

Automatentheorie und ihre Anwendungen

Teil 3: Automaten auf unendlichen Wörtern

Thomas Schneider

28. Mai – 16. Juli 2014

Überblick

- 1 *Motivation und Beispiele*
- 2 Grundbegriffe und Büchi-Automaten
- 3 Abschlusseigenschaften
- 4 Charakterisierung
- 5 Deterministische Büchi-Automaten und Determinisierung
- 6 Entscheidungsprobleme
- 7 *Anwendung: Model-Checking in Linearer Temporallogik (LTL)*

Und nun ...

- 1 *Motivation und Beispiele*
- 2 Grundbegriffe und Büchi-Automaten
- 3 Abschlusseigenschaften
- 4 Charakterisierung
- 5 Deterministische Büchi-Automaten und Determinisierung
- 6 Entscheidungsprobleme
- 7 *Anwendung: Model-Checking in Linearer Temporallogik (LTL)*

Terminierung

Terminierung von Algorithmen ist wichtig für Problemlösung.

Übliches Szenario:

- Eingabe: endliche Menge von Daten
- Lasse Programm P laufen, bis es **terminiert**
- Ausgabe: Ergebnis, das durch P berechnet wurde

Um Ausgabe zu erhalten, **muss P für jede Eingabe terminieren.**

Beispiel: Validierung von XML-Dokumenten für gegebenes Schema

- Konstruiere Automaten für Schema und Dokument (**terminiert**)
- Reduziere auf Leerheitsproblem (**terminiert**)
- Löse Leerheitsprob. (sammle erreichb. Zustände – **terminiert**)

Terminierung unerwünscht

Von manchen Systemen/Programmen fordert man, dass sie **nie terminieren**.

Beispiele:

- (Mehrbenutzer-)Betriebssysteme
sollen beliebig lange laufen ohne abzustürzen, egal was Benutzer tun
- Bankautomaten, Flugsicherungssysteme, Netzwerkkommunikationssysteme, ...

Gängiges Berechnungsmodell:

- endliche Automaten mit nicht-terminierenden Berechnungen
- Terminierung wird als Nicht-Akzeptanz angesehen
- ursprünglich durch Büchi entwickelt (1960)
Ziel: Algorithmen zur Entscheidung mathematischer Theorien

Ziel dieses Kapitels

- Beschreibung von Automatenbegriffen mit nicht-terminierenden Berechnungen
- Betrachtung **unendlicher** Eingaben und Berechnungen
- ausgiebiges Studium von Büchi-Automaten und der von ihnen erkannten Sprachen:
 - Definition, Abschlusseigenschaften
 - Charakterisierung mittels regulärer Sprachen
 - Determinisierung
 - Entscheidungsprobleme
- Anwendung: Spezifikation und Verifikation in Linearer Temporallogik (LTL)

Beispiel 1: Philosophenproblem (Dining Philosophers Problem)

- erläutert Nebenläufigkeit und Verklemmung von Prozessen
- demonstriert auch unendliche Berechnungen
- hier: einfachste Version mit 3 Philosophen

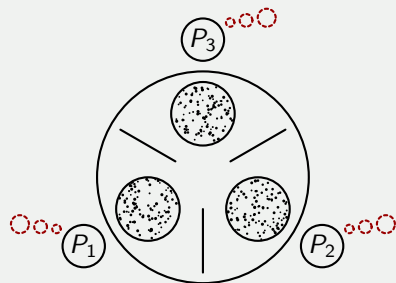
Philosophenproblem

- 3 Philosophen P_1, P_2, P_3
 - Für alle i gilt: entweder denkt P_i , oder P_i isst.
 - Alle P_i sitzen um einen runden Tisch.
 - Jeder P_i hat einen Teller mit Essen vor sich.
 - Zwischen je zwei Tellern liegt ein Esstübchen.
 - Um zu essen, muss P_i beide Stübchen neben seinem Teller benutzen.
- ⇒ Keine zwei P_i, P_j können gleichzeitig essen.

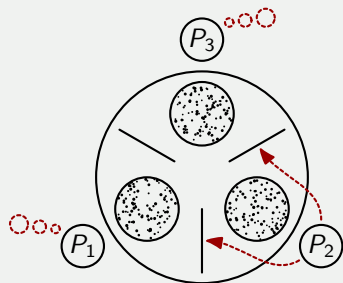
Skizze zum Philosophenproblem

Zusammenfassung

- Für alle i : entweder denkt P_i , oder P_i isst.
- Keine zwei P_i, P_j können gleichzeitig essen.



P_1, P_2, P_3 denken.



P_1, P_3 denken; P_2 isst.

Modellierung durch endliches Transitionssystem

Annahmen

- Am Anfang denken (**d**) alle P_i .
- Reihum können sich P_1, P_2, P_3 entscheiden, ob sie denken oder essen (**e**) wollen.

Zustände des Systems

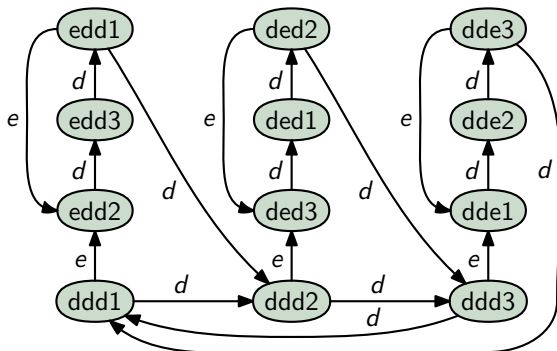
- Anfangszustand ddd1:
alle P_i denken, und P_1 trifft nächste Entscheidung.
- alle zulässigen Zustände:

ddd1	edd1	ded1	dde1
ddd2	edd2	ded2	dde2
ddd3	edd3	ded3	dde3

Zustandsüberführungen:

d oder e – je nach Entscheidung des P_i , der an der Reihe ist

Das Transitionssystem



Was sind die Eingaben in das System?

Endliche Zeichenketten über $\Sigma = \{d, e\}$?

Dann ist das System ein NEA.

► **Unendliche Zeichenketten über $\Sigma = \{d, e\}$!**

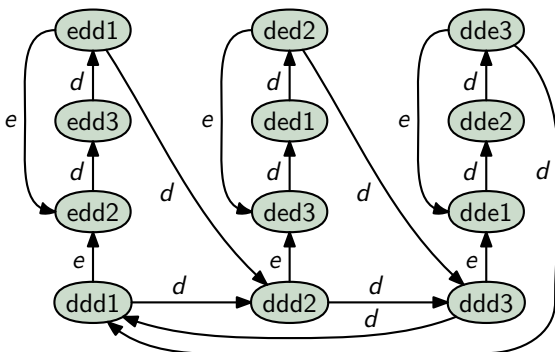
Warum unendliche Zeichenketten?

- Nehmen an, jeder P_i möchte **beliebig oft** denken und essen.
Dann ist P_i **zufrieden**.
- System soll **beliebig lange** ohne Terminierung laufen.

↪ mögliche Fragen:

- 1 Ist es überhaupt möglich, dass das System beliebig lange läuft?
- 2 Ist es zusätzlich möglich, dass P_i zufrieden ist?
- 3 Ist es möglich, dass P_1, P_2 zufrieden sind, aber P_3 nicht?
- 4 Ist es möglich, dass alle P_i zufrieden sind?

Frage 1



Ist es überhaupt möglich, dass das System beliebig lange läuft?

Ja: jeder Zustand hat mindestens einen Nachfolgerzustand.
 $dddddd\dots$ ist ein möglicher unendlicher Lauf.

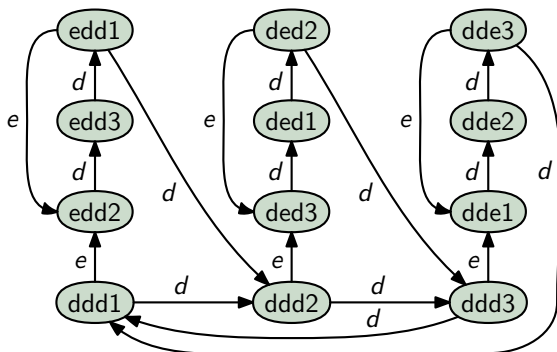
Frage 2



Ist es möglich, dass P_i zufrieden ist?

Ja: z. B. wenn ein Lauf ddd1 und edd1 unendlich oft durchläuft:
 $ed^5ed^5 \dots$

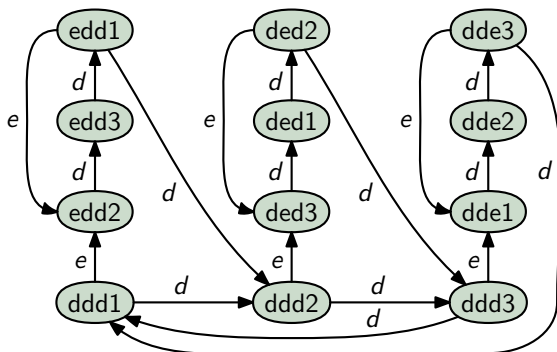
Frage 3



Ist es möglich, dass P_1, P_2 zufrieden sind, aber P_3 nicht?

Ja: z. B. ddd1, edd1, ddd2, ded2 unendlich oft, aber ddei nicht:
 $ed^3ed^4ed^3ed^4 \dots$

Frage 4



Ist es möglich, dass alle P_i zufrieden sind?

Ja: z. B. ddd1, edd1, ddd2, ded2, ddd3, dde3 unendlich oft:
 $ed^3ed^3\dots$ oder $ed^2ed^3ed^2ed^3\dots$ oder \dots

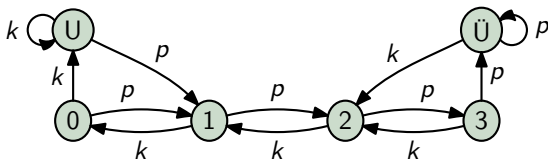
Beispiel 2: Konsument-Produzent-Problem

- P erzeugt Produkte und legt sie einzeln in einem Lager ab
- K entnimmt Produkte einzeln dem Lager
- Lager fasst maximal 3 Stück

Modellierung durch endliches Transitionssystem

- Zustände $0, 1, 2, 3, \bar{U}, U$
 - $0,1,2,3$: im Lager liegen $0,1,2,3$ Stück
 - \bar{U} (Überschuss): P will ein Stück im vollen Lager ablegen
 - U (Unterversorgung): K will ein Stück aus leerem Lager nehmen
- Aktionen P, K (P legt ab oder K entnimmt)

Das Transitionssystem



Eingaben in das System: unendliche Zeichenketten über $\Sigma = \{p, k\}$

Zufriedenheit: P (K) möchte ...

- beliebig oft Produkte produzieren (konsumieren)
- nur endlich oft Überschuss (Unterversorgung) erleiden

Sequenz, die P und K zufrieden stellt:

$p^3k^3p^3k^3 \dots$ oder $ppkpkpk \dots$ oder ...

Sequenz, die weder P noch K zufrieden stellt: $p^4k^4p^4k^4 \dots$

Und nun ...

- 1 *Motivation und Beispiele*
- 2 Grundbegriffe und Büchi-Automaten**
- 3 *Abschlusseigenschaften*
- 4 *Charakterisierung*
- 5 *Deterministische Büchi-Automaten und Determinisierung*
- 6 *Entscheidungsprobleme*
- 7 *Anwendung: Model-Checking in Linearer Temporallogik (LTL)*

Grundbegriffe

Unendliches Wort über Alphabet Σ

- ist Funktion $a : \mathbb{N} \rightarrow \Sigma$
- $a(n)$: Symbol an n -ter Stelle (auch: a_n)
- wird oft geschrieben als $a_0 a_1 a_2 \dots$
- $a[m, n]$: endliche Teilfolge $a_m a_{m+1} \dots a_n$

Σ^ω : Menge aller unendlichen Zeichenketten

ω -Sprache: $L \subseteq \Sigma^\omega$

Büchi-Automaten

Definition 1

Ein **nichtdeterministischer Büchi-Automat (NBA)** über einem **Alphabet** Σ ist ein 5-Tupel $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, wobei

- Q eine endliche nichtleere **Zustandsmenge** ist,
- Σ eine endliche nichtleere Menge von Zeichen ist,
- $\Delta \subseteq Q \times \Sigma \times Q$ die **Überführungsrelation** ist,
- $I \subseteq Q$ die Menge der **Anfangszustände** ist,
- $F \subseteq Q$ die Menge der **Endzustände** ist.

... bisher kein Unterschied zu NEAs!

Berechnungen und Akzeptanz

Definition 2

Sei $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ ein Büchi-Automat.

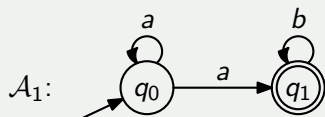
- Ein **Run** von \mathcal{A} auf $w = a_0a_1a_2\dots$ ist eine Folge

$$r = q_0q_1q_2\dots,$$

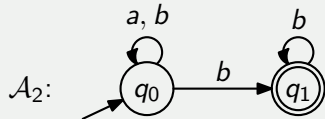
so dass für alle $i \geq 0$ gilt: $(q_i, a_i, q_{i+1}) \in \Delta$.

- **Unendlichkeitsmenge** $\text{Inf}(r)$ von $r = q_0q_1q_2\dots$:
Menge der Zustände, die unendlich oft in r vorkommen
- **Erfolgreicher Run** $r = q_0q_1q_2\dots$: $q_0 \in I$ und $\text{Inf}(r) \cap F \neq \emptyset$
- \mathcal{A} **akzeptiert** w ,
wenn es einen erfolgreichen Run von \mathcal{A} auf w gibt.
- Die von \mathcal{A} **erkannte Sprache** ist
 $L_w(\mathcal{A}) = \{w \in \Sigma^\omega \mid \mathcal{A} \text{ akzeptiert } w\}$.

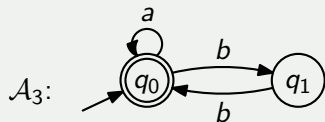
Beispiele



$$L_\omega(\mathcal{A}_1) = \{a^n b^\omega \mid n \geq 1\}$$



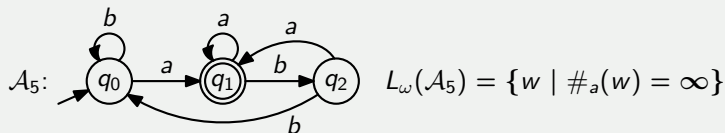
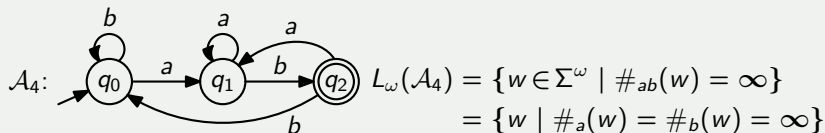
$$L_\omega(\mathcal{A}_2) = \{w \in \Sigma^\omega \mid \#_a(w) < \infty\}$$



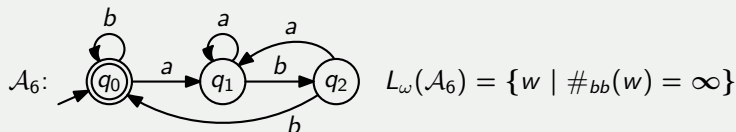
$$L_\omega(\mathcal{A}_3) = \{w \in \Sigma^\omega \mid \text{s. unten}\}$$

Zwischen zwei aufeinanderfolgenden a 's in w
 – und am Anfang von w – steht eine gerade Anzahl von b 's.

Mehr Beispiele



(Idee: q_1 kann nur erreicht werden, wenn ein a gelesen wird)



(Idee: q_0 nur durch bb erreichbar; jeder Teilstring bb führt zu q_0)

Erkennbare Sprache

Definition 3

Eine Sprache $L \subseteq \Sigma^\omega$ ist **Büchi-erkennbar**, wenn es einen NBA \mathcal{A} gibt mit $L = L_\omega(\mathcal{A})$.

Und nun ...

- 1 *Motivation und Beispiele*
- 2 Grundbegriffe und Büchi-Automaten
- 3 Abschlusseigenschaften**
- 4 Charakterisierung
- 5 Deterministische Büchi-Automaten und Determinisierung
- 6 Entscheidungsprobleme
- 7 *Anwendung: Model-Checking in Linearer Temporallogik (LTL)*

Zur Erinnerung

Die Menge der Büchi-erkennbaren Sprachen ist abgeschlossen unter

- **Vereinigung**, wenn gilt:
Falls L_1, L_2 Büchi-erkennbar, so auch $L_1 \cup L_2$.
- **Durchschnitt**, wenn gilt:
Falls L_1, L_2 Büchi-erkennbar, so auch $L_1 \cap L_2$.
- **Komplement**, wenn gilt:
Falls L Büchi-erkennbar, so auch \bar{L} .

Quiz

Unter welchen Operationen gilt Abgeschlossenheit, und wie leicht ist das zu zeigen?

Vereinigung?	✓	(leicht)
Durchschnitt?	✓	(mittel)
Komplement?	✓	(schwer)

Abgeschlossenheit

Satz 4

Die Menge der Büchi-erkennbaren Sprachen ist abgeschlossen unter den Operationen \cup und \cap .

Beweis: Direkte Konsequenz aus den folgenden Lemmata. □

Abgeschlossenheit unter $\bar{}$: siehe Abschnitt „Determinisierung“

Abgeschlossenheit unter Vereinigung

Lemma 5

Seien $\mathcal{A}_1, \mathcal{A}_2$ NBAs über Σ .

Dann gibt es einen NBA \mathcal{A}_3 mit $L_\omega(\mathcal{A}_3) = L_\omega(\mathcal{A}_1) \cup L_\omega(\mathcal{A}_2)$.

Beweis: Seien $\mathcal{A}_i = (Q_i, \Sigma, \Delta_i, I_i, F_i)$ für $i = 1, 2$.

O. B. d. A. gelte $Q_1 \cap Q_2 = \emptyset$.

Konstruieren $\mathcal{A}_3 = (Q_3, \Sigma, \Delta_3, I_3, F_3)$ wie folgt.

► *Idee wie für NEAs: vereinige \mathcal{A}_1 und \mathcal{A}_2 .*

- $Q_3 = Q_1 \cup Q_2$
- $\Delta_3 = \Delta_1 \cup \Delta_2$
- $I_3 = I_1 \cup I_2$
- $F_3 = F_1 \cup F_2$

Dann gilt $L_\omega(\mathcal{A}_3) = L_\omega(\mathcal{A}_1) \cup L_\omega(\mathcal{A}_2)$. □

Abgeschlossenheit unter Durchschnitt

Zur Erinnerung, für NEAs:

Gegeben $\mathcal{A}_1, \mathcal{A}_2$, konstruiere \mathcal{A}_3 mit $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$:

► *Idee: lasse \mathcal{A}_1 und \mathcal{A}_2 „gleichzeitig“ auf Eingabewort laufen.*

- $Q_3 = Q_1 \times Q_2$
- $\Delta_3 = \{((p, p'), a, (q, q')) \mid (p, a, q) \in \Delta_1 \ \& \ (p', a, q') \in \Delta_2\}$
- $I_3 = I_1 \times I_2$
- $F_3 = F_1 \times F_2$

Funktioniert das auch für Büchi-Automaten?

Nein. Wir bekommen Probleme mit Erreichbarkeit.
Beispiel siehe Tafel.



Abgeschlossenheit unter Durchschnitt

Lemma 6

Seien $\mathcal{A}_1, \mathcal{A}_2$ NBAs über Σ .

Dann gibt es einen NBA \mathcal{A}_3 mit $L_\omega(\mathcal{A}_3) = L_\omega(\mathcal{A}_1) \cap L_\omega(\mathcal{A}_2)$.

Beweis: siehe Tafel



Abgeschlossenheit unter Komplement

... siehe Abschnitt

„Deterministische Büchi-Automaten und Determinisierung“

Und nun ...

- 1 *Motivation und Beispiele*
- 2 Grundbegriffe und Büchi-Automaten
- 3 Abschlusseigenschaften
- 4 Charakterisierung**
- 5 Deterministische Büchi-Automaten und Determinisierung
- 6 Entscheidungsprobleme
- 7 *Anwendung: Model-Checking in Linearer Temporallogik (LTL)*

Ziel

Ziel dieses Abschnitts

Charakterisierung der Büchi-erkennbaren Sprachen
mittels regulärer Sprachen

Etwas Notation

Seien $W \subseteq \Sigma^*$ und $L \subseteq \Sigma^\omega$.

- $W^\omega = \{w_0w_1w_2\cdots \mid w_i \in W \setminus \{\varepsilon\} \text{ für alle } i \geq 0\}$
(ist ω -Sprache, weil ε ausgeschlossen wurde)
- $WL = \{wv \mid w \in W, v \in L\}$
(ist ω -Sprache)

Von regulären zu Büchi-erkennbaren Sprachen (1)

Lemma 7

Für jede reguläre Sprache $W \subseteq \Sigma^*$ gilt: W^ω ist Büchi-erkennbar.

Beweis.

- Sei $\mathcal{A}_1 = (Q_1, \Sigma, \Delta_1, \{q_0\}, F_1)$ ein NEA mit $L(\mathcal{A}_1) = W \setminus \{\varepsilon\}$.
- Idee: konstruiere NBA, der
 - \mathcal{A}_1 simuliert, bis ein Endzustand erreicht ist und
 - dann nichtdeterministisch entscheidet, ob die Simulation fortgesetzt wird oder eine neue Simulation von q_0 aus gestartet wird
- Details: siehe Tafel ● □

Von regulären zu Büchi-erkennbaren Sprachen (2)

Lemma 8

Für jede reguläre Sprache $W \subseteq \Sigma^$
und jede Büchi-erkennbare Sprache $L \subseteq \Sigma^\omega$ gilt:
 WL ist Büchi-erkennbar.*

Beweis:

Wie Abgeschlossenheit der reg. Sprachen unter Konkatenation. \square

Satz von Büchi

Satz 9

Eine Sprache $L \subseteq \Sigma^\omega$ ist Büchi-erkennbar genau dann, wenn es reguläre Sprachen $V_1, W_1, \dots, V_n, W_n$ gibt mit $n \geq 1$ und

$$L = V_1 W_1^\omega \cup \dots \cup V_n W_n^\omega$$

Beweis:

Siehe Tafel.



Konsequenz:

Büchi-erkennbare Sprachen durch ω -reguläre **Ausdrücke** darstellbar:

$$r_1 s_1^\omega + \dots + r_n s_n^\omega,$$

wobei r_i, s_i reguläre Ausdrücke sind

Und nun ...

- 1 *Motivation und Beispiele*
- 2 Grundbegriffe und Büchi-Automaten
- 3 Abschlusseigenschaften
- 4 Charakterisierung
- 5 Deterministische Büchi-Automaten und Determinisierung**
- 6 Entscheidungsprobleme
- 7 *Anwendung: Model-Checking in Linearer Temporallogik (LTL)*

Ziel dieses Abschnitts

Wollen zeigen:

- det. und nichtdet. Büchi-Automaten sind **nicht** gleichmächtig
d. h.: es gibt ω -Sprachen, die von NBAs akzeptiert werden,
aber nicht von DBAs
- Komplement-Abgeschlossenheit gilt trotzdem

Etwas Notation:

- **DBA:** NBA $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ mit
 - $|I| = 1$
 - $|\{q' \mid (q, a, q') \in \Delta\}| = 1$ für alle $(q, a) \in Q \times \Sigma$
- Sei $W \subseteq \Sigma^*$.
 $\vec{W} = \{w \in \Sigma^\omega \mid w[0, n] \in W \text{ für unendlich viele } n\}$
 (d. h. w hat ∞ viele Präfixe in W)

Beispiel: siehe Tafel



Zu Hilfe: Charakterisierung der DBA-erkennbaren Sprachen

Satz 10

Eine ω -Sprache $L \subseteq \Sigma^\omega$ ist DBA-erkennbar genau dann, wenn es eine reguläre Sprache $W \subseteq \Sigma^$ gibt mit $L = \overrightarrow{W}$.*

Beweis.

- Idee: sieh beliebigen D?A \mathcal{A} gleichzeitig als DEA und DBA an.
 $\rightsquigarrow L_\omega(\mathcal{A}) = \overrightarrow{L(\mathcal{A})}$.
- Details: siehe Tafel. ● □

DBAs sind schwächer als NBAs

Satz 11

Es gibt eine Büchi-erkennbare Sprache, die nicht durch einen DBA erkannt wird.

Beweis.

- Betrachte $L = \{w \in \{a, b\}^\omega \mid \#_a(w) \text{ ist endlich}\}$
- L ist Büchi-erkennbar: $L = \Sigma^* \{b\}^\omega$, wende Satz 9 an
- Annahme, L sei DBA-erkennbar.
 - \Rightarrow Satz 10: $L = \overrightarrow{W}$ für eine reguläre Sprache W
 - \Rightarrow Wegen $b^\omega \in L$ gibt es ein nichtleeres Wort $b^{n_1} \in W$
 - Wegen $b^{n_1} a b^\omega \in L$ gibt es ein nichtleeres Wort $b^{n_1} a b^{n_2} \in W$
 - \vdots
 - $\Rightarrow w := b^{n_1} a b^{n_2} a b^{n_3} \dots \in \overrightarrow{W}$ **Widerspruch:** $w \notin L$ \square

Nebenprodukt des letzten Beweises

DBAs sind **nicht** unter Komplement abgeschlossen:

- $L = \{w \in \{a, b\}^\omega \mid \#_a(w) \text{ ist endlich}\}$
wird von keinem DBA erkannt
- aber \bar{L} wird von einem DBA erkannt (Ü)

Wie können wir trotzdem determinisieren?

Indem wir das Automatenmodell ändern!

Genauer: ändern die Akzeptanzbedingung

Zur Erinnerung

NBA ist 5-Tupel $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ mit

- ...
- $F \subseteq Q$ (Menge der Endzustände)

Erfolgreicher Run: $r = q_0q_1q_2\dots$ mit $q_0 \in I$ und $\text{Inf}(r) \cap F \neq \emptyset$

Idee: r erfolgreich \Leftrightarrow ein Zustand aus F kommt ∞ oft in r vor

(Julius Richard Büchi, 1924–1984, Logiker/Mathematiker; Zürich, Lafayette)

Muller-Automaten

(David E. Muller, 1924–2008, Math./Inf.; Illinois)

Definition 12

Nichtdet. **Muller-Automat** ist 5-Tupel $\mathcal{A} = (Q, \Sigma, \Delta, I, \mathcal{F})$ mit

- ...
- $\mathcal{F} \subseteq 2^Q$ (Kollektion von Endzustandsmengen)

Erfolgreicher Run $r = q_0 q_1 q_2 \dots$ mit $q_0 \in I$ und $\underbrace{\text{Inf}(r)} \in \mathcal{F}$

Idee: r erfolgreich $\Leftrightarrow \text{Inf}(r)$ stimmt mit einer Menge aus \mathcal{F} überein

Beispiel: Siehe Tafel



Rabin-Automaten (Michael O. Rabin, *1931, Inf.; Jerusalem, Princeton, Harvard)

Definition 13

Nichtdet. **Rabin-Automat** ist 5-Tupel $\mathcal{A} = (Q, \Sigma, \Delta, I, \mathcal{P})$ mit

- ...
- $\mathcal{P} = \{(E_1, F_1), \dots, (E_n, F_n)\}$ mit $E_i, F_i \subseteq Q$
(Menge „akzeptierender Paare“)

Erfolgreicher Run $r = q_0q_1q_2\dots$ mit $q_0 \in I$ und

$$\exists i \in \{1, \dots, n\} \quad \text{mit} \quad \text{Inf}(r) \cap E_i = \emptyset \quad \text{und} \quad \text{Inf}(r) \cap F_i \neq \emptyset$$

Idee: r erfolgreich \Leftrightarrow es gibt Paar (E_i, F_i) , so dass

- **mindestens ein** Zustand aus F_i unendlich oft in r vorkommt &
- **alle** Zustände aus E_i nur endlich oft in r vorkommen (**Bsp. ●**)

Streett-Automaten

(Robert S. Streett, ?; Boston, Oakland)

Definition 14

Nichtdet. **Streett-Automat** ist 5-Tupel $\mathcal{A} = (Q, \Sigma, \Delta, I, \mathcal{P})$ mit

- ...
- $\mathcal{P} = \{(E_1, F_1), \dots, (E_n, F_n)\}$ mit $E_i, F_i \subseteq Q$
(Menge „fairer Paare“)

Erfolgreicher Run $r = q_0q_1q_2\dots$ mit $q_0 \in I$ und

$\forall i \in \{1, \dots, n\} : \text{wenn } \text{Inf}(r) \cap F_i \neq \emptyset, \text{ dann } \text{Inf}(r) \cap E_i \neq \emptyset$

Idee: r erfolgreich \Leftrightarrow **für alle** Paare (E_i, F_i) gilt:

- **wenn** ein Zustand aus F_i unendlich oft in r vorkommt,
- **dann** kommt ein Zustand aus E_i unendlich oft in r vor (**Bsp. ●**)

Gleichmächtigkeit der vier Automatenmodelle

Satz 15

Für jede Sprache $L \subseteq \Sigma^\omega$ sind die folgenden Aussagen äquivalent.

- (B) L ist Büchi-erkennbar.
- (M) L ist Muller-erkennbar.
- (R) L ist Rabin-erkennbar.
- (S) L ist Streett-erkennbar.

Beweis.

- (B), (R), (S) \rightarrow (M):
kodierte F bzw. \mathcal{P} in \mathcal{F} (s. Tafel) ●
- (B) \rightarrow (R), (S):
ersetze F durch das Paar (\emptyset, F) bzw. (F, Q) (s. Tafel) ●
- (M) \rightarrow (B):
komplexere Transformation, benutzt Nichtdeterm. \downarrow □

Von Muller- zu Büchi-Automaten

Lemma 16

Jede Muller-erkennbare Sprache ist Büchi-erkennbar.

Beweis.

- Sei $\mathcal{A} = (Q, \Sigma, \Delta, I, \mathcal{F})$ ein Muller-Automat
 - Dann ist $L_\omega(\mathcal{A}) = \bigcup_{F \in \mathcal{F}} L_\omega((Q, \Sigma, \Delta, I, \{F\}))$
- \rightsquigarrow Wegen \cup -Abgeschlossenheit: nehmen o. B. d. A. $\mathcal{F} = \{F\}$ an
- Konstruiere Büchi-Automaten $\mathcal{A}' = (Q', \Sigma, \Delta', I, F')$, der
 - \mathcal{A} simuliert
 - einen Zeitpunkt rät,
ab dem nur noch Zustände aus F vorkommen
 - ab dort sicherstellt, dass jedes q_i unendlich oft vorkommt
 - Details: siehe Tafel ● □

Gleichmächtigkeit der deterministischen Varianten

Definition 17 (Determinismus)

Ein Büchi-, Muller-, Rabin- oder Streett-Automat $\mathcal{A} = (Q, \Sigma, \Delta, I, Acc)$ ist **deterministisch**, wenn gilt:

- $|I| = 1$
- $\{q' \mid (q, a, q') \in \Delta\} = 1$ für alle $(q, a) \in Q \times \Sigma$

Zu Satz 15 analoge Aussage:

Satz 18

Für jede Sprache $L \subseteq \Sigma^\omega$ sind die folgenden Aussagen äquivalent.

- (M) *L ist von einem deterministischen Muller-Autom. erkennbar.*
- (R) *L ist von einem deterministischen Rabin-Autom. erkennbar.*
- (S) *L ist von einem deterministischen Streett-Autom. erkennbar.*

Ohne Beweis (Variante des Beweises von Satz 15).

Abschlusseigenschaften

Zur Erinnerung

Satz 4

Die Menge der Büchi-erkennbaren Sprachen ist abgeschlossen unter den Operationen \cup und \cap .

Direkte Konsequenz

Folgerung 19

Die Menge der

- *Muller-erkennbaren Sprachen,*
- *Rabin-erkennbaren Sprachen,*
- *Streett-erkennbaren Sprachen*

ist abgeschlossen unter den Operationen \cup und \cap .

Zu Komplement-Abgeschlossenheit kommen wir jetzt.

Determinisierung von Büchi-Automaten

Zur Erinnerung: Satz 11

Es gibt eine Büchi-erkennbare Sprache,
die nicht durch einen DBA erkannt wird.

Ziel: Prozedur, um gegebenen NBA in äquivalenten
deterministischen **Rabin**-Automaten umzuwandeln

- ↪ wegen Satz 18 bekommt man daraus auch eine Umwandlung
in einen äquiv. determ. Muller- bzw. Streett-Automaten
- Resultat geht auf McNaughton zurück
(1965 von Robert McNaughton, ?, Phil./Inform.; Harvard, Rensselaer)
 - Verwenden intuitiveren Beweis von Safra
(1988 von Shmuel Safra, ?, Informatiker; Tel Aviv)

Potenzmengenkonstruktion versagt

Erster naheliegender Versuch:

NBA \rightsquigarrow DBA mittels Potenzmengenkonstruktion (PMK)

- muss wegen Satz 11 fehlschlagen
- Beispiel: siehe Tafel



Zweiter naheliegender Versuch:

NBA \rightsquigarrow deterministischer X -Automat mittels PMK,
 $X \in \{\text{Muller, Rabin, Streett}\}$

- schlägt auch fehl – Beispiel siehe Tafel



Hauptproblem:

- Potenzautomat simuliert mehrere Runs gleichzeitig (wie Produktautomat)
- Endzustände müssen dabei nicht synchron erreicht werden

Safra's Ideen informell dargestellt

- **Ziel:**

$$\text{NBA } \mathcal{A} = (Q, \Sigma, \Delta, I, F) \rightsquigarrow \text{DRA } \mathcal{A}^d = (Q^d, \Sigma, \Delta^d, I^d, \mathcal{P}^d)$$
 mit $L_\omega(\mathcal{A}) = L_\omega(\mathcal{A}^d)$
- **Problem mit PMK:**
bad runs von \mathcal{A}^d , die keinem erfolgr. Run von \mathcal{A} entsprechen
- **Safra's Tricks** erweitern die PMK und vermeiden das Problem

Etwas Notation

- **Makrozustände:** Zustände der alten PMK (Mengen $M \subseteq Q$)
- Zustände von \mathcal{A}^d :
 \approx Bäume, deren Knoten mit Makrozuständen markiert sind

Safra's Tricks

Beginne wie bei der PMK mit Knoten /

- ① Von Makrozuständen mit Endzuständen, beginne neue Runs ●
 - erzeuge neues Kind mit Nachfolgezuständen aller Endzustände
 - wende zukünftig PMK auf jeden Knoten an

- ② Erkenne zusammenlaufende Runs; lösche überflüssige Info ●
 - das beschränkt Weite eines Safra-Baums
 - „horizontal merge“

- ③ Gib überflüssige Makrozustände zur Löschung frei ●
 - wenn alle Kinder eines MZ M bezeugen,
dass jeder Zustand in M einen Endzustand als Vorgänger hat,
dann kann M gelöscht werden
 - „vertical merge“

Safra-Bäume

„Bausteine“:

- Q : Menge von Zuständen \rightsquigarrow Makrozustände **MZ** $M \subseteq Q$
- V : Menge von Knotennamen

Safra-Baum über Q, V :

- geordneter Baum mit Knoten aus V
- jeder Knoten mit einem **nichtleeren** MZ markiert und möglicherweise auch mit $\textcircled{!}$
- Wenn Knoten v mit M und v 's Kinder mit M_1, \dots, M_n markiert sind, dann:
 - 1 $M_1 \cup \dots \cup M_n \subsetneq M$
 - 2 M_i sind paarweise disjunkt

Safra-Bäume sind beschränkt

Zur Erinnerung

Wenn Knoten v mit M und v 's Kinder mit M_1, \dots, M_n markiert sind, dann:

- 1 $M_1 \cup \dots \cup M_n \subsetneq M$
- 2 M_i sind paarweise disjunkt

Konsequenzen

- wegen (1): Höhe jedes SB ist durch $|Q|$ beschränkt
- wegen (2): Anzahl Kinder pro Knoten kleiner als $|Q|$
- sogar: Jeder SB über Q hat höchstens $|Q|$ Knoten (Beweis per Induktion über Baumhöhe)

Details der Konstruktion

Sei $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ ein NBA und $V = \{1, \dots, 2|Q|\}$.

Konstruieren DRA $\mathcal{A}^d = (Q^d, \Sigma, \Delta^d, I^d, \mathcal{P})$:

- $Q^d =$ Menge aller Safra-Bäume über Q, V
- $I^d =$ Safra-Baum mit einzigem Knoten I
- $\Delta^d = \{(S, a, S') \mid S' \text{ wird aus } S \text{ wie folgt konstruiert}\}$

Konstruktion von S' aus S in 6 Schritten

Sei S Safra-Baum mit Knoten $V' \subseteq V$; sei $a \in \Sigma$

- ① Beginne mit S ; entferne alle Markierungen
- ② Für jeden Knoten v mit Makrozustand M und $M \cap F \neq \emptyset$, füge neues Kind $v' \in V \setminus V'$ mit Markierung $M \cap F$ hinzu (als **jüngstes** (rechtes) Geschwister aller evtl. vorhandenen Kinder)
- ③ Wende Potenzmengenkonstruktion auf alle Knoten v an: ersetze MZ M durch $\{q \in Q \mid (m, a, q) \in \Delta \text{ für ein } m \in M\}$
- ④ **Horizontales Zusammenfassen:** Für jeden Knoten v mit MZ M , lösche jeden Zustand q aus M , der im MZ eines älteren Geschwisters vorkommt
- ⑤ Entferne alle Knoten mit leeren MZen
- ⑥ **Vertikales Zusammenfassen:** Für jeden Knoten v , dessen Markierung nur Zustände aus v 's Kindern enthält, lösche alle Nachfolger von v und markiere v mit

Erläuterungen zur Konstruktion

- S' ist wieder ein Safra-Baum:

Wenn Knoten v mit M und v 's Kinder mit M_1, \dots, M_n markiert sind, dann:

① $M_1 \cup \dots \cup M_n \subsetneq M$

“ \subseteq ”: Schritte 2, 3

“ \neq ”: Schritt 6

② M_i sind paarweise disjunkt

Schritt 4

- Beispiel: siehe Tafel



Akzeptanzkomponente von \mathcal{A}^d

$\mathcal{P} = \{(E_v, F_v) \mid v \in V\}$ mit

- $E_v =$ alle Safra-Bäume ohne Knoten v
- $F_v =$ alle Safra-Bäume, in denen v mit $\textcircled{!}$ markiert ist

\rightsquigarrow d. h. Run $r = S_0 S_1 S_2 \dots$ von \mathcal{A}^d ist erfolgreich,
wenn es einen Knotennamen v gibt, so dass

- alle S_i , bis auf endlich viele, einen Knoten v haben und
- unendlich oft auf v Schritt 6 angewendet wurde,
d. h. vorher kamen alle Zustände in v 's MZ in v 's Kindern vor

Korrektheit der Konstruktion: Vorbereitung

Lemma 20 (Lemma von König)

Jeder unendliche Baum mit endlichem Verzweigungsgrad hat einen unendlichen Pfad.

- ohne Beweis
- endlicher Verzweigungsgrad:
jeder Knoten hat endlich viele Kinder
- 1936 von Dénes König (1884–1944, Mathematiker, Budapest)

Korrektheit der Konstruktion

Korrektheit:

\mathcal{A}^d akzeptiert nur Wörter, die \mathcal{A} akzeptiert

Lemma 21

Sei $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ ein NBA und sei $\mathcal{A}^d = (Q^d, \Sigma, \Delta^d, I^d, \mathcal{P})$ der DRA, den man nach Safra's Konstruktion aus \mathcal{A} erhält.

Dann gilt $L_\omega(\mathcal{A}^d) \subseteq L_\omega(\mathcal{A})$.

Beweis: siehe Tafel.



Vollständigkeit der Konstruktion

Vollständigkeit:

\mathcal{A}^d akzeptiert (mindestens) alle Wörter, die \mathcal{A} akzeptiert

Lemma 22

Sei $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ ein NBA und sei $\mathcal{A}^d = (Q^d, \Sigma, \Delta^d, I^d, \mathcal{P})$ der DRA, den man nach Safra's Konstruktion aus \mathcal{A} erhält.

Dann gilt $L_\omega(\mathcal{A}) \subseteq L_\omega(\mathcal{A}^d)$.

Beweis: siehe Tafel.



Konsequenz aus Safras Konstruktion

Satz 23 (Satz von McNaughton)

Sei \mathcal{A} ein NBA. Dann gibt es einen DRA \mathcal{A}^d mit $L_\omega(\mathcal{A}^d) = L_\omega(\mathcal{A})$.

Beweis. Folgt aus Lemmas 21 und 22.

Folgerung 24

Die Klasse der Büchi-erkennbaren Sprachen ist unter Komplement abgeschlossen.

Beweis. Über folgende Transformationskette:

NBA für L \rightarrow DRA für L (gemäß Satz 23)
 \rightarrow DMA für L (gemäß Satz 18)
 \rightarrow DMA für \bar{L} (wie gehabt)
 \rightarrow NBA für \bar{L} (gemäß Satz 15) □

Anmerkungen zur Komplexität

Determinisierung NBA \rightarrow DRA gemäß Safra's Konstruktion

- liefert einen **exponentiell** größeren DRA
- genauer: wenn der NBA n Zustände hat,
 - gibt es 2^n Makrozustände
 - und $2^{O(n \log n)}$ Safrabäume \rightsquigarrow DRA hat $m := 2^{O(n \log n)}$ Zustände
- Das ist optimal (siehe Roggenbachs Kapitel in LNCS 2500)

Komplementierung beinhaltet auch den Schritt DMA \rightarrow NBA

- liefert einen nochmal **exponentiell** größeren DBA
- genauer: wenn der DMA m Zustände hat,
 - hat der NBA $O(m \cdot 2^m)$ Zustände \rightsquigarrow Resultierender NBA hat $2^{2^{O(n^2)}}$ Zustände
- Alternative Prozedur erfordert nur $2^{O(n \log n)}$ Zustände

Und nun ...

- 1 *Motivation und Beispiele*
- 2 Grundbegriffe und Büchi-Automaten
- 3 Abschlusseigenschaften
- 4 Charakterisierung
- 5 Deterministische Büchi-Automaten und Determinisierung
- 6 Entscheidungsprobleme**
- 7 *Anwendung: Model-Checking in Linearer Temporallogik (LTL)*

Vorbetrachtungen

Betrachten 4 Standardprobleme:

- Leerheitsproblem
- Wortproblem (Wort ist durch NBA gegeben)
- Äquivalenzproblem
- Universalitätsproblem

Beschränken uns auf das **Leerheitsproblem** – die anderen . . .

- lassen sich wie üblich darauf reduzieren
- aber teils mit (doppelt) exponentiellem „Blowup“
(Determinisierung, Komplementierung, siehe Folie 64)
↪ höhere, teils nicht optimale Komplexität

Beschränken uns auf **NBA**,
aber Entscheidbarkeit überträgt sich auf die anderen Modelle

Das Leerheitsproblem

Zur Erinnerung:

Gegeben: NBA \mathcal{A}

Frage: Gilt $L_\omega(\mathcal{A}) = \emptyset$?

Mengenschreibweise: $\{\text{NBA } \mathcal{A} \mid L_\omega(\mathcal{A}) = \emptyset\}$

Satz 25

Das Leerheitsproblem für NBAs ist entscheidbar.

Beweis: siehe Tafel. ●

Komplexität: NL-vollständig (Wegsuche in Graphen)

Und nun ...

- 1 *Motivation und Beispiele*
- 2 Grundbegriffe und Büchi-Automaten
- 3 Abschlusseigenschaften
- 4 Charakterisierung
- 5 Deterministische Büchi-Automaten und Determinisierung
- 6 Entscheidungsprobleme
- 7 *Anwendung: Model-Checking in Linearer Temporallogik (LTL)*

Reaktive Systeme und Verifikation

Reaktive Systeme

- interagieren mit ihrer Umwelt
- terminieren oft nicht
- Beispiele:
 - Betriebssysteme, Bankautomaten, Flugsicherungssysteme, ...
 - s. a. Philosophenproblem, Konsument-Produzent-Problem

Verifikation = Prüfen von Eigenschaften eines Systems

- Eingabe-Ausgabe-Verhalten hat hier keine Bedeutung
- Andere Eigenschaften sind wichtig,
z. B.: keine Verklemmung (deadlock) bei Nebenläufigkeit

Repräsentation eines Systems

Bestandteile

- **Variablen:** repräsentieren Werte, die zur Beschreibung des Systems notwendig sind
- **Zustände:** „Schnappschüsse“ des Systems
Zustand enthält Variablenwerte zu einem bestimmten Zeitpunkt
- **Transitionen:** erlaubte Übergänge zwischen Zuständen

Pfad (Berechnung) in einem System:

unendliche Folge von Zuständen entlang der Transitionen

Transitionsgraph als Kripke-Struktur

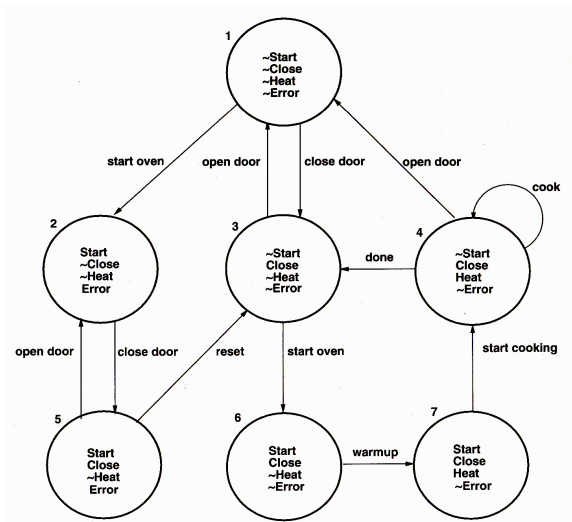
Definition 26

Sei AV eine Menge von Aussagenvariablen. Eine **Kripke-Struktur** \mathcal{S} über AV ist ein Quadrupel $\mathcal{S} = (S, S_0, R, L)$, wobei

- S eine endliche nichtleere Menge von **Zuständen** ist,
- $S_0 \subseteq S$ die Menge der **Anfangszustände** ist,
- $R \subseteq S \times S$ eine **Übergangsrelation** ist,
die **total ist**: $\forall s \in S \exists s' \in S : sRs'$
- $L : S \rightarrow 2^{AV}$ eine Funktion ist, die jeden Zustand mit der Menge von Aussagenvariablen markiert, die dort wahr sind.

Ein **Pfad** in \mathcal{S} ist eine endliche Folge $\pi = s_0s_1s_2 \dots$ von Zuständen mit $s_0 \in S_0$ und s_iRs_{i+1} für alle $i \geq 0$.

Beispiel 1: Mikrowelle



aus: E. M. Clarke et al., Model Checking, MIT Press 1999

Beispiel 2: nebenläufiges Programm

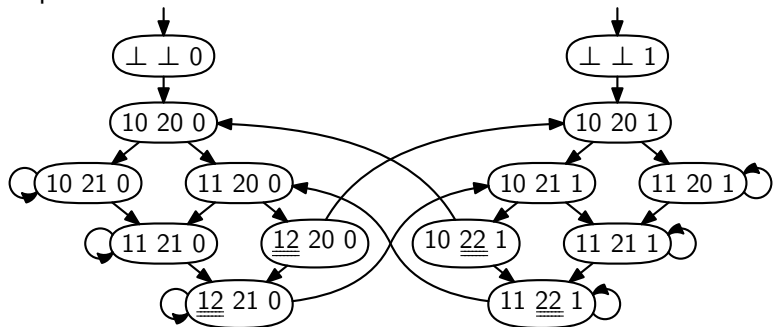
```
P    0  cobegin  
      1     $P_0 \parallel P_1$   
      2  coend  
  
P0 10  while(true) do  
      11    wait(turn = 0)  
      12    turn ← 1      kritischer Bereich  
      13  end while  
  
P1 20  while(true) do  
      21    wait(turn = 1)  
      22    turn ← 0      kritischer Bereich  
      23  end while
```

Beispiel 2: nebenläufiges Programm

Variablen in der zugehörigen Kripke-Struktur: v_1, v_2, v_3 mit

- v_1, v_2 : Werte der Programmzähler für P_0, P_1
(einschl. \perp : Teilprogramm ist nicht aktiv)
- v_3 : Werte der gemeinsamen Variable turn

Kripke-Struktur:



Spezifikationen

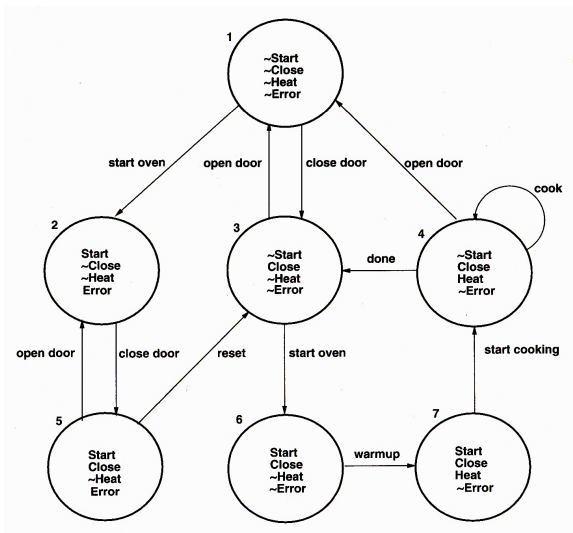
... sind Zusicherungen über die Eigenschaften eines Systems, z. B.:

- Wenn ein Fehler auftritt, ist er nach endlicher Zeit behoben.
- Wenn die Mikrowelle gestartet wird, fängt sie immer nach endlicher Zeit an zu heizen.
- Wenn die Mikrowelle gestartet wird, ist es *möglich*, danach zu heizen.

- Es kommt nie vor, dass beide Teilprogramme zugleich im kritischen Bereich sind.
- Jedes Teilprog. kommt beliebig oft in seinen krit. Bereich.
- Jedes Teilprogramm *kann* beliebig oft in seinen kritischen Bereich gelangen.

- ...

Spezifikationen für das Beispiel Mikrowelle

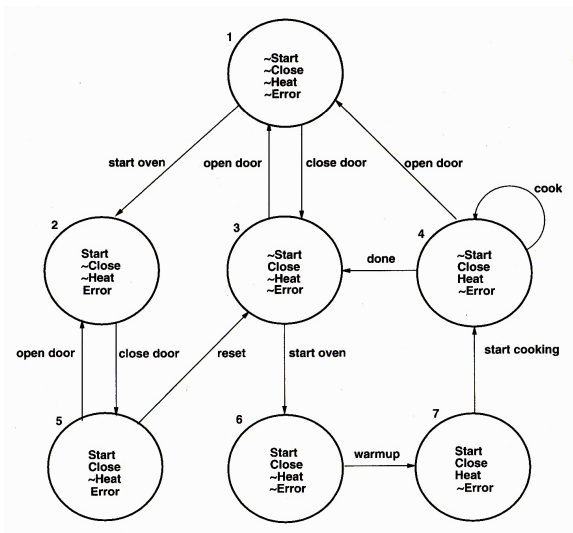


aus:

E. M. Clarke et al.,
Model Checking,
MIT Press 1999

„Wenn ein Fehler auftritt, ist er nach endlicher Zeit behoben.“ ✗

Spezifikationen für das Beispiel Mikrowelle

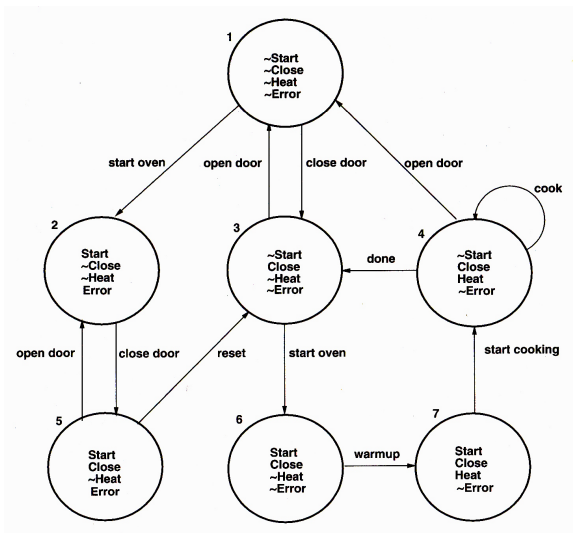


aus:

E. M. Clarke et al.,
Model Checking,
MIT Press 1999

„Wenn MW gestartet, beginnt sie immer nach endl. Zeit zu heizen.“ ✗

Spezifikationen für das Beispiel Mikrowelle

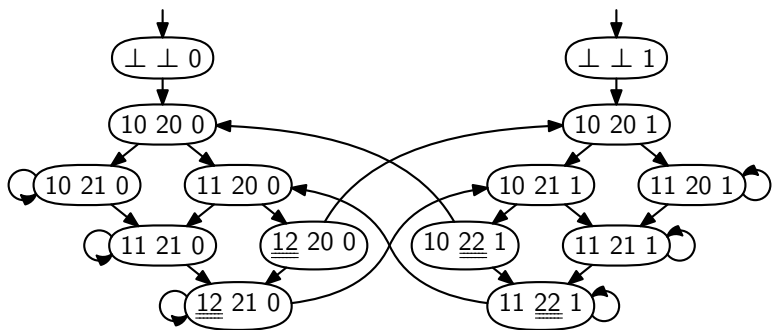


aus:

E. M. Clarke et al.,
Model Checking,
MIT Press 1999

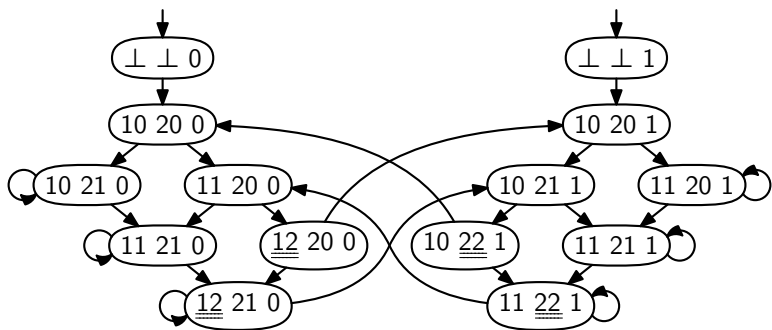
„Wenn MW gestartet, ist es *möglich*, danach zu heizen.“ ✓

Spezifikationen für das Beispiel Nebenläufigkeit



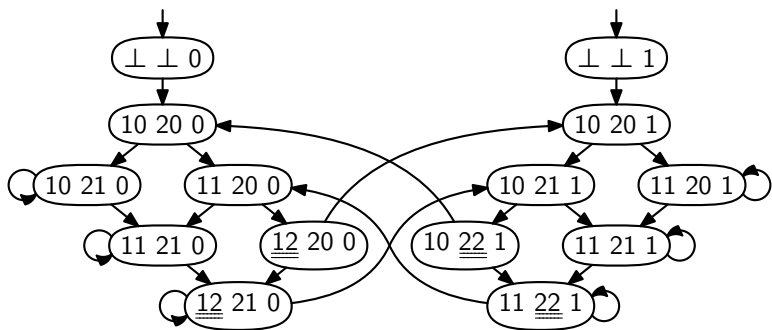
„Es kommt nie vor,
dass beide Teilprogramme zugleich im kritischen Bereich sind.“ ✓

Spezifikationen für das Beispiel Nebenläufigkeit



„Jedes P_i kommt beliebig oft in seinen kritischen Bereich.“ ✗

Spezifikationen für das Beispiel Nebenläufigkeit



„Jedes P_i kann beliebig oft in seinen kritischen Bereich kommen.“ ✓

Model-Checking

... beantwortet die Frage,
ob ein gegebenes System eine gegebene Spezifikation erfüllt

Definition 27 (Model-Checking-Problem MCP)

Gegeben ein System S und eine Spezifikation E ,

- gilt E für jeden Pfad in S ?
(**universelle Variante**)
- gibt es einen Pfad in S , der E erfüllt?
(**existenzielle Variante**)

Frage: Wie kann man Model-Checking

- exakt beschreiben und
- algorithmisch lösen?

Antwort: benutze Büchi-Automaten!

Vorgehen

- Stellen System \mathcal{S} als NBA $\mathcal{A}_{\mathcal{S}}$ dar
 \rightsquigarrow Pfade in \mathcal{S} sind erfolgreiche Runs von $\mathcal{A}_{\mathcal{S}}$

- Stellen Spezifikation E als NBA \mathcal{A}_E dar
 \rightsquigarrow \mathcal{A}_E beschreibt die Pfade, die E erfüllen

\rightsquigarrow Universelles MCP = „ $L(\mathcal{A}_{\mathcal{S}}) \subseteq L(\mathcal{A}_E)$?“

Existenzielles MCP = „ $L(\mathcal{A}_{\mathcal{S}}) \cap L(\mathcal{A}_E) \neq \emptyset$?“

Erweiterung (später)

- intuitivere Beschreibung von E mittels Temporallogik
- Umwandlung von Temporallogik-Formel φ_E in Automaten \mathcal{A}_E

Konstruktion des NBA $\mathcal{A}_{\mathcal{S}}$ für das System \mathcal{S}

Erinnerung: \mathcal{S} gegeben als Kripke-Struktur $\mathcal{S} = (S, S_0, R, L)$
(Zustände, Anfangszustände, Transitionen, Markierungen)

Zugehöriger Automat $\mathcal{A}_{\mathcal{S}} = (Q, \Sigma, \Delta, I, F)$:

- $\Sigma = 2^{AV}$
- $Q = S \uplus \{q_0\}$
- $I = \{q_0\}$
- $F = Q$
- $\Delta = \{ (q_0, L(s), s) \mid s \in S_0 \}$
 $\cup \{ (s, L(s'), s') \mid (s, s') \in R \}$

Beispiel: siehe Tafel. ●

Beschreibung von E durch NBA \mathcal{A}_E

Beispiel Mikrowelle (siehe Bild auf Folie 72)

- (a) Wenn ein Fehler auftritt, ist er nach endlicher Zeit behoben. ●
- (b) Wenn die Mikrowelle gestartet wird, fängt sie nach endlicher Zeit an zu heizen. ●
- (c) Wenn die Mikrowelle gestartet wird, ist es *möglich*, danach zu heizen. ●

Beispiel Nebenläufigkeit

- (d) Es kommt nie vor, dass beide Teilprog. zugleich im kritischen Bereich sind. ●
- (e) Jedes Teilprog. kommt beliebig oft in seinen krit. Bereich. ●
- (f) Jedes Teilprogramm *kann* beliebig oft in seinen kritischen Bereich gelangen. ●

Verifikation mittels der konstruierten NBAs

Gegeben sind wieder System \mathcal{S} und Spezifikation E .

Universelles MCP

- Gilt E für jeden Pfad in \mathcal{S} ?
- äquivalent: $L(\mathcal{A}_{\mathcal{S}}) \subseteq L(\mathcal{A}_E)$?
- äquivalent: $L(\mathcal{A}_{\mathcal{S}}) \cap \overline{L(\mathcal{A}_E)} = \emptyset$?

↪ Komplementierung \mathcal{A}_E , Produktautomat, Leerheitsproblem

- Komplexität: **PSPACE** (expon. Explosion bei Komplementierung)

Existenzielles MCP

- Gibt es einen Pfad in \mathcal{S} , der E erfüllt?
- äquivalent: $L(\mathcal{A}_{\mathcal{S}}) \cap L(\mathcal{A}_E) \neq \emptyset$?

↪ Produktautomat, Leerheitsproblem

- Komplexität: **NL** (keine exponentielle Explosion)

Bemerkungen zur Implementierung

- effizienterer Algorithmus zur Lösung des Leerheitsproblems
- „On-the-fly model checking“
 - $|S|$ ist exponentiell in der Anzahl der Variablen
State space explosion problem
 - Zustände von \mathcal{A}_S werden während des Leerheitstests nur bei Bedarf erzeugt

Spezifikationen mittels Linearer Temporallogik (LTL)

- intuitivere Beschreibung der Spezifikation E durch Formel φ_E
 - Prozedur zur Umwandlung φ_E in \mathcal{A}_E
(!) allerdings ist $|\mathcal{A}_E|$ exponentiell in $|\varphi_E|$
 - dafür Explosion bei Komplementierung vermeiden:
wandle $\neg\varphi_E$ in Automaten um
- ↪ beide MCP für LTL sind **PSPACE**-vollständig

LTL im Überblick

LTL = Aussagenlogik + Operatoren, die über **Pfade** sprechen:

F (Future)

$F\varphi$ bedeutet „ φ ist irgendwann in der Zukunft wahr“

G (Global)

$G\varphi$ bedeutet „ φ ist ab jetzt immer wahr“

X (neXt)

$X\varphi$ bedeutet „ φ ist im nächsten Zeitpunkt wahr“

U : (Until)

$\varphi U\psi$ bedeutet „ ψ ist irgendwann in der Zukunft wahr
und bis dahin ist immer φ wahr“

LTL-Syntax

Sei **PROP** abzählbare Menge von **Aussagenvariablen**.

Definition 28 (LTL-Formeln)

- Jede Aussagenvariable $p \in \text{PROP}$ ist eine LTL-Formel.
- Wenn φ und ψ LTL-Formeln sind, dann sind die folgenden auch LTL-Formeln.
 - $\neg\varphi$ „nicht φ “
 - $\varphi \wedge \psi$ „ φ und ψ “
 - $F\varphi$ „in Zukunft irgendwann φ “
 - $G\varphi$ „in Zukunft immer φ “
 - $X\varphi$ „im nächsten Zeitpunkt φ “
 - $\varphi U \psi$ „in Zukunft irgendwann ψ ; bis dahin immer φ “

Verwenden die üblichen Abkürzungen $\varphi \vee \psi = \neg(\neg\varphi \wedge \neg\psi)$,
 $\varphi \rightarrow \psi = \neg\varphi \vee \psi$, $\varphi \leftrightarrow \psi = (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$

Beispiel-Spezifikationen als LTL-Formeln

Beispiel Mikrowelle (siehe Bild auf Folie 72)

- „Wenn ein Fehler auftritt, ist er nach endlicher Zeit behoben.“

$$G(e \rightarrow F\neg e) \quad (e \in \text{PROP} \text{ steht für „Error“})$$
- „Wenn die Mikrowelle gestartet wird, fängt sie nach endlicher Zeit an zu heizen.“

$$G(s \rightarrow Fh) \quad (s, h \in \text{PROP} \text{ stehen für „Start“ bzw. „Heat“})$$
- „Irgendwann ist für genau einen Zeitpunkt die Tür geöffnet.“

$$F(c \wedge X(\neg c \wedge Xc)) \quad (c \in \text{PROP} \text{ steht für „Close“})$$
- „Irgendwann ist für genau einen Zeitpunkt die Tür geöffnet, und bis dahin ist sie geschlossen.“

$$c U (\neg c \wedge Xc)$$

Beispiel-Spezifikationen als LTL-Formeln

Beispiel Nebenläufigkeit

- Es kommt nie vor,
dass beide Teilprog. zugleich im kritischen Bereich sind.
 $G\neg(p_{12} \wedge p_{22})$ ($p_i \in \text{PROP}$ stehen für „Programmzähler in Zeile i “)
- Jedes Teilprog. kommt beliebig oft in seinen krit. Bereich.
 $GFp_{12} \wedge GFp_{22}$

LTL-Semantik

Pfad: Abbildung $s : \mathbb{N} \rightarrow 2^{\text{PROP}}$
 schreiben $s_0s_1s_2 \dots$ statt $s(0)s(1)s(2) \dots$

Definition 29

Sei φ eine LTL-Formel, s ein Pfad und $i \in \mathbb{N}$.

Das **Erfülltsein** von φ in s, i ($s, i \models \varphi$) ist wie folgt definiert.

- $s, i \models p$, falls $p \in s_i$, für alle $p \in \text{PROP}$
- $s, i \models \neg\psi$, falls $s, i \not\models \psi$
- $s, i \models \varphi \wedge \psi$, falls $s, i \models \varphi$ und $s, i \models \psi$
- $s, i \models F\varphi$, falls $s, j \models \varphi$ für ein $j \geq i$
- $s, i \models G\varphi$, falls $s, j \models \varphi$ für alle $j \geq i$
- $s, i \models X\varphi$, falls $s, i+1 \models \varphi$
- $s, i \models \varphi U \psi$, falls $s, j \models \psi$ für ein $j \geq i$
 und $s, k \models \varphi$ für alle k mit $i \leq k < j$

Beispiele

- $s, i \models p$, falls $p \in s_i$, für alle $p \in \text{PROP}$
- $s, i \models \neg\psi$, falls $s, i \not\models \psi$
- $s, i \models \varphi \wedge \psi$, falls $s, i \models \varphi$ und $s, i \models \psi$
- $s, i \models F\varphi$, falls $s, j \models \varphi$ für ein $j \geq i$
- $s, i \models G\varphi$, falls $s, j \models \varphi$ für alle $j \geq i$
- $s, i \models X\varphi$, falls $s, i+1 \models \varphi$
- $s, i \models \varphi U \psi$, falls $s, j \models \psi$ für ein $j \geq i$
und $s, k \models \varphi$ für alle k mit $i \leq k < j$

Siehe Tafel.



Model-Checking mit LTL-Formeln

Zur Erinnerung:

Definition 27: Model-Checking-Problem **MCP**

Gegeben ein System \mathcal{S} und eine Spezifikation E ,

- gilt E für jeden Pfad in \mathcal{S} ?
(**universelle Variante**)
- gibt es einen Pfad in \mathcal{S} , der E erfüllt?
(**existenzielle Variante**)

Model-Checking mit LTL-Formeln

Für LTL:

(jedem Pfad $s_0s_1s_2\dots$ in einer Kripke-Struktur $\mathcal{S} = (S, S_0, R, L)$ entspricht ein LTL-Pfad $s'_0s'_1s'_2\dots$ mit $s'_i = L(s_i)$)

Definition 30 (Model-Checking-Problem)

Gegeben Kripke-Struktur $\mathcal{S} = (S, S_0, R, L)$ und LTL-Formel φ ,

- gilt $s, 0 \models \varphi$ für alle Pfade s , die in einem $s_0 \in S_0$ starten?
(**universelle Variante**)
- gibt es Pfad s , der in einem $s_0 \in S_0$ startet, mit $s, 0 \models \varphi$?
(**existenzielle Variante**)

- ✓ Exakte Beschreibung des Model-Checking-Problems
- ▶ Algorithmische Lösung?

MCP weiterhin mittels Büchi-Automaten lösen!

Vorgehen wie gehabt:

- Wandle Kripke-Struktur \mathcal{S} in NBA \mathcal{A}_S um
 \rightsquigarrow Pfade in \mathcal{S} sind erfolgreiche Runs von \mathcal{A}_S
- Wandeln LTL-Formel φ_E in NBA \mathcal{A}_E um
 \rightsquigarrow \mathcal{A}_E beschreibt Pfade, die E erfüllen

\rightsquigarrow Universelles MCP = „ $L(\mathcal{A}_S) \subseteq L(\mathcal{A}_E)$?“

Existenzielles MCP = „ $L(\mathcal{A}_S) \cap L(\mathcal{A}_E) \neq \emptyset$?“

Frage: Wie wandeln wir φ_E in \mathcal{A}_E um?

Umwandlung von LTL-Formeln in Automaten (Überblick)

- Wandeln φ_E in **verallgemeinerten Büchi-Automaten (GNBA)** um
 - $\mathcal{A}_{\varphi_E} = (Q, \Sigma, \Delta, I, \mathcal{F})$ mit $\mathcal{F} \subseteq 2^Q$
 - $r = q_0q_1q_2 \dots$ ist erfolgreich: $\text{Inf}(r) \cap F \neq \emptyset$ für alle $F \in \mathcal{F}$
 - GNBA und NBA sind äquivalent
- Im Folgenden grobe Vorgehensweise

Vorbetrachtungen

- Genügt, die Operatoren \neg, \wedge, X, U zu betrachten (die anderen kann man mit diesen ausdrücken)
- Sei $\text{cl}(\varphi_E)$ die Menge aller Teilformeln von φ_E und derer Negationen
- $\Sigma = 2^{\text{PROP}}$

Intuitionen

Erweiterung von Pfaden

- Betrachten Pfade $s = s_0s_1s_2 \dots$ mit $s_i \subseteq \text{PROP}$
- Erweitern jedes s_i mit den $\psi \in \text{cl}(\varphi_E)$, für die $s, i \models \psi$ gilt
- Resultat: Folge $\bar{s} = t_0t_1t_2 \dots$ mit $t_i \subseteq \text{cl}(\varphi_E)$

Bestandteile des GNBA \mathcal{A}_{φ_E}

- Zustände: $\approx t_i$
- $\bar{s} = t_0t_1t_2 \dots$ wird ein Run von \mathcal{A}_{φ_E} für $s_0s_1s_2 \dots$ sein
- Run \bar{s} wird erfolgreich sein gdw. $s, 0 \models \varphi_E$
- Kodieren Bedeutung der logischen Operatoren in
 - Zustände (\neg, \wedge , teilweise U)
 - Überführungsrelation (X , teilweise U)
 - Akzeptanzbedingung (teilweise U)

Zustandsmenge des GNBA \mathcal{A}_{φ_E}

$Q =$ Menge aller **elementaren** Formelmengen $t \subseteq \text{cl}(\varphi_E)$:

- ① t ist **konsistent** bzgl. Aussagenlogik, d. h.
für alle $\psi_1 \wedge \psi_2 \in \text{cl}(\varphi_E)$ und $\psi \in \text{cl}(\varphi_E)$:
 - $\psi_1 \wedge \psi_2 \in t$ gdw. $\psi_1 \in t$ und $\psi_2 \in t$
 - wenn $\psi \in t$, dann $\neg\psi \notin t$
- ② t ist **lokal konsistent** bzgl. des U -Operators, d. h.
für alle $\psi_1 U \psi_2 \in \text{cl}(\varphi_E)$:
 - wenn $\psi_2 \in t$, dann $\psi_1 U \psi_2 \in t$
 - wenn $\psi_1 U \psi_2 \in t$ und $\psi_2 \notin t$, dann $\psi_1 \in t$
- ③ t ist **maximal**, d. h. für alle $\psi \in \text{cl}(\varphi_E)$:
wenn $\psi \notin t$, dann $\neg\psi \in t$

Beispiel: $a U (\neg a \wedge b)$, siehe Tafel



Überführungsrelation des GNBA \mathcal{A}_{φ_E}

- Betrachten Tripel (t, s, t')
mit $t, t' \in Q$ (elem. FM) und $s \in \Sigma$ ($\Sigma = 2^{\text{PROP}}$)
- $(t, s, t') \in \Delta$ wenn
 - $s = t \cap \text{PROP}$
 - für alle $X\psi \in \text{cl}(\varphi_E)$: $X\psi \in t$ gdw. $\psi \in t'$
 - für alle $\psi_1 U \psi_2 \in \text{cl}(\varphi_E)$:
 $\psi_1 U \psi_2 \in t$ gdw. $\psi_2 \in t$ oder $(\psi_1 \in t$ und $\psi_1 U \psi_2 \in t')$
 (“Aufschieben” von $\psi_1 U \psi_2$)

Anfangszustände und Akzeptanzkomponente von \mathcal{A}_{φ_E}

- $I = \{t \in Q \mid \varphi_E \in t\}$
- $\mathcal{F} = \{M_{\psi_1 U \psi_2} \mid \psi_1 U \psi_2 \in \text{cl}(\varphi_E)\}$ mit

$$M_{\psi_1 U \psi_2} = \{t \in Q \mid \psi_1 U \psi_2 \notin t \text{ oder } \psi_2 \in t\}$$

Intuition:

Ein $t \in M_{\psi_1 U \psi_2}$ kommt unendlich oft vor

gdw.

$\psi_1 U \psi_2$ wird immer nur höchstens endlich lange “aufgeschoben”

Beispiel: Xa , siehe Tafel

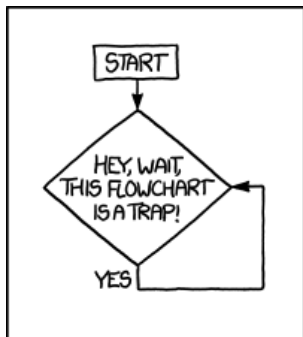


Beispiel: aUb , siehe Übung

Abschließende Betrachtungen

- $|Q|$ ist exponentiell in $|\varphi_E|$
 - Dafür beim universellen MCP auf Komplementierung \mathcal{A}_{φ_E} verzichten:
Wandle $\neg\varphi_E$ in Automaten um
- \rightsquigarrow beide MCP-Varianten in **PSPACE**
- beide MCP-Varianten sind **PSPACE**-vollständig

Damit sind wir am Ende dieses Kapitels.



Quelle: <http://xkcd.com/1195> (Lizenz CC BY-NC 2.5)

Vielen Dank.

Literatur für diesen Teil (1)



Wolfgang Thomas.

Automata on Infinite Objects.

In J. van Leeuwen (Hrsg.):
Handbook of Theoretical Computer Science.
Volume B: Formal Models and Semantics.

Elsevier, 1990, S. 133–192.

SUB, Zentrale: a inf 400 ad/465-2



Wolfgang Thomas.

Languages, automata, and logic.

In G. Rozenberg and A. Salomaa (Hrsg.):
Handbook of Formal Languages. Volume 3: Beyond Words.

Springer, 1997, S. 389–455.

SUB, Zentrale: a inf 330/168-3

Literatur für diesen Teil (2)



Markus Roggenbach.

Determinization of Büchi Automata.

In E. Grädel, W. Thomas, T. Wilke (Hrsg.):
Automata, Logics, and Infinite Games.

LNCS 2500, Springer, 2002, S. 43–60.

Erklärt anschaulich Safra's Konstruktion.

<http://www.cs.tau.ac.il/~rabinoa/LnCS2500.zip>

Auch erhältlich auf Anfrage in der BB Mathematik im MZH:
19h inf 001 k/100-2500



Meghyn Bienvenu.

Automata on Infinite Words and Trees.

Vorlesungsskript, Uni Bremen, WS 2009/10.

Kapitel 2.

[http://www.informatik.uni-bremen.de/tdki/lehre/ws09/
automata/automata-notes.pdf](http://www.informatik.uni-bremen.de/tdki/lehre/ws09/automata/automata-notes.pdf)

Literatur für diesen Teil (3)



Christel Baier, Joost-Pieter Katoen.

Principles of Model Checking.

MIT Press 2008.

Abschnitt 4.3 „Automata on Infinite Words“

Abschnitt 5.2 „Automata-Based LTL Model Checking“

SUB, Zentrale: `a inf 440 ver/782`, `a inf 440 ver/782a`



Edmund M. Clarke, Orna Grumberg, Doron A. Peled.

Model Checking.

MIT Press 1999.

Abschnitt 2 „Modeling Systems“ bis Mitte S. 14,

Abschnitt 2.2.3 + 2.3 „Concurrent Programs“ und „Example ...“,

Abschnitt 3 „Temporal Logics“,

Abschnitt 9.1 „Automata on Finite and Infinite Words“.

SUB, Zentrale: `a inf 440 ver/780(6)`, `a inf 440 ver/780(6)a`