ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze

HS 2009

Dr. Kai Lampka
Jasmin Smula
Raphael Eidenbenz

Discrete Event Systems
Exercise 11

1 Computation Tree Logic (CTL) Model Checking

Sei folgende Kripke-Struktur $\mathcal{K}$ gegeben:

$$\mathcal{K} := \left\{ \begin{array}{l} \mathbb{S} := \{1,2,3,4\}; \\ \mathbb{S}_0 := \{1\}; \\ \mathbb{E} := \{(1,3),(3,2),(2,1),(2,4),(4,2)\}; \\ \mathcal{AP} := \{green, yellow, red, black\}; \\ \mathcal{L} := \{1 \mapsto red, 2 \mapsto yellow, 3 \mapsto green, 4 \mapsto black\}; \end{array} \right\}.$$

a) Bitte geben Sie den Graphen der Kripke-Struktur $\mathcal{K}$ an.

b) Entwickeln Sie nun den Computation-Tree für den Zustand $\mathbb{S}_0$ bis zur Tiefe 7.
Anmerkung: Der Wurzelknoten hat Tiefe 1.

Seien folgende CTL-Formeln gegeben:

$$\begin{array}{lll} \Omega_1 = \exists\Box green & \Omega_2 = \exists\Box\neg green & \Omega_3 = \exists(yellow \Rightarrow (\forall \bigcirc black)) \\ \Omega_4 = \forall\Box yellow & \Omega_5 = \exists \bigcirc (true \bigcup black) & \Omega_6 = \forall(black \bigcup black) \\ \Omega_7 = \exists(black \bigcup black) & \Omega_8 = \forall(\neg black \bigcup black) & \Omega_9 = \forall(\neg yellow \bigcup (\exists \bigcirc black)) \\ \Omega_{10} = \exists\Diamond black & \Omega_{11} = \forall\Diamond black & \Omega_{12} = \forall(green \bigcup (\forall(yellow \bigcup red))) \end{array}$$

c) Welche dieser Formeln sind syntaktisch inkorrekt? Begründen Sie Ihre Entscheidung.

d) Überführen Sie die syntaktisch korrekten CTL-Formeln in ihre existenzielle Normalform (ENF).

e) Konstruieren Sie die Syntaxbäume *(parse trees)* für die syntaktisch korrekten CTL-Formeln in ENF. Annotieren Sie die Knoten der Syntaxbäume mit den dazugehörigen Erfüllbarkeitsmengen $Satisfy_{\mathcal{K}}(\Omega)$ bzgl. der Kripke-Struktur $\mathcal{K}$.

f) Entscheiden Sie nun, welche dieser Formeln von $\mathcal{K}$ erfüllt werden und welche nicht.

g) Geben Sie für alle unerfüllbaren Formeln, die mit einem All-Quantor ($\forall$) beginnen, einen Pfad an, der die Formel falsifiziert.

2 Binary Decision Diagrams

Gegeben sei die Boolsche Funktion $f(a,b,c,d) = \neg dab + \neg ad\neg c + abd + \neg a\neg c\neg d$.

a) Erstellen Sie die Funktionstabelle von f.

b) Erzeugen Sie nun den zu f äquivalenten Binary Decision Tree (BDT).

c) Überführen Sie den BDT in ein Binary Decision Diagram (BDD).

d) Minimieren Sie gegebenenfalls das BDD und notieren Sie die daraus ersichtliche, vereinfachte Boolsche Funktion f'. Ist f' äquivalent zu f?

Hinweis: Nützen Sie Isomorphismen und Don't Care-Knoten!

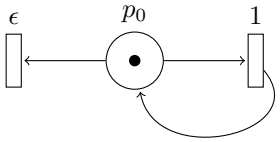
3 Petrinetze & Sprachen

Wir betrachten die Sprache $\mathcal{L} = \{w \in (0 \cup 1)^* \mid w \text{ enthält gleich viele 0en wie 1en}\}$.

a) Geben Sie eine kontextfreie Grammatik an, die $\mathcal{L}$ beschreibt.

b) Zeichnen Sie den zugehörigen Pushdown-Automaten zu $\mathcal{L}$.

Sind die Transitionen eines Petrinetzes P mit Symbolen aus einem Sprachalphabet Σ beschriftet, so kann P auch eine Sprache beschreiben. Spielt man das Token-Game so entspricht die Sequenz der gefeuerten Transitionen einem String bestehend aus Symbolen aus Σ. Ein Wort w wird von P genau dann *akzeptiert*, wenn es einer gültigen Feuersequenz σ_w in P entspricht und P nach Ausführen von σ_w tot[1] ist.
Das folgende Petrinetz akzeptiert bspw. das Wort $v = 11$, weil es eine entsprechende Feuersequenz $\sigma_v = 1, 1, \epsilon$ gibt, nach deren Ausführung das Netz tot ist.



Dieses Petrinetz akzeptiert übrigens die Sprache $L = 1^*$.

c) Geben Sie ein Petrinetz an, das $\mathcal{L}$ erkennt.

d) Ist Ihr Petrinetz *k-bounded* für ein bestimmtes k? Begründen Sie Ihre Antwort.

[1] Ein Petrinetz ist *tot* wenn es keine aktivierte Transition mehr gibt.