ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2014 Prof. R. Wattenhofer / K.-T. Förster, T. Langner, J. Seidel

Diskrete Ereignissysteme

Prüfung

Montag, 19. Januar 2015, 9:00 – 12:00 Uhr

Nicht öffnen oder umdrehen bevor die Prüfung beginnt!

Die Prüfung dauert 180 Minuten und es gibt insgesamt 180 Punkte. Die Anzahl Punkte pro Teilaufgabe steht jeweils in Klammern bei der Aufgabe. Sie dürfen die Prüfung auf Englisch oder Deutsch beantworten. **Begründen Sie** alle Ihre Antworten und beschriften Sie Skizzen und Zeichnungen verständlich.

Schreiben Sie zu Beginn Ihren Namen und Ihre Legi-Nummer in das folgende dafür vorgesehene Feld und beschriften Sie jedes beschriebene Blatt mit Ihrem Namen und Ihrer Legi-Nummer.

Name	**Legi-Nr.**

Punkte

Aufgabe	**Erreichte Punktzahl**	**Maximale Punktzahl**
1		20
2		36
3		22
4		47
5		27
6		28
Summe		**180**

1 Multiple Choice (20 Punkte)

Beurteilen Sie, ob die folgenden Aussagen richtig oder falsch sind, und kreuzen Sie die entsprechenden Felder an. Eine richtig beurteilte Aussage gibt 1 Punkt, eine nicht beurteilte Aussage 0 Punkte, eine **nicht richtig beurteilte Aussage -1 Punkt**. Die gesamte Aufgabe wird mit minimal 0 Punkten bewertet. **Bei dieser Aufgabe müssen Sie ihre Aussagen nicht begründen!**

Aussage	**wahr**	**falsch**
Für eine Markovkette und zwei Zustände i und j gilt: Wenn $f_{ij} < 1$, dann $h_{ij} = \infty$.	☐	☐
Wenn ein Problem in der Komplexitätsklasse NP enthalten ist, dann weiss man, dass es nicht in polynomieller Zeit gelöst werden kann.	☐	☐
Es gibt einen PDA, der die Sprache $L = \{0^n 1^n \mid n \geq 10\}$ erkennen kann.	☐	☐
Ein strikt 2-kompetitiver Online-Algorithmus A kann für gewisse Probleminstanzen schlechter sein als ein 3-kompetitiver Online-Algorithmus B.	☐	☐
Wenn jede Transition in einem Petri-Netz L_1-live ist, dann gibt es auch eine Transition die L_4-live ist.	☐	☐
Wenn die Anzahl der Plätze in einem Petri-Netz eins ist, dann ist es safe.	☐	☐
Jede Petri-Netz-Sprache ist regulär.	☐	☐
Für die Modellierung der Ankunft von Anfragen von vielen unabhängigen und ähnlichen Benutzern bieten sich Poisson-Prozesse an.	☐	☐
Jeder endliche Automat kann auch als State Chart dargestellt werden.	☐	☐
Variablen in State Charts sind nur lokal für Zustände oder Superzustände definiert, nicht global.	☐	☐
Wenn man deterministische 3- und 5-kompetitive Online-Algorithmen für ein Problem hat, dann kann man daraus auch einen randomisierten 4-kompetitiven Online-Algorithmus erstellen.	☐	☐
Für keinen Algorithmus A kann entschieden werden, ob A bei jeder endlichen Eingabe hält.	☐	☐
Die Laufzeit eines strikt 2-kompetitiven Algorithmus darf höchstens doppelt so gross wie die des optimalen Algorithmus sein.	☐	☐
In der Warteschlangentheorie wird die Bearbeitungsdauer eines Jobs als feste Zeitspanne modelliert.	☐	☐
Man kann für jede kontextfreie Grammatik G_{CF} eine äquivalente kontextsensitive Grammatik G_{CS} konstruieren, die genau dieselbe Sprache akzeptiert.	☐	☐
Bei State Charts können UND-Superzustände durch die Negation von ODER-Superzuständen erstellt werden.	☐	☐
State Charts sind u.a. durch Variablenzuweisungen nichtdeterministisch.	☐	☐
Es gibt kein deterministisches Petri-Netz mit zwei Transitionen.	☐	☐
Wenn eine kontextfreie Sprache regulär ist, dann ist die Pumpingzahl 0.	☐	☐
Wenn in einem Petri-Netz eine Transition L_4-live ist, dann sind alle Wörter in einer zugehörigen Petri-Netz Sprache unendlich lang.	☐	☐

Musterlösung

Aussage	**wahr**	**falsch**
Für eine Markovkette und zwei Zustände i und j gilt: Wenn $f_{ij} < 1$, dann $h_{ij} = \infty$. *(Grund:)*	✓	
Wenn ein Problem in der Komplexitätsklasse NP enthalten ist, dann weiss man, dass es nicht in polynomieller Zeit gelöst werden kann. *(Grund: . . .)*		✓
Es gibt einen PDA, der die Sprache $L = \{0^n 1^n \mid n \geq 10\}$ erkennen kann. *(Grund:)*	✓	
Ein strikt 2-kompetitiver Online-Algorithmus A kann für gewisse Probleminstanzen schlechter sein als ein 3-kompetitiver Online-Algorithmus B. *(Grund:)*	✓	
Wenn jede Transition in einem Petri-Netz L_1-live ist, dann gibt es auch eine Transition die L_4-live ist. *(Grund:)*		✓
Wenn die Anzahl der Plätze in einem Petri-Netz eins ist, dann ist es safe. *(Grund:)*		✓
Jede Petri-Netz-Sprache ist regulär. *(Grund:)*		✓
Für die Modellierung der Ankunft von Anfragen von vielen unabhängigen und ähnlichen Benutzern bieten sich Poisson-Prozesse an. *(Grund:)*	✓	
Jeder endliche Automat kann auch als State Chart dargestellt werden. *(Grund:)*	✓	
Variablen in State Charts sind nur lokal für Zustände oder Superzustände definiert, nicht global. *(Grund: global..wer in VL war, der weiss das...)*		✓
Wenn man deterministische 3- und 5-kompetitive Online-Algorithmen für ein Problem hat, dann kann man daraus auch einen randomisierten 4-kompetitiven Online-Algorithmus erstellen. *(Grund:)*	✓	
Für keinen Algorithmus A kann entschieden werden, ob A bei jeder endlichen Eingabe hält. *(Grund:)*		✓
Die Laufzeit eines strikt 2-kompetitiven Algorithmus darf höchstens doppelt so gross wie die des optimalen Algorithmus sein. *(Grund: Trololololo)*		✓
In der Warteschlangentheorie wird die Bearbeitungsdauer eines Jobs als feste Zeitspanne modelliert. *(Grund:)*		✓
Man kann für jede kontextfreie Grammatik G_{CF} eine äquivalente kontextsensitive Grammatik G_{CS} konstruieren, die genau dieselbe Sprache akzeptiert. *(Grund:)*	✓	
Bei State Charts können UND-Superzustände durch die Negation von ODER-Superzuständen erstellt werden. *(Grund: Grober Unfug)*		✓
State Charts sind u.a. durch Variablenzuweisungen nichtdeterministisch. *(Grund: Ja, write-write race)*	✓	
Es gibt kein deterministisches Petri-Netz mit zwei Transitionen. *(Grund: doch.)*		✓
Wenn eine kontextfreie Sprache regulär ist, dann ist die Pumpingzahl 0. *(Grund: wtf)*		✓
Wenn in einem Petri-Netz eine Transition L_4-live ist, dann sind alle Wörter in einer zugehörigen Petri-Netz Sprache unendlich lang. *(Grund: unendlich mal epsilon...)*		✓

2 Reguläre Sprachen (36 Punkte)

a) Geben Sie jeweils einen endlichen Automaten an, der die Sprache L erkennt, oder beweisen Sie, dass die Sprache nicht regulär ist.

(i) [5] $L = \{w \in \{a,b\}^* \mid w \text{ enthält die Zeichenketten } aa \textbf{ und } bb\}$

(ii) [5] $L = \{w \in \{a,b\}^* \mid w = a^n b^{2n},\ n \geq 0\}$

(iii) [5] $L = \{w \in \{a,b\}^* \mid abw = wba\}$

(iv) [5] $L = \{w \in \{a,b\}^* \mid \text{es existiert } u \in \{a,b\}^*, \text{ so dass } www = uu\}$

b) [10] Konstruieren Sie einen möglichst kleinen NFA, der die folgende Sprache akzeptiert:

$$L = \{w \in \{a\}^* \mid |w| \neq 30 \cdot i \text{ für irgendein } i \in \mathbb{N}_{>0}\}$$

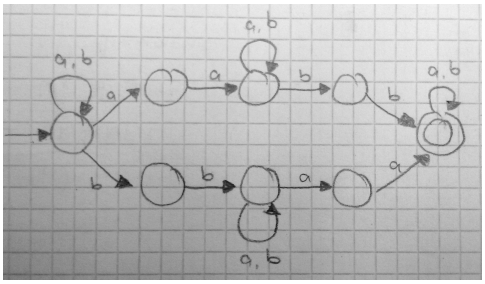
Automaten mit weniger Zuständen ergeben mehr Punkte.

Hinweis: Es existieren Automaten mit weniger als 30 Zuständen.

c) [6] Beweisen oder widerlegen Sie die folgenden Aussage: Eine Sprache kann nicht regulär sein, wenn sie die *Konkatenation* zweier nicht-regulären Sprachen ist.

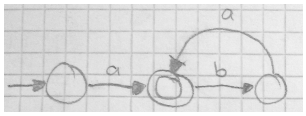
Musterlösung

a) (i) Die Sprache ist regulär, da sie vom folgenden Automaten akzeptiert wird.



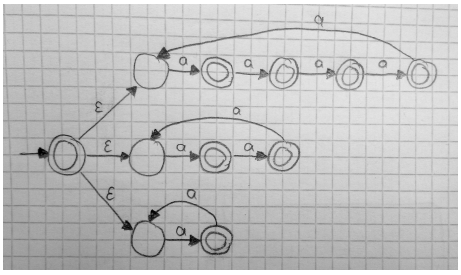
(ii) Die Sprache ist nicht regulär, da sie das Pumping-Lemma nicht erfüllt.
Sei p die Pumpingzahl von L. Wir betrachten das Wort $w = a^p b^{2p} \in L$. Für alle Zerlegungen $w = xyz$ mit $|xy| \leq p$ und $|y| \geq 1$ ist das Wort $w' = xy^0z = a^{p-|y|}b^{2p}$ nicht in L. Damit kann L nicht regulär sein.

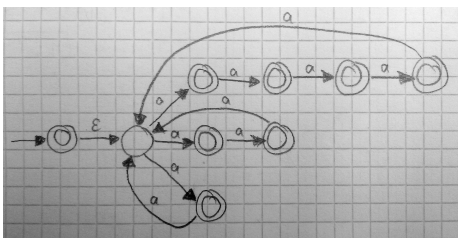
(iii) Durch Inspektion erkennt man, dass die Sprache durch den regulären Ausdruck $a(ba)^*$ beschrieben werden kann. Der folgende Automat akzeptiert sie.



(iv) Die Sprache ist nicht regulär, da sie das Pumping-Lemma nicht erfüllt.
Sei p die Pumpingzahl von L. Wir betrachten das Wort $w = 10^p110^p1$ was zu L gehört, da $www = uu$ mit $u = 10^p110^p110^p1$. Für alle Zerlegungen $w = xyz$ mit $|xy| \leq p$ und $|y| \geq 1$ ist das Wort $w' = xy^0z$ nicht in L, da man w' entweder eine ungerade Anzahl von Einsen enthält oder einen Block mit weniger als p Nullen. Entsprechend lässt es sich nicht mehr in identische Hälften zerlegen.

b) Der folgende Automat akzeptiert die Sprache, da nur genau dann kein Ausführungspfad in einem akzeptierenden Zustand weilt, wenn das eingelesene Wort nicht leer und seine Länge durch 5, 3 und 2 teilbar ist. Ist dies der Fall, so ist das Wort durch 30 teilbar. Für Automaten welche beliebige Längen von Wörtern akzeptieren, aber die richtige Idee zu Grunde hatten, wie z.B. der zweite Automat wurden 2 Punkte abgezogen.





c) Die Konkatenation zweier nicht-regulärer Sprachen kann regulär sein.

Wir betrachten die beiden Sprachen $L_1 = \{w \in \{a,b\}^* \mid \#_a(w) \geq \#_b(w)\}$ und $L_2 = \{w \in \{a,b\}^* \mid \#_a(w) \leq \#_b(w)\}$ wobei $\#_x(w)$ die Anzahl der Vorkommen des Symbols x im Wort w bezeichnet. L_1 und L_2 sind nicht-regulär, da sie geschnitten mit der regulären Sprache a^*b^* die nicht-regulären Sprachen $\{a^{n+i}b^n \mid i \geq 0\}$ bzw. $\{a^nb^{n+i} \mid i \geq 0\}$ ergeben.

Konkateniert man L_1 mit L_2, so erhält man die reguläre Sprache $L_3 = \{a,b\}^*$, da jedes Wort aus L_3 entweder in L_1 oder L_2 sein muss und man es damit durch Konkatenation mit ε konstruieren kann.

3 Kontextfreie Sprachen (22 Punkte)

Gegeben sind die folgenden Produktionsregeln einer mehrdeutigen (ambiguous) kontextfreien Grammatik G über dem Alphabet $\Sigma = \{a, b\}$.

$$S \to Ab \mid aaB$$
$$A \to a \mid Aa$$
$$B \to b$$

a) [4] Finden Sie das Wort in der von G erzeugten Sprache $L(G)$, das durch zwei verschiedene Linksableitungen (left-most derivations) erzeugt werden kann, und geben Sie die beiden Ableitungen an.

b) [3] Geben Sie eine äquivalente eindeutige (unambiguous) Grammatik für $L(G)$ an.

c) [3] Geben Sie die Produktionsregeln einer kontextfreien Grammatik an, die die Sprache $\{a^n b^m c^m d^{2n} \mid n \geq 0, m > 0\}$ erzeugt.

d) [6] Geben Sie die Produktionsregeln einer kontextfreien Grammatik an, die die Sprache

$$L = \{w\#x \mid w^{\text{rev}} \text{ ist in } x \text{ enthalten und } w, x \in \{a, b\}^+\}$$

erzeugt, wobei w^{rev} der Zeichenkette w mit umgekehrter Zeichenreihenfolge entspricht.

Beispiel: Es gilt $ab\#ababb \in L$, da $w = ab$ und die Zeichenkette $w^{\text{rev}} = ba$ in $x = ababb$ enthalten ist. Weiterhin gilt $ba\#baa \notin L$, da $w^{\text{rev}} = ab$ nicht in $x = baa$ enthalten ist.

e) [6] Geben Sie einen PDA an, der die Sprache aus Teil **d)** akzeptiert.

Musterlösung

a) Durch die zwei folgenden unterschiedlichen Linksableitungen kann das Wort aab erzeugt werden:

$$S \Rightarrow Ab \Rightarrow Aab \Rightarrow aab$$
$$S \Rightarrow aaB \Rightarrow aab$$

b) Die Regel $S \to aaB$ fügt eine Ambiguität in $L(G)$ hinzu. Das Weglassen dieser Regel macht $L(G)$ zu einer äquivalenten und eindeutigen Grammatik mit folgenden Produktionsregeln:

$$S \to Ab$$
$$A \to a \mid Aa$$

c)

$$S \to M \mid aMdd$$
$$M \to bc \mid bMc$$

Die Regel $S \to M$ erlaubt, dass $n = 0$ sein kann und damit $n \geq 0$ erfüllt ist. Die Regel $S \to aMdd$ stellt sicher, dass die die Anzahl der Symbole a und d genau n und $2n$ betragen. $M \to bc$ führt dazu, dass $m > 0$ gilt und dass die Symbole b und c gleich oft auftreten.

d)

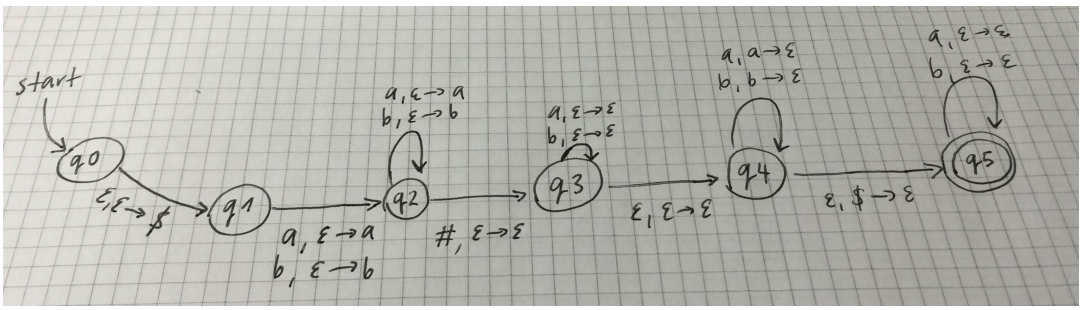
$$S \to TA$$
$$T \to aTa \mid bTb \mid M$$
$$M \to a\#Aa \mid b\#Ab$$
$$A \to \epsilon \mid a \mid b \mid aA \mid bA$$

Beim einmaligen oder mehrmaligen Ausführen der Regeln $T \to aTa$ und $T \to bTb$ entstehen Palindrome. Die Sequenzen links und rechts des Symbols T sind gegenseitig spiegelverkehrte Abfolgen. Die Regel $T \to M$ generiert eine einmalige Mitte der Symbolkette, in der mit den Regeln $M \to a\#Aa$ und $M \to b\#Ab$ das Abtrennungssymbol # gesetzt wird. Das Setzen dieses Abtrennungssymbols definiert die Länge und Beschaffenheit der Symbolketten w und w^{rev}. Damit ist ausgeschlossen, dass die Symbolketten w und w^{rev} leer sind, da immer jeweils mindestens ein a oder b vor und nach dem # erzwungen wird.

$$S \Rightarrow TA \Rightarrow .. \Rightarrow a..bTb..aA \Rightarrow a..bMb..aA \Rightarrow a..ba\#Aab..aA \qquad \to w\#Aw^{\text{rev}}A \quad (1)$$

Durch das Symbol A kann eine beliebige Anzahl von a, b entstehen. Aus der Ableitung von (1) ist ersichtlich, dass die Sequenz x, welche die gesamte Symbolkette nach dem # Symbol umfasst, $Aw^{\text{rev}}A$ beträgt. Somit ist sichergestellt, dass w^{rev} ganz und unverändert in x zu finden ist und x beidseitig von w^{rev} beliebig wachsen kann durch die Regeln $S \to TA$ und $M \to a\#Aa \mid b\#Ab$.

e) Der PDA beginnt bei $q0$ indem er das Startzeichen \$ auf den Stack legt.
Bei den beiden Folgezuständen $q1$ und $q2$ wird die Inputsequenz bestehend aus den Symbolen a und b auf den Stack gelegt. Wir teilen diesen Vorgang in zwei Zustände auf weil die Symbolkette vor dem # Symbol nicht leer sein darf und somit beim Zustand q1 noch nicht direkt ein # eingelesen werden darf. Per definition entspricht die Sequenz vor dem Symbol # der Zeichenkette w.
Falls das Symbol # gelesen wird bewegt sich der PDA zum Zustand $q3$. Bei $q3$ bleibt der nondeterministische Automat beliebig lange und kann Symbole a und b einlesen ohne den Stack zu verändern.
Die Sequenz w ist auf dem Stack in verkehrter Reihenfolge gespeichert. Nach erreichen von Zustand $q4$ wird sequentiell die Symbolkette w vom Stack entfernt aber dies nur wenn exakt eine Inputsequenz von w^{rev} anliegt. Der Stack ist in diesem Fall wieder leer bis auf das \$ Zeichen, welches gelöscht wird auf dem Weg zum akzeptierenden Endzustand $q5$.
Die Grammatik erlaubt auch nach der Symbolkette w^{rev} eine beliebige Sequenz von a und b Symbolen. Der Endzustand $q5$ zeigt dies indem er beliebig viele Eingangssymbole a und b akzeptiert ohne den Stack zu verändern.



4 Mobile Roboter (47 Punkte)

In dieser Aufgabe befassen wir uns mit einem mobilen Roboter, der sich in diskreten Schritten auf einer Linie bewegen kann. Die einzelnen Zellen der Linie entsprechenden den ganzen Zahlen $\mathbb{Z}$. Der Roboter beginnt die Ausführung im Ursprung (Zelle 0) und kann sich dann in jedem Schritt entweder eine Zelle nach links (-1) oder nach rechts $(+1)$ bewegen oder in der Zelle stehen bleiben. Das Ziel des Roboters ist, eine Bombe zu finden. Dies passiert genau dann, wenn er sich zum ersten Mal in der selben Zelle wie die Bombe befindet.

4.1 Dumme Roboter

In dieser Teilaufgabe wird der Roboter von einem *deterministischen endlichen Automaten* (DFA) gesteuert. Um zu entscheiden, was der Roboter in einem Schritt tut, spezifiziert die Übergangsfunktion des DFAs zusätzlich zum nächsten Zustand auch eine der Bewegungsrichtungen L (links), R (rechts) oder S (stehen bleiben). Formal wird der Automat beschrieben durch das Tupel $\langle Q, \delta \rangle$ wobei Q die endliche Zustandsmenge und $\delta : Q \to Q \times \{L, R, S\}$ die Übergangsfunktion ist. Befindet sich der Roboter also beispielsweise in Schritt i in Zelle z im Zustand q und gilt $\delta(q) = (q', L)$, dann befindet er sich in Schritt $i + 1$ im Zustand q' in Zelle $z - 1$, da er sich in Schritt i um eine Zelle nach links bewegt hat.

a) [5] Der Roboterhersteller *Skynet* behauptet, einen funktionsfähigen DFA konstruieren zu können, wenn sich die Bombe höchstens im Abstand 100 vom Ursprung befindet.

Geben Sie an, wie ein solcher DFA konstruiert werden kann, oder beweisen Sie, dass dies nicht möglich ist.

b) [10] Skynet behauptet weiterhin, über einen allmächtigen DFA zu verfügen, der es dem Roboter ermöglicht, die Bombe auch bei unbekanntem (beliebigem) Abstand zu finden.

Beweisen Sie, dass so ein DFA nicht existieren kann.

c) [5] Skynet veröffentlicht eine kleinlaute Pressemitteilung, in welcher zu lesen ist, dass der DFA aus Aufgabe **b)** eigentlich ein *nichtdeterministischer endlicher Automat* (NFA) sein sollte.

Geben Sie einen NFA an, der jede beliebige Zelle (nichtdeterministisch) findet und begründen Sie, wieso dies der Fall ist.

4.2 Schlaue Roboter

In dieser Teilaufgabe wird der Roboter durch eine Turingmaschine gesteuert. Die Bombe befindet sich im Abstand D vom Ursprung, wobei der Roboter den Parameter D aber nicht kennt.

a) [5] Der Roboter verwendet die folgende Strategie:

Gehe einen Schritt nach rechts, dann zwei nach links, dann drei rechts, vier links, fünf rechts und so weiter.

Nach wie vielen Schritten findet der Roboter die Bombe, wenn sie sich links vom Ursprung in Abstand D befindet?

b) [12] Geben Sie eine Strategie mit besserer Kompetitivität als in **a)** an, und bestimmen

Sie die Kompetitivität der Strategie. Je besser die Kompetitivität ist, die Sie nachweisen, desto mehr Punkte erhalten Sie.

c) [10] Nehmen Sie nun an, dass der Roboter im Ursprung startet und von dort m Strahlen jeweils indiziert durch die natürlichen Zahlen $\mathbb{N}$ ausgehen (siehe Abbildung 1). Die Bombe befindet sich in einer Zelle auf einem der m Strahlen im (unbekannten) Abstand D vom Ursprung. Der Roboter darf sich innerhalb eines Strahls bewegen und nur im Ursprung in einen anderen Strahl wechseln. Geben Sie eine Strategie mit möglichst guter Kompetitivität an, und bestimmen Sie die Kompetitivität der Strategie in Abhängigkeit von m. Je besser die Kompetitivität ist, die Sie nachweisen, desto mehr Punkte erhalten Sie.

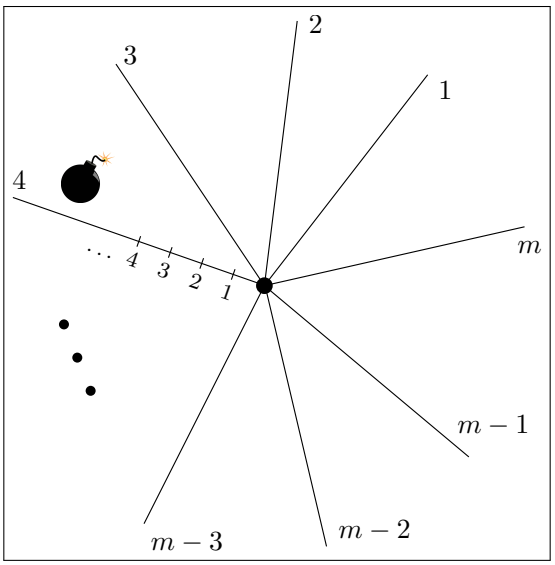
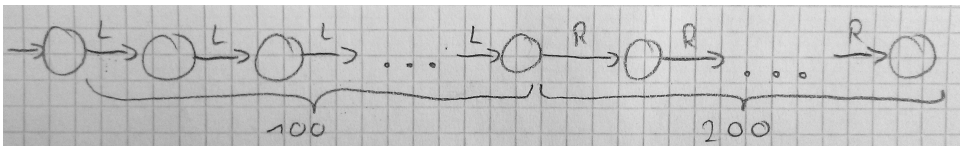


Abbildung 1: Der Roboter startet im Schnittpunkt der m Strahlen und kann nur dort die Strahlen wechseln. Die Bombe befindet sich auf einem der Strahlen.

Musterlösung

4.1 Dumme Roboter

a) Ein solcher DFA lässt sich mit 301 sequenziellen Zuständen konstruieren. Der Automat beginnt im Zustand, der dem Ursprung entspricht. Dann bewegt er sich beim Wechseln in den jeweils nächsten Zustand zunächst 100 mal nach links und danach 200 mal nach rechts. So erkundet er alle Zellen im Abstand ≤ 100 und findet daher die Bombe.



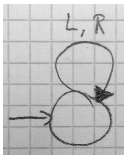
Ein kleinerer Automat mit nur 101 Zuständen geht zunächst 100 Zellen nach links und ab dann nur noch (immer wieder im selben Zustand) nach rechts.

b) Der DFA A von Skynet verfügt über eine endliche Anzahl k von Zuständen. Nach $k+1$ Schritten muss daher mindestens ein Zustand zweimal besucht worden sein. Da der DFA nicht erkennen kann, wie oft er sich im selben Zustand befindet, muss er ab dann in eine Endlosschleife geraten.

Sei $q \in Q$ der Zustand, der als erstes zweimal besucht wird und seien weiterhin c und c' die Zellen, in welchen sich der Roboter befindet, wenn er sich zum ersten bzw. zweiten Mal in Zustand q befindet. Wir unterscheiden drei Fälle:

- $c > c'$, der Roboter hat sich also zwischen den zwei Aufenthalten in Zustand q in der Summe nach links bewegt. Da er sich in einer Endlosschleife befindet, wird er das für alle Ewigkeit tun. Wenn man die Bombe in Zelle $k+1$ platziert, findet er sie also nicht.
- $c < c'$. Analog zu $c > c'$.
- $c = c'$, der Roboter hat sich also zwischen den zwei Aufenthalten in Zustand q in der Summe nicht bewegt. Da A nur k Zustände hat, kann er dazwischen nur maximal k verschiedene Zellen besucht haben. Wenn man die Bombe in Zelle $k+1$ platziert, findet er sie also nicht.

c) Der folgende NFA erfüllt die Aufgabe.



Grund 1: Jeder Zelle entspricht mindestens ein Ausführungspfad, weshalb jede Zelle nicht-deterministisch erkundet wird.

Grund 2: Die Bewegung des Roboters entspricht einem Random Walk, welcher jede Zelle mit Wahrscheinlichkeit 1 besucht.

4.2 Schlaue Roboter

a) Die Bombe ist links, d.h. der Roboter braucht bei Abstand D genau $1+2+3+4+...+(2D-1)+2D = D(2D+1) = 2D^2+D$ Schritte (Summenformel nach Gauss ueber i von $i=0$ bis $i=2D$).

b) Optimale Strategie: Gehe 2^0 nach rechts, dann zum Ursprung. Dann 2^1 nach links, dann zum Ursprung. Dann 2^2 nach rechts, dann zum Ursprung. Usw. Fuer $D = 2+\epsilon$ nach links ergibt sich etwa $1+1+2+2+4+4+2+\epsilon = 2*1+2*2+2*4+2+\epsilon = 2*(1+2+4)+2+\epsilon = 2(2^0+2^1+2^2)+2+\epsilon = 2(2^3-1)+2+\epsilon = (2^4-2)+2+\epsilon = 16+\epsilon$ Schritte vs $2+\epsilon$ Schritte im Optimalfall. Allgemein erhaelt asymptotisch bei einem Abstand von $2^{k-1} < D \leq 2^{k+1}$ Kosten von $2^{k+2}+D$, was eine kompetitive Ratio von hoechstens $8+1 = 9$ ergibt (im schlechtesten Fall ist $D = 2^{k-1}+\epsilon$ auf der "falschen" Seite).

c) Moegliche Strategie: Besuche alle Strahlen mit jeweils einem Schritt. Dann fuehre dieses analog fort, verdopple aber jeweils die Suchtiefe. Angenommen, die Bombe ist in Abstand 1 auf dem letzten Strahl. Dann waere die Kompetitivitaet $(2m-1)/1$. Wenn die Bombe $D = 2$ haette, dann waere es $(2m+4m-1)/2 = 3m-1/2$. Aehnlich wie vorher, nehme an die Bombe weare in Abstand $D = 2^k+\epsilon$. Dann haette man (vor der richtigen Schrittverdoppelungstiefe von k+1) pro Strahl Kosten von $2^{k+2}-2$, und im letzten Teil dann pro falschem Strahl Kosten von $2*2^{k+1} = 2^{k+2}$. D.h. es ergeben sich Kosten von $K_A = m*(2^{k+2}-2)+(m-1)*2^{k+2}+2^k+\epsilon$, was grob nach oben abgeschaetzt eine Kompetitivitaet von $8m$ ergibt, da $K_A/2^k \leq 8m$. Anmerkung: Eine optimale Strategie wuerde noch eine Kompetitivitaet von ca. $1+2em$ erreichen, dieses geht hier aber zu weit - bei $8m$ gibt es die volle Punktzahl.

5 Matchmaking für Videospiele (27 Punkte)

Sie verwalten einen Matchmaking-Server für ein 2-Spieler-Videospiel. Spieler melden sich bei Ihrem Server an, um einen Gegner zu erhalten. Wenn kein anderer menschlicher Spieler bereitsteht, muss ein neu ankommender Spieler warten. Da Spieler nicht gerne warten und manchmal nicht ausreichend Gegenspieler zur Verfügung stehen, haben Sie sich die Option geschaffen, einem Spieler einen Computergegner zuzuweisen. Gegen einen Computer zu spielen reduziert allerdings den Spielspass.

Formal: Immer, wenn ein Spieler sich anmeldet, müssen Sie *online* entscheiden, wie lange Sie den Spieler auf einen Spielpartner warten lassen, bevor Sie ihm einen Computergegner zuweisen. Das Ziel ist es nun, die Kostenfunktion zu minimieren, die sich aus der Summe der folgenden Kosten bildet:

- Einen Spieler Zeit t warten zu lassen, kostet t.
- Einem Spieler einen Computergegner zuzuweisen, hat Kosten von 1 (zusätzlich zu der Wartezeit des Spielers).

Der optimale Algorithmus arbeitet offline, d.h. er kennt die Ankunftszeiten aller Spieler im Voraus.

a) `[5]` Sie beginnen mit einer sehr einfachen Strategie: Sie weisen nie einen Computergegner zu. Was ist die Kompetitivität dieser Strategie?

b) `[4]` Sie benutzen nun eine bekannte Strategie aus dem Wintersport: Jeder Spieler erhält eine maximale Wartezeit von 1, dann wird ihm ein Computergegner zugewiesen. Damit verursacht jeder Spieler maximal Kosten von 2. Was ist die Kompetitivität dieser Strategie?

c) `[6]` Sie haben nun die Idee, die Wartezeit dynamisch zu variieren. Sie beginnen mit einer maximalen Wartedauer von $W := 2$. Wenn sich ein neuer Spieler anmeldet und noch kein zweiter Spieler wartet, dann lassen Sie ihn bis zu W Zeiteinheiten warten und – falls sich in dieser Zeit kein Gegner findet – weisen ihm dann einen Computergegner zu. Meldet sich rechtzeitig ein zweiter Spieler nach Zeit $w < W$, so paaren Sie die beiden und setzen $W := W - w$. Der nächste Spieler, der sich anmeldet, hat also eine kürzere maximale Wartezeit. Immer wenn ein Computergegner zugeteilt wird, setzen Sie die Wartedauer zurück auf $W := 2$.

Geben Sie ein möglichst schlechtes Beispiel für diese Strategie an, so dass die Kompetitivität möglichst gross wird.

d) `[12]` Geben Sie für dieses Problem eine möglichst gute untere Schranke für die Kompetitivität jedes randomisierten Online-Algorithmus an. Höhere untere Schranken ergeben mehr Punkte.

Hinweis: Von Neumann/Yao-Prinzip!

Musterlösung

a) Meldet sich nur 1 Spieler, sind die Kosten dieser Strategie beliebig hoch, während der optimale Algorithmus nur Kosten von 1 hat.
→ ∞-kompetitiv / nicht c-kompetitiv / beliebig schlecht.

b) Melden sich Spieler zu den Zeitpunkten $[0, 1-\epsilon, 1, 2-\epsilon, \dots]$, so sind die Kosten dieser Strategie $1-\epsilon$ für jede 2 Spieler, während es den optimalen Algorithmus nur einmal 1 und danach ϵ für jede 2 Spieler kostet.
→ ∞-kompetitiv / nicht c-kompetitiv / beliebig schlecht.

c) Melden sich Spieler zu den Zeitpunkten $[0, 2-\epsilon, 2]$, so hat diese Strategie Kosten von 3, während der optimale Algorithmus nur Kosten von $1+\epsilon$ hat.
→ Die Strategie ist höchstens 3-kompetitiv.

d) Dem Yao-Prinzip zufolge reicht es, wenn wir die Schranke für alle deterministischen Algorithmen zeigen.

Jeder deterministische Algorithmus muss sich für eine maximale Wartezeit W_1 für den ersten ankommenden Spieler entscheiden.

Falls $W_1 \geq 1$ so ist der Algorithmus höchstens 2-kompetitiv für die Ankunftszeiten $[0]$.

Falls $W_1 < 1$ so ist der Algorithmus höchstens 2-kompetitiv für die Ankunftszeiten $[0, W_1+\epsilon]$. ($cost_{ALG} \geq W_1 + 1 + 1$, $cost_{OPT} = W_1 + \epsilon$)

→ Kein deterministischer oder randomisierter Online-Algorithmus kann besser als 2-kompetitiv sein.

6 Kontinuierliche Markov-Toiletten (28 Punkte)

Sie werden von einem Unterhaltungsunternehmer gebeten, ihn bei der Dimensionierung der Räumlichkeiten seiner DISCO zu unterstützen. Insbesondere möchte ihr Auftraggeber besser verstehen, warum es bei den Toiletten ständig zu langen Schlangen kommt. Er bittet Sie daher um eine detailliertere Modellierung der Toiletten.

Sie entschliessen sich, die Herrentoilette als **M/M/m-System** zu modellieren. Ankunfts- und Bedienrate sind exponentialverteilt mit Parameter λ bzw. μ, und es sind m Toiletten vorhanden.

a) [3] Wie lang dauert es im Erwartungswert, bis der erste Gast die Toilette aufsucht? Wie viele Toiletten sind nötig, damit die Warteschlange nicht ins unermessliche wächst?

Lösung: $E[\exp(\lambda)] = 1/\lambda$. Stabil, wenn $\rho < 1 \iff m > \lambda/\mu$.

b) [3] Welche Zeitspanne ist länger, und warum?

(i) Die Zeit, die es dauert, bis erstmalig alle m Toiletten besetzt sind.

(ii) Die Zeit, die es bei bereits vollständig besetzten Toiletten dauert, bis m weitere Personen in der Schlange stehen müssen.

Lösung: (ii), denn die Schlange wird mit $m\mu$ abgearbeitet, die Toiletten nur mit $i\mu$.

c) [4] Der erste Gast geht auf die noch leere Toilette (er muss also nicht warten). Was ist die Wahrscheinlichkeit dafür, dass er dort weniger lang als erwartet benötigt?

Lösung: Bezeichne mit der Zufallsvariable X die Zeit, die der Gast in der Toilette verbringt. Weil $X \sim \exp(\mu)$ ist, ist die Verteilungsfunktion $F_X(t) = 1 - \exp(t)$. Damit erhält man:

$$\Pr[X \leq E[X]] = \Pr[X \leq 1/\mu] = F_X(1/\mu) = 1 - \exp(-\mu \cdot 1/\mu) = 1 - \exp(-1) \approx 0.632\,.$$

d) [4] Nun wollen wir uns mit der *Skalierungseigenschaft* der Exponentialverteilung befassen. Sei X eine exponentialverteilte Zufallsvariable mit Parameter $\lambda > 0$. Zeigen Sie für $a > 0$, dass die Zufallsvariable $Y = a \cdot X$ exponentialverteilt mit Parameter λ/a ist.

Lösung: Da X mit Parameter λ exponentialverteilt ist, ist $\Pr[X > t] = \exp(\lambda t)$. Man erhält:

$$\Pr[aX > t] = \Pr[X > t/a] = \exp(-\lambda(t/a)) = \exp((-\lambda/a)t)\,.$$

Dies bedeutet nichts anderes, als dass die Zuvallsvariable aX exponentialverteilt mit Parameter λ/a ist.

e) [6] Der Unternehmer möchte die Örtlichkeiten renovieren. Er postuliert, dass männliche DISCO-Besucher in der Toilette sowieso nur Urinale benötigen. Für den Umbau kann er entweder m günstige Urinale mit Wasserspülung oder $m/2$ doppelt so teure wasserlose Urinale kaufen. Da der Spülvorgang im wasserlosen Fall entfällt, ist die Bedienrate dort doppelt so gross, also 2μ. Da der Unternehmer Markov-Fan ist, interessieren ihn Argumente wie laufende Kosten für verbrauchtes Wasser, Umweltaspekte und Platzverbrauch nicht.

Wie unterscheidet sich die Stabilität des Systems in den beiden Varianten? Geben Sie (mit der Theorie von Markov-Ketten) je ein Argument sowohl für die teuren wie auch die günstigen Urinale.

Lösung: Die Verkehrsdichten für den günstigen bzw. teuren Fall sind

$$\rho_{\mathrm{g}} = \frac{\lambda}{m\mu} \quad \text{und} \quad \rho_{\mathrm{t}} = \frac{\lambda}{m/2 \cdot 2\mu}\,.$$

Es gilt also $\rho_g = \rho_t$, d.h. die Stabilität ist in beiden Fällen gleich.

Folgendes Argument spricht für die günstigere Variante: Nehmen wir an, es befinden sich $i > m$ Gäste im System, und betrachten wir den i-ten Gast. In der wasserlosen teureren Variante dauert es $(i-1)/(2\mu)$ Zeiteinheiten bis der i-te Gast zu einer Toilette kommt. In der günstigen Variante dauert es allerdings nur $(i-2)/(2\mu)$ Zeiteinheiten, bis Gast i zu einer Toilette kommt. Es ist also zur günstigen Lösung zu raten, da der i-te Gast im Erwartungswert schneller bedient wird (also anfangen kann, eine Toilette zu benutzen) als im teuren Fall.

Folgendes Argument spricht für die teurere Variante: Nehmen wir an, es befinden sich zwei Gäste im System. In der günstigen Variante benötigen beide Gäste erwartet Zeit $1/\mu$. In der wasserlosen teureren Variante braucht der erste Gast erwartet nur $1/(2\mu)$, der zweite Gast braucht erwartet $2 \cdot 1/(2\mu) = 1/\mu$ Zeit. Es ist also zur wasserlosen teuren Lösung zu raten, da der erste Gast schneller fertig wird, und die Bedienrate im Falle einer Schlange gleich bleibt.

f) [8] Man kann auch für kontinuierliche Markov-Ketten die *expected hitting time* definieren. Für diese wollen wir eine allgemeine Formel herleiten. Wie in der Vorlesung (Folie 4/79) bezeichnen wir für zwei Zustände $i \neq j$ mit $\nu_{i,j}$ die *Übergangsrate* von Zustand i nach j, und mit $\nu_{i,i}$ die Summe der $\nu_{i,j}$ mit $i \neq j$.

Für zwei Zustände i und j ist die *expected hitting time* $h_{i,j}$ die erwartete Zeit, die benötigt wird, um von i nach j zu kommen. Zeigen Sie, dass die $h_{i,j}$ das folgende lineare Gleichungssystem erfüllen:

$$\begin{aligned} h_{i,i} &= 0 \\ \sum_{k \in S} \nu_{j,k} h_{k,j} &= -1 \qquad \text{für } j \neq i\,. \end{aligned}$$

Lösung: Natürlich gilt $h_{i,i} = 0$. Die Zufallsvariable $T_{j,k}$ bezeichne die Zeit, die es dauert, um von Zustand j nach k zu gehen. Damit ist $h_{j,k} = E[T_{j,k}]$. Für $s \in S$ sei D_s die Verweildauer in Zustand s. Die Verteilung der D_s ist $D_s \sim \exp(\nu_{i,i})$.

$$\begin{aligned} h_{j,k} = E[T_{j,k}] &= E[D_j] + \sum_{k \in S \setminus \{j\}} E[T_{k,j} | X_0 = k] p_{j,k} \\ &= E[D_j] + \sum_{k \in S \setminus \{j\}} h_{k,j} p_{j,k} \\ &= 1/\nu_{j,j} + \sum_{k \in S \setminus \{j\}} h_{k,j} \nu_{j,k} / \nu_j \\ \iff -1 &= h_{j,k} \nu_{j,j} + \sum_{k \in S \setminus \{j\}} h_{k,j} \nu_{j,k} \\ &= \sum_{k \in S} h_{k,j} \nu_{j,k}\,. \end{aligned}$$

□