

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  $\varphi_1$  und  $\varphi_2$  CTL-Formeln sind, dann sind  $\neg\varphi_1$ ,  $\varphi_1 \wedge \varphi_2$  und  $\varphi_1 \vee \varphi_2$  ebenfalls CTL-Formeln.
3. Wenn  $\varphi_1$  eine CTL-Formel ist, dann sind auch  $EX \varphi_1$  und  $AX \varphi_1$  CTL-Formeln.
4. Wenn  $\varphi_1$  und  $\varphi_2$  CTL-Formeln sind, dann sind  $E[\varphi_1 U \varphi_2]$ ,  $A[\varphi_1 U \varphi_2]$ ,  $E[\varphi_1 W \varphi_2]$  und  $A[\varphi_1 W \varphi_2]$  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  $\mathcal{L} : V \rightarrow 2^{AP}$  ist, die jedem Zustand eine Menge an atomaren Aussagen zuweist [CE82, CES86].

Weiter bezeichnen wir einen Pfad als  $\sigma = s_0 s_1 \dots$  mit  $s_i \in V$  für alle  $i \in \{0, 1, \dots\}$  und  $\sigma[i] = s_i$  für  $i \in \mathbb{N}$ , bzw. für endliche Pfade  $\sigma = s_0 \dots s_n$  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  $v \models \perp \Leftrightarrow \mathcal{K} \models \exists x(x \neq x)$ .
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  $\sigma$  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  $\varphi_2$  und alle Zustände davor  $\varphi_1$  erfüllen.
6. Es gilt  $v \models E[\varphi_1 W \varphi_2] \Leftrightarrow$  es existiert ein (unendlicher) Pfad  $\sigma$  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  $A[\varphi_1 W \varphi_2]$  ist die Definition analog, bloß mit einem All- anstatt Existenzquantor.

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

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

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

- $EF \varphi \equiv E[\top U \varphi]$  und  $AF \varphi \equiv A[\top U \varphi]$  bedeuten intuitiv, dass es einen Pfad gibt bzw. für alle Pfade irgendwann  $\varphi$  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  $\varphi$  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  $\text{idle}$  erfüllt. Das heißt,  $\text{idle}$  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  $\text{error}$  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  $\varphi_1$  und  $\varphi_2$  PCTL-Formeln sind, dann sind  $\neg\varphi_1$  und  $(\varphi_1 \wedge \varphi_2)$  auch PCTL-Formeln.
4. Für zwei PCTL-Formeln  $\varphi_1$  und  $\varphi_2$ ,  $t \in \mathbb{N} \cup \{\infty\}$  und  $p \in [0, 1] \subseteq \mathbb{R}$ , sind  $\varphi_1 \text{ U}_{\geq p}^{\leq t} \varphi_2$  und  $\varphi_1 \text{ U}_{> p}^{\leq t} \varphi_2$  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

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

korrekte PCTL-Formeln.

Inkorrekt gebildete Formeln wären zum Beispiel

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

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  $\sigma \upharpoonright n := s_0 \dots s_n$  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.

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

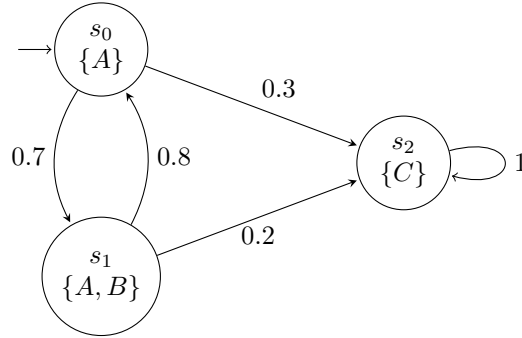


Abbildung 1: Graph für die Markov-Kette  $\mathfrak{S}$

**Definition 3.3** (Wahrscheinlichkeitsmaß)

Sei  $\pi = s_0 \dots s_n$  eine Folge an Zuständen einer Markov-Kette  $\mathfrak{S}$  und  $X = \{\sigma \in \text{paths}_{s_0}^{\mathfrak{S}} : \sigma \uparrow n = \pi\}$  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 Zustände der Folge. [HJ94]

Weiter ergeben sich einige Gleichheiten [HJ94]:

- Wählt man eine leere Folge mit Länge ergibt sich

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

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

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]:

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

Intuitiv bedeutet  $\varphi_1 \mathcal{U}_{\geq p}^{\leq t} \varphi_2$ , dass mit einer Wahrscheinlichkeit von mindestens  $p$  entweder  $\varphi_1$  für  $t$  Zeiteinheiten gilt, oder innerhalb von  $t$  Zeiteinheiten  $\varphi_2$  erfüllt wird und bis dahin  $\varphi_1$  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]:

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



- $\text{AG } \varphi \equiv \varphi \mathcal{U}_{\geq 1}^{\leq \infty} \perp$
- $\text{EG } \varphi \equiv \varphi \mathcal{U}_{\geq 0}^{\leq \infty} \perp$

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  $\varphi$  für  $t$  Zeiteinheiten mindestens  $p$  ist und  $F_{\geq p}^{\leq t} \varphi$  bedeutet, dass mit einer Wahrscheinlichkeit von  $p$ ,  $\varphi$  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:

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

mit der intuitiven Bedeutung, dass wenn  $\varphi_1$  gilt, dann wird mit einer Wahrscheinlichkeit von mindestens  $p$ , in  $t$  Schritten  $\varphi_2$  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  $\varphi = \neg(X \wedge \neg(\top \mathcal{U}_{\geq 90\%}^{\leq 15} Y))$ . Es fällt schnell auf, dass  $\neg(X \wedge \neg(\top \mathcal{U}_{\geq 90\%}^{\leq 15} Y)) \equiv X \rightarrow (\top \mathcal{U}_{\geq 90\%}^{\leq 15} Y) \equiv X \rightarrow F_{\geq 90\%}^{\leq 15} Y$ . 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  $\psi = (A \mathcal{U}_{\geq 90\%}^{\leq 10} B) \wedge (A \mathcal{U}_{\geq 95\%}^{\leq 20} B)$ . 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  $s_0 \models_{\mathfrak{S}} \neg \text{EG } A$  bzw.  $s_0 \models_{\mathfrak{S}} A \rightsquigarrow_{\geq 1}^{\leq \infty} C$ .

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  $\varphi$ , 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  $\varphi$  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  $\varphi$  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  $\psi$ , dann können wir nach Induktionsvoraussetzung annehmen, dass  $\psi$  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  $\varphi = \varphi_1 U_{\geq p}^{\leq t} \varphi_2$  und  $\varphi = \varphi_1 U_{> p}^{\leq t} \varphi_2$  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  $s \models \varphi_1 U_{\geq p}^{\leq 0} \varphi_2$  offensichtlich zu  $s \models \varphi_2$  vereinfachen lässt (und analog für  $U_{> p}^{\leq 0}$ ). Die anderen beiden Fälle sollen nun aber weiter betrachtet werden, beginnend mit  $0 < t < \infty$ .

Sei  $\varphi := \varphi_1 U_{\geq p}^{\leq t} \varphi_2$ ,  $0 < t < \infty$  und die Zustände in denen  $\varphi_1$  und  $\varphi_2$  gelten wurden bereits bestimmt. Betrachtet man die Definition der Semantik für  $U_{\geq p}^{\leq t}$  fällt einem auf, dass eine gesonderte Relation für Pfade und Formeln der Form  $\varphi_1 U_{\geq p}^{\leq t} \varphi_2$  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  $\sigma$ , die in  $s$  beginnen und für

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

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

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

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

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

---

**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
       $\mathcal{P}(i, s) \leftarrow 0$ 
      if  $i > 1$  then
        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 if
  end for
end for

```

---

Ein anderer Ansatz zum berechnen von  $\mathcal{P}$  lässt sich mithilfe von Matrizen finden. Dafür partitionieren wir die Zustandsmenge  $S = \{s_1, \dots, s_n\}$  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  $\varphi_2 \in \text{label}(s)$ ,  $S_f$  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

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

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

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  $s_i \in S$  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 Erfolgzustände, also die, in denen  $\varphi_2$  gilt. Um die abgelehnten Zustände zu identifizieren reicht es nun nicht mehr nur die betrachten, in denen  $\varphi_1$  nicht gilt, da ein noch unentschiedener Zustand in  $S_i$ , von dem kein Erfolgzustand erreichbar ist auch abgelehnt werden muss. Dies liefert uns die Definition der Menge  $Q$  als die Knoten, in denen  $\varphi_1$  nicht gilt oder denen, in denen  $\varphi_1$  gilt,  $\varphi_2$  nicht gilt und von denen kein Knoten in  $R$  erreichbar ist. Die restlichen Knoten, in denen  $\varphi_1$  gilt,  $\varphi_2$  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  $\text{rand}$  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  $\varphi_2$  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  $i = 0, \dots, |S_i|$  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
 $Q := S \setminus \text{markiert}$ 
```

---

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:

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

Betrachtet man  $\mathcal{P}(\infty, s)$  bildet sich ein Gleichungssystem mit  $|S|$  unbekannten. Dieses lässt sich mithilfe des Gaußschen Eliminationsverfahren 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  $\varphi = \varphi_1 \bigcup_{\geq p}^{\leq t} \varphi_2$  (bzw.  $\varphi = \varphi_1 \bigcup_{> p}^{\leq t} \varphi_2$ ) und  $\varphi_1$ , sowie  $\varphi_2$  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  $\overline{\mathcal{P}}(t)$ . Setze  $\text{label}(s) = \text{label}(s) \cup \{\varphi\}$ , 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  $\text{label}(s) = \text{label}(s) \cup \{\varphi\}$ , 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  $\varphi$ , 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  $\mathcal{U}_{\geq p}^{\leq t}$ , 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  $U_{\geq p}^{\leq t}$  und  $U_{> p}^{\leq t}$ . 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  $AP = \{\text{running}, \text{stopped}, \text{warning}, \text{error}\}$ . 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  $\mathcal{L} = \{s_0 \mapsto \{\text{running}\}, s_1 \mapsto \{\text{stopped}, \text{warning}\}, s_2 \mapsto \{\text{stopped}, \text{error}\}\}$  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  $\mathcal{T} : S \times S \rightarrow [0, 1]$  für das Model-Checking Beispiel

Dadurch ergibt sich dann der Graph aus Abbildung 2.

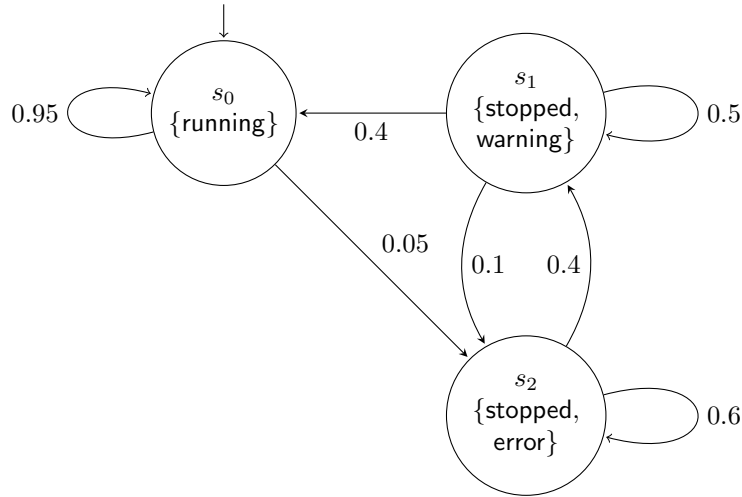


Abbildung 2: Graph für die Markov-Kette  $\mathfrak{S}$  des Model-Checking Beispiels

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  $\varphi = F_{\geq 60\%}^{\leq 2} \text{ running}$ . Mithilfe der Model-Checking Algorithmen aus Kapitel 3.2 wollen wir nun überprüfen, von welchen Zuständen aus  $\mathfrak{S}$   $\varphi$  erfüllt wird. Nach den Definitionen aus Kapitel 3 gilt  $\varphi \equiv \top \cup_{\geq 60\%}^{\leq 2} \text{ running}$ . Offensichtlich gilt  $\top \in \text{label}(s)$  für alle  $s \in S$  und  $\text{running} \in \text{label}(s_0)$ . 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  $\mathcal{P} : \mathbb{N} \times S \rightarrow [0, 1]$

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  $\overline{\mathcal{P}}(0) = (1, 0, 0)^T$ , da  $\text{running} \in \text{label}(s_0)$ , aber  $\text{running} \notin \text{label}(s_1)$  bzw.  $\text{running} \notin \text{label}(s_2)$ . Nun müssen wir die Matrix  $M$  definieren. Es gilt

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

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

Als zweites Beispiel wollen wir die Formel  $\psi = \text{EG running}$  betrachten. Wieder formulieren wir eine äquivalente Formel mit den Operatoren, für die wir Model-Checking Algorithmen definiert haben. Es gilt  $\psi \equiv \text{running} \mathcal{U}_{\geq 0}^{\leq \infty} \perp \equiv \neg(\neg \text{running} \cup_{\geq 1}^{\leq \infty} \neg \text{running})$ . Definiere  $\theta$  so, dass  $\neg(\neg \text{running} \cup_{\geq 1}^{\leq \infty} \neg \text{running}) = \neg(\theta)$  und werte zuerst  $\theta$  für alle  $s \in S$  aus. Die einzige Subformel in  $\theta$  ist  $\neg \text{running}$  und es gilt offensichtlich  $\neg \text{running} \in \text{label}(s)$  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

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

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  $s \models_{\infty} \theta$  für jedes  $s \in S$  gilt. Nach Definition von  $\theta$  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 Vergleich mit Halbringsemantik

Ein weiterer Ansatz um die Ausdrucksstärke einer Logik zu erweitern ist es, Formeln nicht nur zu *wahr* oder *falsch* auszuwerten, sondern einer Formel einen Wert aus einem Halbring zuzuordnen. Diese Methode wird als Halbringsemantik bezeichnet. Abhängig davon, in welcher Halbring verwendet wird, lassen sich unterschiedliche Anwendungen darstellen. So gibt es zum Beispiel den natürlichen Halbring  $\mathbb{N} = (\mathbb{N}, +, \cdot, 0, 1)$  zum Zählen, den tropischen Halbring  $\mathbb{T} = (\mathbb{R}_+^\infty := \mathbb{R} \cup \{0, \infty\}, \min, +, \infty, 0)$  für Kostenanalyse oder den Viterbi-Halbring  $\mathbb{V} = \{[0, 1], \max, \cdot, 0, 1\}$  für Vertrauensanalyse [DG19]. Letzterer soll im Weiteren behandelt und, verbunden mit CTL im Vergleich zu PCTL betrachtet werden. Dafür wird nun aber erst die Halbringsemantik für die Prädikatenlogik erster Ordnung (FO) besprochen.

### 4.1 Halbringsemantik für FO

Die Syntax als auch Semantik von FO wird als bekannt vorausgesetzt. Falls dies nicht der Fall ist, lassen sich die wichtigen Informationen in [Bar77] oder [EFTF94] nachlesen. Um die hier verwendete Notation zu erklären soll FO aber dennoch definiert werden.



**Definition 4.1** (Syntax von FO)

Eine Signatur  $\tau$  ist eine Menge an Relationen und Funktionen fester Stelligkeit. Dann ist die Menge der  $\tau$ -Terme  $\text{Terms}(\tau)$  definiert durch

- Wenn  $x$  eine Variable ist, dann ist  $x \in \text{Terms}(\tau)$
- Wenn  $a$  eine Funktion mit Stelligkeit 0 ist (also eine Konstante), dann gilt  $a \in \text{Terms}(\tau)$
- Wenn  $t_1, \dots, t_k$   $\tau$ -Terme sind und  $f$  eine  $k$ -stellige Funktion ist, dann gilt  $f(t_1 \dots t_k) \in \text{Terms}(\tau)$ .

Damit lässt sich nun die Menge der FO-Formeln definieren

1. Wenn  $t_1, t_2 \in \text{Terms}(\tau)$ , dann ist  $t_1 = t_2 \in \text{FO}$
2. Wenn  $t_1, \dots, t_k \in \text{Terms}(\tau)$ , und  $R$  ist eine  $k$ -stellige Relation in  $\tau$ , dann ist  $R(t_1 \dots t_k) \in \text{FO}$
3. Wenn  $\varphi \in \text{FO}$ , dann ist  $\neg\varphi \in \text{FO}$
4. Wenn  $\varphi_1, \varphi_2 \in \text{FO}$ , dann ist  $\varphi_1 \wedge \varphi_2, \varphi_1 \vee \varphi_2, \varphi_1 \rightarrow \varphi_2 \in \text{FO}$
5. Wenn  $x$  eine Variable ist und  $\varphi \in \text{FO}$ , dann ist  $\exists x\varphi(x) \in \text{FO}$  und  $\forall x\varphi(x) \in \text{FO}$

Formeln, die nur durch die Regeln 1 und 2 verwendet werden bezeichnen wir als atomar.

In diesem Kapitel soll die Halbringsemantik für FO betrachtet werden. Wie der Name schon sagt, wird eine alternative Semantik für FO verwendet werden. Aus diesem Grund soll die klassische Semantik hier nicht behandelt werden, Informationen dazu lassen sich ebenfalls in [Bar77] und [EFTF94] finden.

Bevor wir nun aber Halbringsemantik betrachten soll erklärt werden, was ein Halbring ist. Ein (kommutativer) Halbring ist eine Struktur  $\mathfrak{S} = (S, +, \cdot, 0, 1)$ , wobei  $0 \neq 1$ ,  $(S, +, 0)$ , als auch  $(S, \cdot, 1)$  (abelsche) Monoide sind,  $\cdot$  ist distributiv über  $+$  und  $0 \cdot a = a \cdot 0 = 0$  für alle  $a \in S$ . [DG19]

Jetzt können wir die Halbringsemantik für FO definieren. Sei  $A$  ein Universum und  $\tau$  eine relationale Signatur. Dann ist  $\text{Facts}_A(\tau) := \{Ra : R \in \tau, a \in A^{\text{st}(R)}\}^1$  und  $\text{Lit}_A(\tau) := \{\alpha, \neg\alpha : \alpha \in \text{Facts}_A(\tau)\}$ . Sei  $\mathfrak{S}$  nun ein kommutativer Halbring. Damit können wir eine  $\mathfrak{S}$ -Interpretation  $\pi : \text{Lit}_A(\tau) \rightarrow S$  definieren, welche jeder atomaren und negierten atomaren Formel einen Wert im Halbring zuweist. [DG19]

**Definition 4.2** (Halbringsemantik FO)

Sei  $A$  ein Universum,  $\tau$  eine Signatur und  $\pi$  eine  $\mathfrak{S}$ -Interpretation für einen Halbring  $\mathfrak{S}$ . Dann erweitert sich  $\pi$  zu einer Interpretation  $\pi : \text{FO} \rightarrow S$  wie folgt [DG19]

- Für eine Formel  $\varphi$  schreiben wir  $\text{nnf}(\varphi)$  für die Negations-Normalform von  $\varphi$ . Dann gilt  $\pi \llbracket \neg\varphi \rrbracket = \pi \llbracket \text{nnf}(\neg\varphi) \rrbracket$

<sup>1</sup>Für eine Relation  $R$  ist  $\text{st}(R) \in \mathbb{N}$  die Stelligkeit von  $R$ .

- $\pi[\varphi_1 \wedge \varphi_2] = \pi[\varphi_1] \cdot \pi[\varphi_2]$
- $\pi[\varphi_1 \vee \varphi_2] = \pi[\varphi_1] + \pi[\varphi_2]$
- $\pi[\exists x \varphi(x)] = \pi[\sum_{a \in A} \varphi(a)]$
- $\pi[\forall x \varphi(x)] = \pi[\prod_{a \in A} \varphi(a)]$

Es lässt sich erkennen, dass  $\vee$  und  $\exists$  durch  $+$ , und  $\wedge$  und  $\forall$  durch  $\cdot$  dargestellt werden. Informell stellt  $+$  also eine alternative Nutzung von Informationen dar, da bei  $\vee$  nur eine der beiden Optionen benötigt wird, während  $\cdot$  für gemeinsame Nutzung steht, da bei  $\wedge$  alle Optionen gelten müssen. Um diese Definition zu verdeutlichen soll ein Beispiel vorgeführt werden.

**Beispiel 6** (Beispiel für FO mit dem Viterbi-Halbring)

Sei  $A = \{1, 2, 3, 4\}$ ,  $\tau = \{E\}$ , wobei  $E$  ein 2-stelliges Relationssymbol ist und  $\pi : \text{Lit}_A(\tau) \rightarrow [0, 1]$  eine  $\mathbb{V}$ -Interpretation in den Viterbi-Halbring  $\mathbb{V} = ([0, 1], \max, \cdot, 0, 1)$  gegeben durch Tabelle 4.

$E(x, y)$	$y = 1$	$y = 2$	$y = 3$	$y = 4$
$x = 1$	0	0.15	0.32	0
$x = 2$	0	0	0	0
$x = 3$	0	0	0	0.21
$x = 4$	0.99	0	0	0.06

$\neg E(x, y)$	$y = 1$	$y = 2$	$y = 3$	$y = 4$
$x = 1$	1	0	0	1
$x = 2$	1	1	1	1
$x = 3$	1	1	1	0
$x = 4$	0	1	1	0

Tabelle 4: Tabelle zur Definition der  $\mathbb{V}$ -Interpretation

Wenn man sich diese Definition anschaut fällt einem auf, dass diese einen Graphen definiert, auf dem jede Kante durch eine Wahrscheinlichkeit annotiert ist. Dies liegt daran, dass für jedes  $(x, y) \in A^2$  gilt, dass  $\pi[E(x, y)] = 0$  genau dann, wenn  $\pi[\neg E(x, y)] \neq 0$ , diese Eigenschaft nennt man auch *modelldefinierend* [DG19]. Der zugehörige Graph  $G$  ist in Abbildung 3 einsehbar.

Eine mögliche Interpretation des Viterbi-Halbrings ist es, diese als Vertrauenswerte zu verwenden. Ein Wert im Viterbi-Halbring gibt also Informationen darüber, wie sicher man über die Korrektheit eines Faktums ist. Hier am Beispiel heißt das zum Beispiel, dass die Kante von Knoten 1 zu 2 mit einer Wahrscheinlichkeit von  $\pi[E(1, 2)] = 0.15$  existiert und mit einer Wahrscheinlichkeit von  $\pi[\neg E(1, 2)] = 0$  nicht existiert.

Betrachten wir bspw. die Formel  $\varphi = \forall x \exists y (Exy)$  die aussagt, dass jeder Knoten eine ausgehende Kante besitzt. Wenn wir also wissen wollen, wie sicher man sich sein kann,

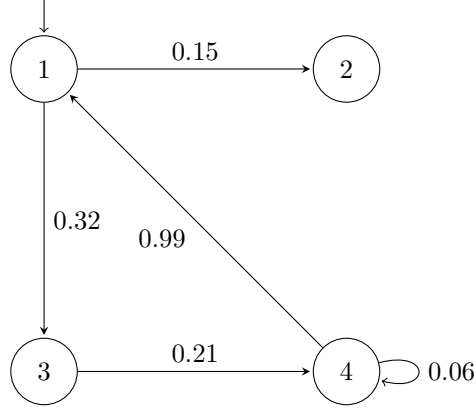


Abbildung 3: Graph  $G$  für die den durch  $\pi$  definierten Graphen

dass Graph  $G$  diese Eigenschaft besitzt, müssen wir bloß  $\pi[\varphi]$  auswerten. Es gilt

$$\begin{aligned}
 \pi[\forall x \exists y E(x, y)] &= \prod_{a \in A} \pi[\exists y E(a, y)] \\
 &= \prod_{a \in A} \sum_{b \in A} \pi[E(a, b)] \\
 &= \left( \prod_{a \in A \setminus \{2\}} \sum_{b \in A} \pi[E(a, b)] \right) \cdot \underbrace{\left( \sum_{b \in A} \pi[E(2, b)] \right)}_{=0} \\
 &= 0.
 \end{aligned}$$

Die Formel gilt also nicht für den durch  $\pi$  definierten Graphen.

Für die Formel  $\psi = \forall y \exists x E(x, y)$  die besagt, dass jeder Knoten eine eingehende Kante hat lässt sich eine ähnliche Rechnung durchführen und man erhält  $\pi[\psi] = 0.99 \cdot 0.15 \cdot 0.32 \cdot 0.21 \approx 1\%$

Somit können wir also ähnliche Aussagen treffen wie dies mit PCTL und  $t = \infty$  möglich ist. Das Problem ist jedoch, dass FO auf Graphen sehr wenig Aussagekraft besitzt, weshalb sich diese nicht zur Verifikation anbietet. Aus diesem Grund soll nun Halbringsemantik für CTL eingeführt werden.

## 4.2 Halbringsemantik für CTL

In diesem Kapitel wollen wir die im letzten Kapitel für FO entwickelte Semantik auf CTL ausweiten. Die so erhaltene Logik bezeichnen wir als c-CTL, was für *constraint-semiring-*

CTL steht [LLM05]. Ein wichtiger Unterschied zu FO ist aber, dass CTL eine modale Logik ist, wodurch es nicht ausreicht eine Formel in einen einzelnen Wert eines Halbrings auszuwerten. Aus diesem Grund definiert eine Formel eine Abbildung von den Kanten des Transitionssystems (siehe Definition 2.2) in einen Halbring. Formal definieren wir für einen Halbring  $\mathfrak{A} = (A, +, \cdot, 0, 1)$  und eine Kripkestruktur  $\mathcal{S} = (S, E, \mathcal{L})$  eine Abbildung  $v : E \rightarrow A$  und weiter  $V := A^E$  die Menge aller Bewertungen der Zustände. Nun erhalten wir einen weiteren Halbring  $\mathcal{V} = (V, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ , wobei  $\mathbf{0} : E \rightarrow A, e \mapsto 0$ ,  $\mathbf{1} : E \rightarrow A, e \mapsto 1$  gelten, und die beiden zweistelligen Funktionen wie folgt definiert sind:  $(v \oplus v')(e) := v(e) + v'(e)$  und  $(v \otimes v')(e) := v(e) \cdot v'(e)$ .

**Definition 4.3** (Syntax von c-CTL)

Sei  $\mathfrak{A} = (A, +, \cdot, 0, 1)$  ein wie in Kapitel 4.1 definierter Halbring und sei  $a \in A$  und  $v \in V$ . Dann lässt sich c-CTL mithilfe folgender Grammatiken definieren:

$$\begin{aligned}\varphi &::= a \mid v \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg \varphi \mid \kappa \psi \mid \kappa \mathbf{X} \varphi \\ \kappa &::= \sum \mid \prod \\ \psi &::= \mathbf{F} \varphi \mid \mathbf{G} \varphi \mid [\varphi \mathbf{U} \varphi] \mid [\varphi \mathbf{W} \varphi]\end{aligned}$$

Dann ist die Menge der c-CTL-Formeln die Ausdrücke, die von  $\varphi$  erzeugt werden. [LLM05]

Die intuitiven Bedeutungen von  $\wedge$ ,  $\vee$  und  $\neg$  sind wie bekannt.  $a$  stellt eine konstante Bewertung aller Kanten dar, während  $v$ , analog zu booleschen atomaren Aussagen, jeder Kante einen Halbring-Wert zuweist. Die durch  $\kappa$  erzeugten Ausdrücke sind analog zu Quantoren und die durch  $\psi$  erzeugten Ausdrücke stellen die aus CTL bekannten Begriffe *finally* ( $\mathbf{F}$ ), *globally* ( $\mathbf{G}$ ), *until* ( $\mathbf{U}$ ) und *weak until* ( $\mathbf{W}$ ) dar.

Bevor wir diese Bedeutungen formal definieren, soll noch über Negation gesprochen werden. Im Booleschen-Halbring ist es offensichtlich möglich einen Wert zu invertieren bzw. zu negieren, da es lediglich zwei mögliche Werte gibt und demnach immer *der andere* Wert eindeutig definiert ist. Im Allgemeinen ist dies aber nicht so trivial. Zum Lösen dieses Problems existieren zwei verschiedene Ansätze. Zum Einen ist es möglich einen Negationsoperator  $- : A \rightarrow A$  zu definieren. Zum Anderen lässt sich  $v$  ausweiten auf eine Bewertung  $v^* : \{E(x, y), \neg E(x, y) : x, y \in S\} \rightarrow A$  so, dass, wie in Kapitel 4.1, nicht nur der Existenz einer Kante ein Wert zugeordnet wird, sondern auch dem Fehlen der Kante. Im Weiteren soll der Letztere Ansatz verwendet werden. Demnach definieren wir die Menge  $V^* := A^{\{E(x, y), \neg E(x, y) : x, y \in S\}}$  aller erweiterten Bewertungen.

**Definition 4.4** (Semantik von c-CTL)

Für einen Halbring  $\mathfrak{A} = (A, +, \cdot, 0, 1)$  und eine Kripke-Struktur  $\mathfrak{S} = (S, E, \mathcal{L})$  wird eine c-CTL-Formel  $\varphi$  zu einer Abbildung  $v \in V$  ausgewertet. Diese Auswertung lässt sich induktiv mithilfe folgender Regeln definieren [LLM05]:

- $\llbracket a \rrbracket(s) = a$  und  $\llbracket v \rrbracket(s) = v(s)$  für alle  $s \in S$
- $\llbracket \varphi_1 \wedge \varphi_2 \rrbracket = \llbracket \varphi_1 \rrbracket \otimes \llbracket \varphi_2 \rrbracket$ ,  $\llbracket \varphi_1 \vee \varphi_2 \rrbracket = \llbracket \varphi_1 \rrbracket \oplus \llbracket \varphi_2 \rrbracket$ ,  $\llbracket \neg \varphi \rrbracket = -\llbracket \varphi \rrbracket$
- $\llbracket \mathbf{F} \varphi \rrbracket(s) = \text{true}$  falls es ein  $s' \in S$  mit  $(s, s') \in E$  und  $\llbracket \varphi \rrbracket(s') = \text{true}$  gibt, sonst  $\text{false}$

### 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.
- [Bar77] Jon Barwise. An introduction to first-order logic. In *Studies in Logic and the Foundations*

- of Mathematics*, Jgg. 90, Seiten 5–46. Elsevier, 1977.
- [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.
  - [DG19] Katrin M Dannert und Erich Grädel. Provenance Analysis: A Perspective for Description Logics? *Description Logic, Theory Combination, and All That: Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday*, Seiten 266–285, 2019.
  - [EFTF94] Heinz-Dieter Ebbinghaus, Jörg Flum, Wolfgang Thomas und Ann S Ferebee. *Mathematical logic*, Jgg. 1910. Springer, 1994.
  - [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  $\mu$ -calculus and CTL based on constraint semirings. *Electronic Notes in Theoretical Computer Science*, 112:37–59, 2005.
  - [LS82] Daniel Lehmann und Saharon Shelah. Reasoning with time and chance. *Information and Control*, 53(3):165–198, 1982.