma(\mathfrak{S})$, $p \in \mathsf{AP}$, $c \in \mathcal{N}$ und $\sim \in \{<, \leq, =, \geq, >\}$. [ACD90]

- $\langle s, \nu \rangle \models p$ gdw. $p \in \mu(s)$
- $\langle s, \nu \rangle \models \varphi_1 \wedge \varphi_2$ gdw. $s \models \varphi_1$ und $s \models \varphi_2$
- $\langle s, \nu \rangle \models \neg\varphi$ gdw. $s \not\models \varphi$
- $\langle s, \nu \rangle \models \exists\varphi_1 \, \mathrm{U}_{\sim c} \, \varphi_2$ gdw. es einen $\langle s, \nu \rangle$-Lauf gibt, mit $t_i \sim c$, $s_i \models \varphi_2$ und $s_j \models \varphi_1$ für $1 \leq j < i$.
- $\langle s, \nu \rangle \models \forall\varphi_1 \, \mathrm{U}_{\sim c} \, \varphi_2$ gdw. für alle $\langle s, \nu \rangle$-Läufe gilt, dass es ein $i \in \mathbb{N}$ gibt, mit $t_i \sim c$, $s_i \models \varphi_2$ und für alle $1 \leq j < i$ gilt $s_j \models \varphi_1$.

Damit können wir bestimmen, wann eine Konfiguration eine TCTL-Formel erfüllt.

Beispiel 7 (Beispiel für das Auswerten von TCTL-Formeln)
Betrachte bspw. den zeitlichen Graphen aus Abbildung 3 und die Formel

$$\varphi := \exists \mathsf{stopped} \, \mathrm{U}_{<3} \, \mathsf{running},$$

die besagt, dass das Systems stoppt und in weniger als drei Zeiteinheiten in einen laufenden Zustand wechselt. Sei ν eine zeitliche Belegung mit $\nu(x) = 0$ und $\nu(y) \in \mathbb{R}_{\geq 0}$ beliebig. Dann stellt die Konfiguration $\alpha = \langle s_2, \nu \rangle$ dar, dass gerade in den Zustand s_2 gewechselt wurde, wodurch die Uhr x zurückgesetzt wurde. Wenn wir nun $\alpha \models \varphi$ betrachten, dann fällt auf, dass der einzige Pfad zu einem Zustand mit running der Pfad s_2, s_1, s_0 ist. Weiter ist die Transition (s_1, s_0) erst möglich, wenn $x \geq 3$ gilt. Da wir unsere Auswertung mit $\nu_1(x) = 0 = t_1$ beginnen, ist $\nu_i(x) = t_i$ für alle $i \geq 1$. Daraus ergibt sich aber auch, dass für alle $\langle s, \nu \rangle$-Läufe gilt, dass wenn $s_i = s_0$ gilt, $\nu_i(x) = t_i \geq 3$ gelten muss. Es folgt also $\alpha \not\models \varphi$.

Wie man an diesem Beispiel erkennen konnte, ist das Auswerten von TCTL-Formeln nicht trivial. Aus diesem Grund ist die algorithmische Betrachtung des Model-Checking Spiels für TCTL äußerst interessant und wird in [ACD90] sehr konkret besprochen. Da der vorgestellte Algorithmus aber um einiges komplexer ist, also die in Kapitel 3.2 diskutierten, soll hier bloß die grobe Idee angemerkt werden. Es lässt sich feststellen, dass im Kontext des Model-Checkings zwei Zeitbelegungen ν und ν' äquivalent sind ($\nu \cong \nu'$), wenn die abgerundeten Ganzzahlwerte und die Reihenfolge der Nachkomma-Werte gleich sind.

Formal verwenden wir $\lfloor x \rfloor$ für die größte ganze Zahl y mit $y \leq x$ und $\mathrm{frac}(x)$ für $x - \lfloor x \rfloor$. Weiter sei für $x \in C$ der Wert $c_x \in \mathcal{N}$ definiert als $c_x := \max\{c : c \leq x \text{ oder } x \leq c \text{ kommt in } \tau(e) \text{ vor}, e \in E \text{ bel.}\}$, also als größte Konstante, mit der x verglichen wird. Dann gilt für $\nu, \nu' \in \Gamma(\mathfrak{S})$, dass $\nu \cong \nu'$ genau dann, wenn

1. für alle $x \in C$ gilt, dass entweder $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ oder dann $\nu(x)$ und $\nu'(x)$ größer als c_x sind und
2. für alle $x, y \in C$ mit $\nu(x) \leq c_x$ und $\nu(y) \leq c_y$ gilt, dass $\mathrm{fract}(\nu(x)) \leq \mathrm{fract}(\nu(y))$ gdw. $\mathrm{fract}(\nu'(x)) \leq \mathrm{fract}(\nu'(y))$. [ACD90]

Damit lässt sich dann erkennen, dass für eine beliebige TCTL-Formel φ und zwei äquivalente $\nu, \nu' \in \Gamma(\mathfrak{S})$ gilt, dass $\langle s, \nu \rangle \models \varphi$ gdw. $\langle s, \nu' \rangle \models \varphi$. [ACD90]

Betrachtet man nun die Äquivalenzklassen $\{[\nu] : \nu \in \Gamma(\mathfrak{S})\}$ der soeben definierten Äquivalenzrelation erhält man Mengen an Konfigurationen $\langle s, [\nu] \rangle$, welche wir Regionen nennen. Schlussendlich lässt sich der Model-Checking Algorithmus dann über Pfade dieser Regionen definieren. [ACD90]

Der daraus resultierende Model-Checking Algorithmus hat eine Komplexität von

$$\mathcal{O}\left[c(\varphi) \cdot |\varphi| \cdot (|S| + |E|) \cdot |C|! \cdot \prod_{x \in C} c_x\right],$$

wobei $c(\varphi)$ die größte in φ vorkommende Konstante ist. Der Algorithmus wächst also linear in Formellänge als auch Graphengröße, aber wächst exponentiell mit der Anzahl an Uhren. [ACD90]

Dies stellt eine sehr viel höhere Komplexität dar, als die in Kapitel 3.2 vorgestellten Algorithmen. Zudem waren die PCTL-Algorithmen nicht optimiert für $p \in \{0, 1\}$, was hier ja aber der Fall ist. Offensichtlich ist die Art, wie PCTL Zeit implementiert sehr viel schwächer als TCTLs Implementation, jedoch ist diese Steigerung an Aussagekraft auch sehr klar in der Model-Checking Komplexität erkennbar. Ein Ausweiten von TCTL bzw. zeitlichen Graphen auf Markov-Ketten würde zusätzliche Komplikationen hinzufügen. So würde es bspw. nicht mehr nur ausreichen, $\mathcal{T}$ von einer Kante abhängig zu machen. Da es in einem zeitlichen Graphen nicht möglich ist, Kanten zu verwenden, deren boolesche Formeln nicht erfüllt werden, würde die das Wahrscheinlichkeitsmaß in diesen Fällen 0 sein müssen. Demnach wäre es nötig, $\mathcal{T}$ von der Kante, dem ausgehenden Knoten und einem zeitlichen Parameter abhängig zu machen.

Es lässt sich also erkennen, dass PCTL trotz der, bzgl. Zeit, geringeren Aussagekraft auch Vorteile gegenüber TCTL bietet.

4.2 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].

5 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.

[AH94] Rajeev Alur und Thomas A Henzinger. A really temporal logic. *Journal of the ACM (JACM)*, 41(1):181–203, 1994.

[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.

[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.

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