

Automatentheorie und ihre Anwendungen

Teil 3: endliche Automaten auf unendlichen Wörtern

Wintersemester 2018/19

Thomas Schneider

AG Theorie der künstlichen Intelligenz (TdKI)

<http://tinyurl.com/ws1819-autom>

Überblick

- 1 *Motivation*
- 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*
- 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 Leerheitsproblem
(sammle erreichbare 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 und Vorgehen dieses Kapitels

Ziel

Beschreibung von Automatenmodellen mit **unendlichen** Eingaben und **nicht-terminierenden** Berechnungen

Vorgehen

- Theorie: ausgiebiges Studium von Büchi-Automaten und der von ihnen erkannten Sprachen
 - Definition, Abschlusseigenschaften
 - Charakterisierung mittels regulärer Sprachen
 - Determinisierung
 - Entscheidungsprobleme
- Anwendung von Büchi-Automaten:
Spezifikation & Verifikation in Linearer Temporallogik (LTL)

Beispiel: 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.

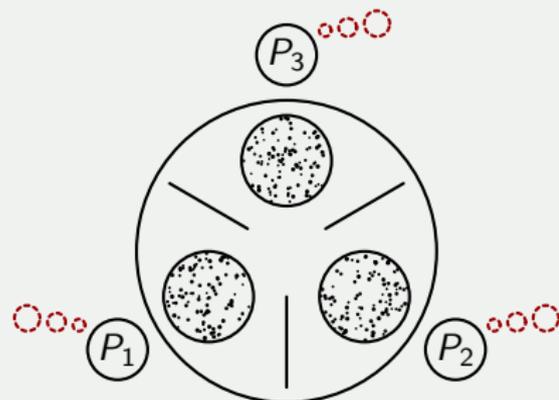
Um zu essen, benötigt P_i beide Stübchen neben seinem Teller.

⇒ Keine zwei P_i, P_j können gleichzeitig essen.

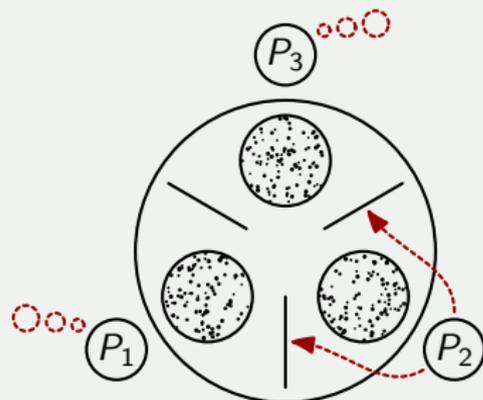
Skizze zum Philosophenproblem

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

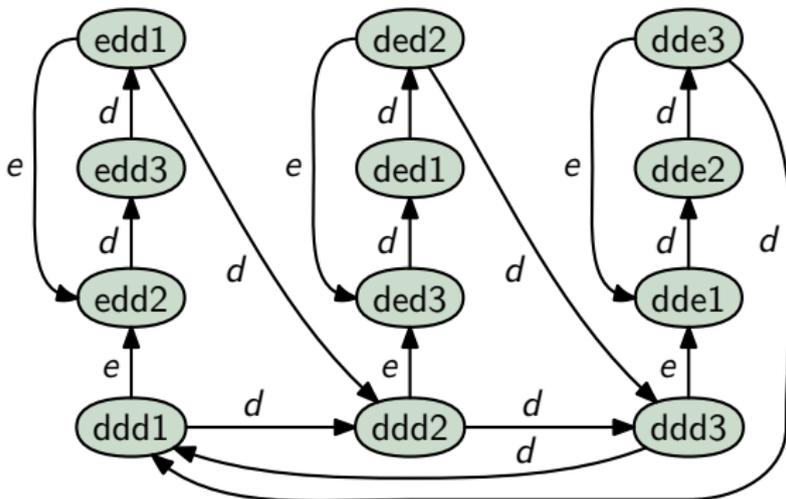
- Anfangszustand ddd1:
alle P_i denken, und P_1 trifft nächste Entscheidung.
- alle zulässigen Zustände:

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.

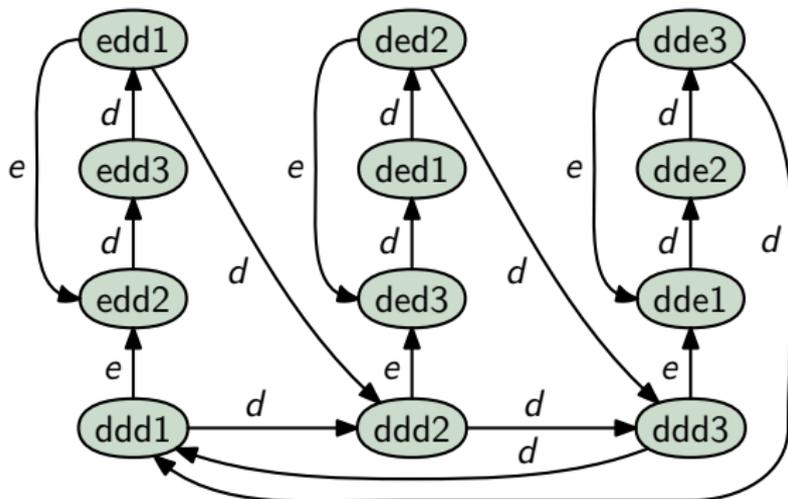
System soll dazu **beliebig lange** ohne Terminierung laufen.

Philosoph P_i heißt **zufrieden**, wenn er währenddessen **unendlich oft** denkt und isst.

↪ **Mögliche Fragen:**

- 1 Kann das System überhaupt beliebig lange laufen?
- 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?

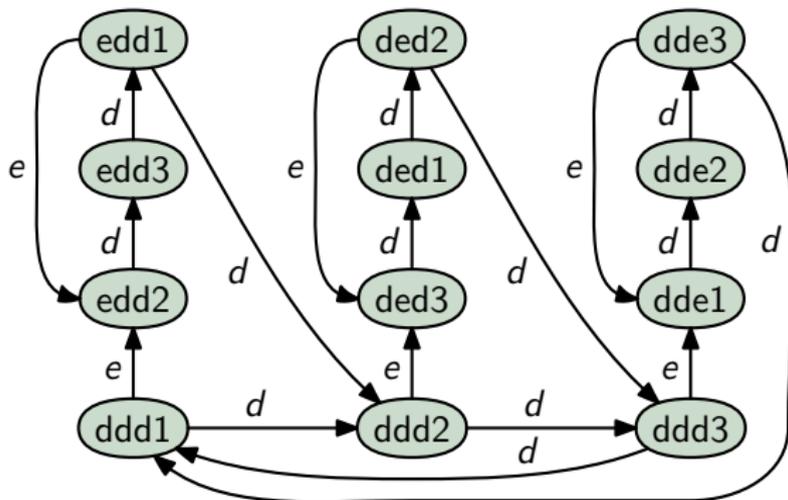
Frage 2



Ist es möglich, dass P_1 zufrieden ist?

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

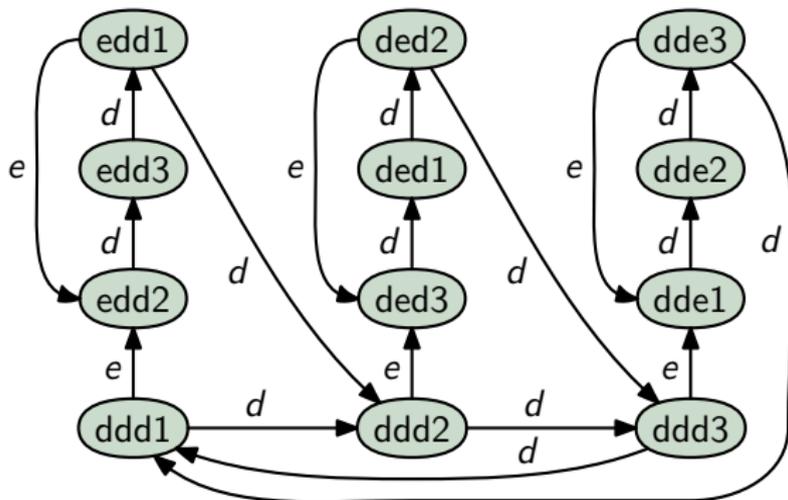
Frage 3



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

Weiteres Beispiel

... siehe Anhang, Folie 122 ...

Und nun ...

- 1 *Motivation*
- 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 $\alpha : \mathbb{N} \rightarrow \Sigma$
- $\alpha(n)$: Symbol an n -ter Stelle (auch: α_n)
- wird oft geschrieben als $\alpha = \alpha_0\alpha_1\alpha_2\dots$

Weitere Notation

- $\alpha[m, n]$: endliche Teilfolge $\alpha_m\alpha_{m+1}\dots\alpha_n$
- $\#_w(\alpha)$: Anzahl der Vorkommen von w als Teilwort in α
 $= \#\{(m, n) \mid \alpha[m, n] = w\}$
- w^ω : unendliche Verkettung von w
 $(\alpha \text{ mit } \alpha[i \cdot n, (i+1)n - 1] = w \text{ f. alle } i \geq 0, n = |w|)$

Σ^ω : Menge aller unendlichen Wörter

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

Büchi-Automaten

Definition 3.1

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 **Überföhrungsrelation** ist,
- $I \subseteq Q$ die Menge der **Anfangszustände** ist,
- $F \subseteq Q$ die Menge der **akzeptierenden Zustände** ist.

Bisher kein Unterschied zu NEAs, aber ...

Berechnungen und Akzeptanz

Definition 3.2

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

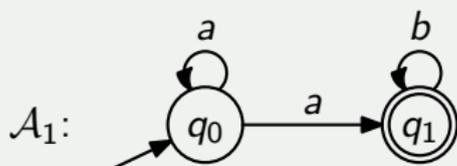
- Ein **Run** von \mathcal{A} auf ω -Wort α ist eine Folge

$$r = q_0 q_1 q_2 \dots,$$

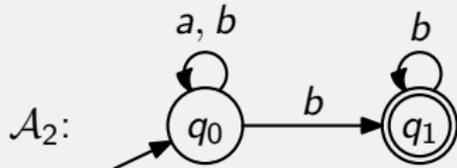
so dass für alle $i \geq 0$ gilt: $(q_i, \alpha_i, q_{i+1}) \in \Delta$.

- **Unendlichkeitsmenge $\text{Inf}(r)$** von $r = q_0 q_1 q_2 \dots$:
Menge der Zustände, die unendlich oft in r vorkommen
- **Erfolgreicher Run** $r = q_0 q_1 q_2 \dots$: $q_0 \in I$ und $\text{Inf}(r) \cap F \neq \emptyset$
- \mathcal{A} **akzeptiert** α ,
wenn es einen erfolgreichen Run von \mathcal{A} auf α gibt.
- Die von \mathcal{A} **erkannte Sprache** ist
 $L_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mathcal{A} \text{ akzeptiert } \alpha\}$.

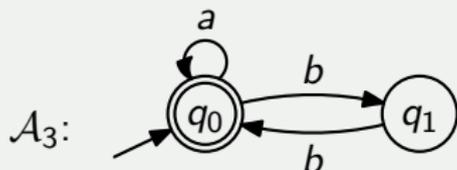
Beispiele



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



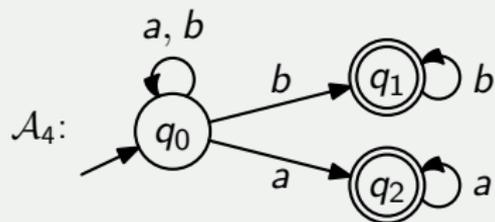
$$L_\omega(\mathcal{A}_2) = \{\alpha \in \Sigma^\omega \mid \#_a(\alpha) < \infty\}$$



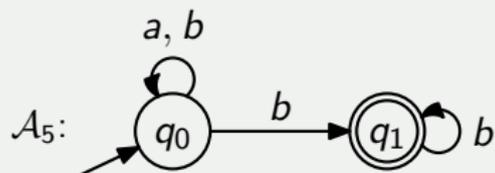
$$L_\omega(\mathcal{A}_3) = \{\alpha \in \Sigma^\omega \mid \text{s. unten}\}$$

Zwischen je zwei a 's in α sowie vor dem ersten a steht jeweils eine gerade Anzahl von b 's.

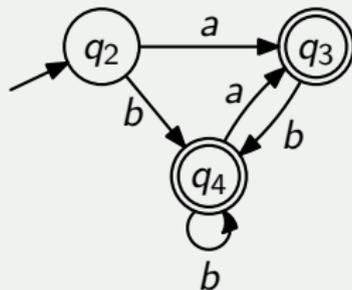
Mehr Beispiele



$$L_\omega(\mathcal{A}_4) = \{\alpha \in \Sigma^\omega \mid \#_a(\alpha) < \infty \\ \text{oder } \#_b(\alpha) < \infty\}$$



$$L_\omega(\mathcal{A}_5) = \{\alpha \in \Sigma^\omega \mid \#_a(\alpha) < \infty \\ \text{oder } \#_{aa}(\alpha) = 0\}$$



(Letzteres heißt:
auf jedes a in α folgt direkt ein b)

Erkennbare Sprache

Definition 3.3

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*
- 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)*

Operationen auf ω -Sprachen

Zur Erinnerung: die Menge der Büchi-erkennbaren Sprachen heißt abgeschlossen unter

- **Vereinigung**, wenn gilt:

Falls L_1, L_2 Büchi-erkennbar, so auch $L_1 \cup L_2$.

- **Schnitt**, 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 sind die Büchi-erkennbaren Sprachen abgeschlossen, und wie leicht ist das zu zeigen?

Vereinigung? ✓ (leicht)

Schnitt? ✓ (mittel)

Komplement? ✓ (schwer)

Abgeschlossenheit

Satz 3.4

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

Direkte Konsequenz aus den folgenden Lemmata.

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

Abgeschlossenheit unter Vereinigung

Lemma 3.5

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. analog zu NEAs und NEBAs:

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.

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

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



Abgeschlossenheit unter Schnitt

Für NEAs: Produktautomat

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

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

- $Q_3 = Q_1 \times Q_2$
- $\Delta_3 = \{((p, p'), a, (q, q')) \mid (p, a, q) \in \Delta_1 \ \& \ (p', a, q') \in \Delta_2\}$
- $I_3 = I_1 \times I_2$
- $F_3 = F_1 \times F_2$ **T 3.1**

Funktioniert das auch für Büchi-Automaten?

Nein. \mathcal{A}_1 und \mathcal{A}_2 besuchen ihre akzeptierenden Zustände
möglicherweise nicht synchron! **T 3.1 Forts.**

Abgeschlossenheit unter Schnitt

Neue Idee für Schnitt-Automat \mathcal{A} :

- \mathcal{A} simuliert $\mathcal{A}_1, \mathcal{A}_2$ nach wie vor parallel, aber mit **2 Modi 1, 2**
 - Modus i bedeutet: warte auf einen akz. Zustand f von \mathcal{A}_i
 - Sobald so ein f erreicht ist, wechsele den Modus.
 - Run von \mathcal{A} ist erfolgreich, wenn er ∞ oft den Modus wechselt.
- ↪ Es werden genau die Wörter akzeptiert,
für die $\mathcal{A}_1, \mathcal{A}_2$ jeweils einen erfolgreichen Run haben.

Absgeschlossenheit unter Schnitt

Lemma 3.6

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

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

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

Konstruieren $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ wie folgt.

$$Q = Q_1 \times Q_2 \times \{1, 2\}$$

$$\begin{aligned} \Delta = & \{((p, p', 1), a, (q, q', 1)) \mid p \notin F_1 \ \& \ (p, a, q) \in \Delta_1 \ \& \ (p', a, q') \in \Delta_2\} \\ & \cup \{((p, p', 1), a, (q, q', 2)) \mid p \in F_1 \ \& \ (p, a, q) \in \Delta_1 \ \& \ (p', a, q') \in \Delta_2\} \\ & \cup \{((p, p', 2), a, (q, q', 2)) \mid p' \notin F_2 \ \& \ (p, a, q) \in \Delta_1 \ \& \ (p', a, q') \in \Delta_2\} \\ & \cup \{((p, p', 2), a, (q, q', 1)) \mid p' \in F_2 \ \& \ (p, a, q) \in \Delta_1 \ \& \ (p', a, q') \in \Delta_2\} \end{aligned}$$

$$I = I_1 \times I_2 \times \{1\}$$

$$F = Q_1 \times F_2 \times \{2\}$$

T 3.2

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

T 3.2 Forts.

□

Abgeschlossenheit unter Komplement

... siehe Abschnitt

„Deterministische Büchi-Automaten und Determinisierung“

Und nun ...

- 1 *Motivation*
- 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 $L \subseteq \Sigma^\omega$.

- $W^\omega = \{w_0 w_1 w_2 \cdots \mid w_i \in W \setminus \{\varepsilon\} \text{ für alle } i \geq 0\}$
(ist ω -Sprache, weil ε ausgeschlossen wurde)
- $WL = \{w\alpha \mid w \in W, \alpha \in L\}$
(ist ω -Sprache)

Von regulären zu Büchi-erkennbaren Sprachen (1)

Lemma 3.7

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

Beweis. (Schritt 1)

Sei \mathcal{A} ein **NEA** mit $L(\mathcal{A}) = W$.

Dann gibt es NEA \mathcal{A}_1 mit $L(\mathcal{A}_1) = W \setminus \{\varepsilon\}$ (Abschlusseig.!).

O. B. d. A. habe $\mathcal{A}_1 \dots$

- ① einen **einzigen** Anfangszustand q_I und
- ② **keine** in q_I **eingehenden** Kanten: keine Transitionen (\cdot, \cdot, q_I)
- ③ und sei $q_I \notin F$.

Diese Form lässt sich durch Hinzufügen eines frischen Anfangszustandes (und der entsprechenden Transitionen) erreichen! (Ü)

Von regulären zu Büchi-erkennbaren Sprachen (1)

Lemma 3.7

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

Beweis. (Schritt 2a)

Sei also $\mathcal{A}_1 = (Q_1, \Sigma, \Delta_1, \{q_I\}, F)$ mit den genannten Eigenschaften und $L(\mathcal{A}_1) = W \setminus \{\varepsilon\}$.

Idee: konstruiere **NBA** \mathcal{A}_2 , der

- \mathcal{A}_1 simuliert, bis ein akzeptierender Zustand erreicht ist und
- dann **nichtdeterministisch entscheidet**, ob die Simulation fortgesetzt wird oder eine neue Simulation von q_0 aus gestartet wird

Von regulären zu Büchi-erkennbaren Sprachen (1)

Lemma 3.7

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

Beweis. (Schritt 2b)

Sei also $\mathcal{A}_1 = (Q_1, \Sigma, \Delta_1, \{q_I\}, F)$ mit den genannten Eigenschaften und $L(\mathcal{A}_1) = W \setminus \{\varepsilon\}$.

Definiere NBA $\mathcal{A}_2 = (Q_1, \Sigma, \Delta_2, \{q_I\}, \{q_I\})$ mit

$$\Delta_2 = \Delta_1 \cup \{(q, a, q_I) \mid (q, a, q_f) \in \Delta_1 \text{ für ein } q_f \in F\}$$

(d. h. alle Kanten, die in \mathcal{A}_1 zu einem akz. Zustand führen, können in \mathcal{A}_2 zusätzlich zu q_I führen

– siehe „nichtdeterministisch entscheidet“ auf voriger Folie!)

Noch zu zeigen: $L_\omega(\mathcal{A}_2) = L(\mathcal{A}_1)^\omega$

T 3.3 \square

Von regulären zu Büchi-erkennbaren Sprachen (2)

Lemma 3.8

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 regulären Sprachen unter
Konkatenation. □

Satz von Büchi

Satz 3.9

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

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

Beweisskizze:

“ \Leftarrow ”: folgt aus Lemmas 3.5, 3.7 und 3.8

“ \Rightarrow ”: bilden V_i, W_i aus denjenigen Wörtern, die zum jeweils nächsten Vorkommen eines akzeptierenden Zustandes führen

Details siehe Tafel.

T 3.4 \square

Konsequenz:

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

$$r_1 s_1^\omega + \dots + r_n s_n^\omega \quad (r_i, s_i \text{ sind reguläre Ausdrücke})$$

Und nun ...

- 1 *Motivation*
- 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
(der Beweis wird aber anspruchsvoll sein)

Definition 3.10

Ein **deterministischer Büchi-Automat (DBA)** ist ein NBA

$\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ mit

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

Zu Hilfe: Charakterisierung der DBA-erkennbaren Sprachen

Sei $W \subseteq \Sigma^*$.

$\vec{W} = \{\alpha \in \Sigma^\omega \mid \alpha[0, n] \in W \text{ für unendlich viele } n\}$
 (d. h. α hat ∞ viele Präfixe in W)

T 3.5

Satz 3.11

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

Beweis. Genügt zu zeigen, dass für jeden **DEA/DBA** $\mathcal{A} = (Q, \Sigma, \Delta, \{q_I\}, F)$ gilt:

$$L_\omega(\mathcal{A}) = \overline{\vec{L(\mathcal{A})}}$$

T 3.6 \square

DBAs sind schwächer als NBAs

Satz 3.12

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

Beweis.

- Betrachte $L = \{\alpha \in \{a, b\}^\omega \mid \#_a(\alpha) \text{ ist endlich}\}$
- L ist Büchi-erkennbar: $L = \Sigma^* \{b\}^\omega$, wende Satz 3.9 an
- Annahme, L sei DBA-erkennbar.
 - \Rightarrow Satz 3.11: $L = \overrightarrow{W}$ 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
 - $\Rightarrow \alpha := b^{n_1} a b^{n_2} a b^{n_3} \dots \in \overrightarrow{W}$ **Widerspruch:** $\alpha \notin L$ \square

Nebenprodukt des letzten Beweises

Die DBA-erkennbaren Sprachen sind **nicht** unter Komplement abgeschlossen:

- $L = \{\alpha \in \{a, b\}^\omega \mid \#_a(\alpha) \text{ ist endlich}\}$
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 akz. Zustände)

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 3.13

Nichtdet. **Muller-Automat (NMA)** 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 $\underbrace{\text{Inf}(r)}_{\text{~~~~~}} \in \mathcal{F}$

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

T 3.7

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

Definition 3.14

Nichtdet. **Rabin-Automat (NRA)** 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 $r = q_0 q_1 q_2 \dots$ 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 **T 3.8**

Streett-Automaten

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

Definition 3.15

Nichtdet. **Streett-Automat (NSA)** 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 $r = q_0 q_1 q_2 \dots$ 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 **T 3.9**

Gleichmächtigkeit der vier Automatenmodelle

Für $X \in \{\text{Muller, Rabin, Streett}\}$ werden analog definiert:

- $L_\omega(\mathcal{A})$ für (nichtdeterministische) X -Automaten
- X -erkennbar

Satz 3.16

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

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

Beweis: Konsequenz aus Lemmas 3.17–3.19. \Downarrow **T 3.10** \square

Von B-, R-, S- zu Muller-Automaten

Lemma 3.17

- 1 Wenn L Büchi-erkennbar, dann auch Muller-erkennbar.
- 2 Wenn L Rabin-erkennbar, dann auch Muller-erkennbar.
- 3 Wenn L Streett-erkennbar, dann auch Muller-erkennbar.

Beweis.

(1) Sei $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ NBA.

Konstruiere NMA $\mathcal{A}' = (Q, \Sigma, \Delta, I, \mathcal{F})$ mit

$$\mathcal{F} = \{Q' \subseteq Q \mid Q' \cap F \neq \emptyset\}.$$

Leicht zu sehen: $L_\omega(\mathcal{A}') = L_\omega(\mathcal{A})$.

Von B-, R-, S- zu Muller-Automaten

Lemma 3.17

- 1 Wenn L Büchi-erkennbar, dann auch Muller-erkennbar.
- 2 Wenn L Rabin-erkennbar, dann auch Muller-erkennbar.
- 3 Wenn L Streett-erkennbar, dann auch Muller-erkennbar.

Beweis.

(2) Sei $\mathcal{A} = (Q, \Sigma, \Delta, I, \mathcal{P})$ NRA.

Konstruiere NMA $\mathcal{A}' = (Q, \Sigma, \Delta, I, \mathcal{F})$ mit

$$\mathcal{F} = \{Q' \subseteq Q \mid \exists i \leq n : Q' \cap E_i = \emptyset \text{ und } Q' \cap F_i \neq \emptyset\}.$$

Leicht zu sehen: $L_\omega(\mathcal{A}') = L_\omega(\mathcal{A})$.

(3) Analog. □

Von Büchi- zu R- und S-Automaten

Lemma 3.18

Wenn L Büchi-erkennbar, dann auch

- 1 Rabin-erkennbar und
- 2 Streett-erkennbar.

Beweis.

(1) Sei $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ NBA.

Konstruiere NRA $\mathcal{A}' = (Q, \Sigma, \Delta, I, \mathcal{P})$ mit

$$\mathcal{P} = \{(\emptyset, F)\}.$$

Leicht zu sehen: $L_\omega(\mathcal{A}') = L_\omega(\mathcal{A})$.

(2) Analog, aber mit $\mathcal{P} = \{(F, Q)\}$. □

Von Muller- zu Büchi-Automaten

Lemma 3.19

Jede Muller-erkennbare Sprache ist Büchi-erkennbar.

Beweis.

- Sei $\mathcal{A} = (Q, \Sigma, \Delta, I, \mathcal{F})$ ein Muller-Automat
- Dann ist $L_\omega(\mathcal{A}) = \bigcup_{F \in \mathcal{F}} L_\omega((Q, \Sigma, \Delta, I, \{F\}))$
- Wegen \cup -Abgeschlossenheit genügt es zu zeigen, dass $L_\omega((Q, \Sigma, \Delta, I, \{F\}))$ Büchi-erkennbar ist
- 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 *alle* diese unendlich oft vorkommen

Von Muller- zu Büchi-Automaten

Sei also $\mathcal{A} = (Q, \Sigma, \Delta, I, \{F\})$ (Muller-Automat)

Konstruieren NBA $\mathcal{A}' = (Q', \Sigma, \Delta', I', F')$ mit

$$\bullet Q' = \underbrace{Q}_{\text{Phase 1}} \cup \underbrace{\{\langle q_f, S \rangle \mid q_f \in F, S \subseteq F\}}_{\text{Phase 2}}$$

Ph. 1: \mathcal{A}' simuliert \mathcal{A} , bis \mathcal{A} **irgendwann** in einem $q_f \in F$ ist

Ph. 2: \mathcal{A}' will nur noch Zustände $\in F$ sehen und **jeden** ∞ oft

- \mathcal{A}' wechselt in $\langle q_f, S \rangle$ mit $S = \{q_f\}$
- S enthält die seit dem letzten Zurücksetzen besuchten $q \in F$
- Wenn $S = F$, wird S auf \emptyset „zurückgesetzt“
- akz. Zustände: ein $\langle q_f, F \rangle$ muss ∞ oft gesehen werden

Von Muller- zu Büchi-Automaten

Sei also $\mathcal{A} = (Q, \Sigma, \Delta, I, \{F\})$ (Muller-Automat)

Konstruieren NBA $\mathcal{A}' = (Q', \Sigma, \Delta', I', F')$ mit

- $Q' = \underbrace{Q}_{\text{Phase 1}} \cup \underbrace{\{\langle q_f, S \rangle \mid q_f \in F, S \subseteq F\}}_{\text{Phase 2}}$
- $\Delta' = \Delta$
 $\cup \{(q, a, \langle q_f, \{q_f\} \rangle) \mid (q, a, q_f) \in \Delta, q_f \in F\}$
 $\cup \{(\langle q, S \rangle, a, \langle q', S \cup \{q'\} \rangle) \mid (q, a, q') \in \Delta, q, q' \in F, S \neq F\}$
 $\cup \{(\langle q, F \rangle, a, \langle q', \{q'\} \rangle) \mid (q, a, q') \in \Delta, q, q' \in F\}$
- $I' = I$
- $F' = \{\langle q_f, F \rangle \mid q_f \in F\}$

Dann gilt: $L_\omega(\mathcal{A}') = L_\omega(\mathcal{A})$.

T 3.11



Abschlusseigenschaften

Direkte Konsequenz aus

- Satz 3.4 (Abschlusseigenschaften der Büchi-erkennbaren Spr.)
- und Satz 3.16 (Gleichmächtigkeit der Automatenmodelle):

Folgerung 3.20

Die Menge der

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

ist abgeschlossen unter den Operationen \cup und \cap .

Zu Komplement-Absgeschlossenheit kommen wir jetzt.

Benötigen zunächst deterministische Varianten von Muller-, Rabin-, Streett-Automaten.

Deterministische Varianten

Deterministische Varianten sind analog zu NBA definiert:

Ein 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 3.16 analoge Aussage:

Satz 3.21

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 (ähnlich wie Lemmas 3.17–3.19).

Überblick der Automatenmodelle

Büchi-Automat (NBA):

- $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ mit $F \subseteq Q$
- Erfolgreicher Run r : $\text{Inf}(r) \cap F \neq \emptyset$

Muller-Automat (NMA):

- $\mathcal{A} = (Q, \Sigma, \Delta, I, \mathcal{F})$ mit $\mathcal{F} \subseteq 2^Q$
- Erfolgreicher Run r : $\text{Inf}(r) \in \mathcal{F}$

Rabin-Automat (NRA):

- $\mathcal{A} = (Q, \Sigma, \Delta, I, \mathcal{P})$ mit $\mathcal{P} \subseteq 2^Q \times 2^Q$
- Erfolg: $\exists(E, F) \in \mathcal{P} : \text{Inf}(r) \cap F \neq \emptyset$ und $\text{Inf}(r) \cap E = \emptyset$

Streett-Automat (NSA):

- $\mathcal{A} = (Q, \Sigma, \Delta, I, \mathcal{P})$ mit $\mathcal{P} \subseteq 2^Q \times 2^Q$
- Erfolg: $\forall(E, F) \in \mathcal{P} : \text{Inf}(r) \cap F \neq \emptyset$ impliziert $\text{Inf}(r) \cap E \neq \emptyset$

Determinisierung von Büchi-Automaten

Erinnerung an Satz 3.12: Es gibt eine Büchi-erkennbare Sprache, die nicht durch einen DBA erkannt wird.

Ziel

Prozedur zur Umwandlung eines gegebenen NBA in einen äquivalenten deterministischen **Rabin**-Automaten

- ↪ wegen Satz 3.21 erhält man daraus auch äquivalente deterministische Muller-/Streett-Automaten
- Resultat geht auf McNaughton zurück
(1965 von Robert McNaughton, Philosoph/Inform., Harvard, Rensselaer)
 - Wir verwenden intuitiveren Beweis von Safra
(1988 von Shmuel Safra, Informatiker, Tel Aviv)

Potenzmengenkonstruktion versagt

Zwei naheliegende Versuche:

- ① NBA \rightsquigarrow DBA mittels Potenzmengenkonstruktion (**PMK**)
muss wegen Satz 3.12 fehlschlagen – Bsp. siehe Tafel **T 3.12**
- ② NBA \rightsquigarrow determ. Muller-(Rabin-/Streett-)Automat via PMK
schlägt auch fehl – mit demselben Gegenbeispiel **T 3.13**

Hauptproblem:

- Potenzautomat simuliert mehrere Runs gleichzeitig
- akzeptierende Zustände (**akzZ**) müssen dabei **nicht synchron** erreicht werden

- **Bad runs:**

Wenn DBA \mathcal{A}^d für α eine ∞ Folge von akzZ findet, dann können diese akzZ von **verschiedenen** Runs des NBA \mathcal{A} auf **Präfixen** von α stammen.

Diese Runs müssen nicht zu einem Run auf α fortsetzbar sein.

Abhilfe: Safra „Tricks“

Ziel

- Wandle NBA $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$
in determ. Rabin-Automaten $\mathcal{A}^d = (Q^d, \Sigma, \Delta^d, I^d, \mathcal{P}^d)$ um
mit $L_\omega(\mathcal{A}) = L_\omega(\mathcal{A}^d)$
- Vermeide “bad runs”: **Safra Tricks**

Vorbetrachtungen

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

Konsequenzen aus Trick 1

- Organisation dieser Mengen von Makrozuständen:
als geordnete Bäume – **Safra-Bäume**
- Trick 1 fügt neue Kinder/Geschwister hinzu
↪ Höhe/Breite des Safra-Baums wächst
- Zum Begrenzen der Höhe/Breite: Trick 2 und 3

Safra's Trick 2

Trick 2:

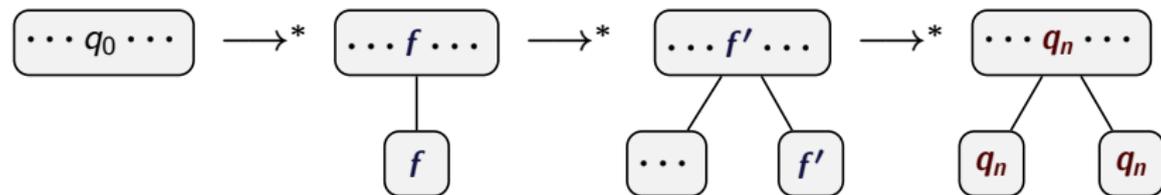
Erkenne zusammenlaufende Teilruns und lösche überflüssige Info

Bsp.: Betrachte Teilruns, die in demselben Zustand q_n enden:

$$r = q_0 q_1 q_2 \dots f \dots q_{n-1} q_n$$

$$r' = q_0 q'_1 q'_2 \dots f' \dots q'_{n-1} q_n \quad (f, f' \in F)$$

Zugehörige n Schritte von \mathcal{A}^d unter Anwendung von Trick 1:



Trick 2 vereinigt die beiden $\{q_n\}$ -Kinder („horizontal merge“)

\rightsquigarrow **Weite** von Safra-Bäumen wird beschränkt

Safra's Trick 3

Trick 3:

Gib überflüssige Makrozustände zur Löschung frei

Wenn alle Kinder eines MZ M bezeugen,
dass *jeder* Zustand in M einen akz. Zustand als Vorgänger hat,
dann können die Kinder gelöscht werden

Genauer: wenn M Kinder M_1, \dots, M_n hat mit $M_1 \cup \dots \cup M_n = M$,
dann werden die M_i gelöscht und M mit $\textcircled{!}$ markiert

\rightsquigarrow „vertical merge“, beschränkt die **Tiefe** von Safra-Bäumen

Definition Safra-Baum

Sei Q Zustandsmenge des ursprünglichen NBA
und V eine nichtleere Menge von **Knotennamen**.

Makrozustand (MZ) über Q : Teilmenge $M \subseteq Q$

Safra-Baum über Q, V :

- geordneter Baum mit Knoten aus V
(der leere Baum ist erlaubt!)
- 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

„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)

↪ Anzahl der möglichen SB ist beschränkt durch $2^{O(|Q| \cdot \log |Q|)}$

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 Knotennamen $V' \subseteq V$; sei $a \in \Sigma$

- ① Beginne mit S ; entferne alle Markierungen $\textcircled{!}$
- 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 , der im MZ eines älteren Geschwisters vorkommt, aus M und aus den MZen der Kinder von v
- 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 $\textcircled{!}$

Illustration der Schritte 2–5

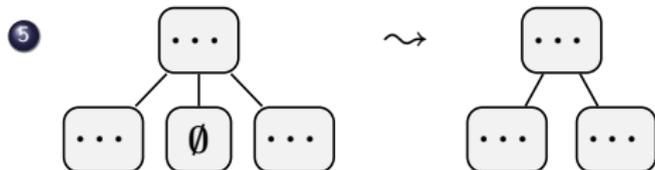
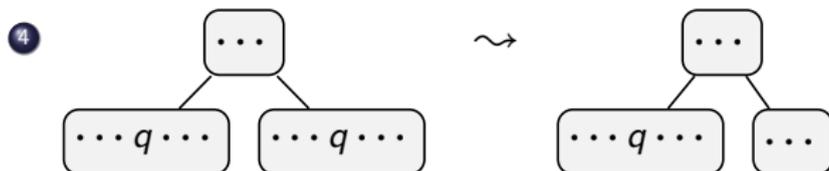
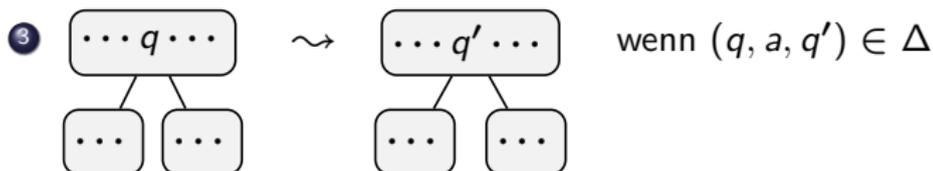
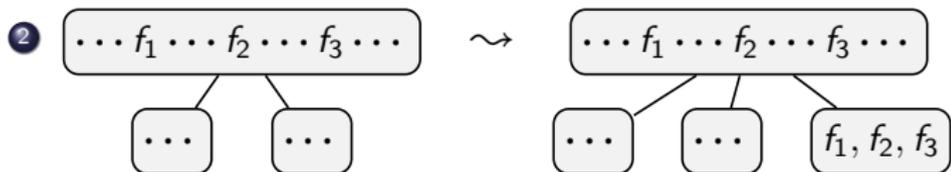
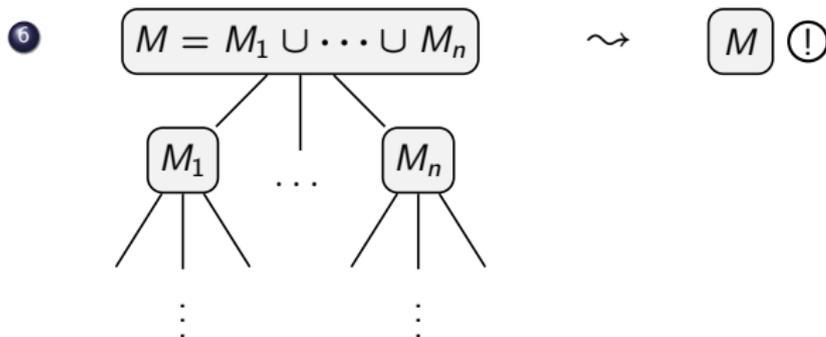


Illustration von Schritt 6



d. h. alle Zustände in M kommen im Makrozustand eines Kindes M_i vor

d. h. jeder Zustand in M hat einen akzZ als Vorgänger!

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

T 3.15

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

T 3.15 Forts.

Korrektheit und Vollständigkeit der Konstruktion

Lemma 3.22

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 $L_\omega(\mathcal{A}^d) = L_\omega(\mathcal{A})$.

Korrektheit:

(Soundness)

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

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

Vollständigkeit:

(Completeness)

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

$$L_\omega(\mathcal{A}^d) \supseteq L_\omega(\mathcal{A})$$

Beweis: Folgerung aus den nächsten beiden Lemmas

Korrektheit

Lemma 3.23

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

Beweisidee. Sei $I = \{q_I\}$ und $I^d = \{S_I\}$. Sei $\alpha \in L_\omega(\mathcal{A}^d)$.

- Betrachte erfolgreichen Run s von \mathcal{A}^d auf α .
- „Konstruiere“ daraus erfolgr. Run von \mathcal{A} auf α *stückweise*:

$$s = S_I \dots T_1 \dots T_2 \dots T_3 \dots, \quad (\text{alle } T_i \text{ laut } \mathcal{P} \text{ gewählt})$$

- Jeder Teilrun $T_i \dots T_{i+1}$ induziert Teilrun von \mathcal{A} auf Teilwort von α , der einen akz. Zustand enthält
- Ordnen diese endl. Teilruns in einem ∞ Baum \mathcal{T} an
- Gesuchter Run von \mathcal{A} ist ein ∞ Pfad in \mathcal{T}

Korrektheit

Beweis. Sei also $\alpha \in L_\omega(\mathcal{A}^d)$.

Dann gibt es erfolgreichen Run $s = S_0 S_1 S_2 \dots$ von \mathcal{A}^d auf α und ein Knoten v , der (wegen \mathcal{P}^d)

- in allen Safra-Bäumen S_j, S_{j+1}, \dots vorkommt, für ein $j \geq 0$, und
- in ∞ vielen Safra-Bäumen mit $\textcircled{!}$ markiert ist.

Seien diese T_1, T_2, \dots und sei $T_0 = S_0$:

$$s = T_0 \dots T_1 \dots T_2 \dots T_3 \dots$$

Zeigen **Hilfsaussage [HA]**:

Für alle T_i und alle Zustände p im MZ von v in T_{i+1} gibt es einen Zustand q im MZ von v in T_i und einen endlichen Run $q \dots p$ von \mathcal{A} auf dem zugehörigen Teilwort von α , der einen akzZ enthält.

Beweis der Hilfsaussage: s. Tafel

T 3.16

Korrektheit

Kombiniere nun Runs aus [HA] zu ∞ Run von \mathcal{A}

- Seien $0 = i_0 < i_1 < i_2 < \dots$ Positionen der T_i in s
- Sei M_j der MZ von v an Positionen i_j , $j \geq 0$

Konstruiere Baum \mathcal{T} :

- Knoten = Paare (q, j) mit $q \in M_j$, $j \geq 0$
- Jeder Knoten $(p, j + 1)$ bekommt *genau ein* Elternteil: beliebiger (q, j) mit $q \in M_j$ und \exists Run $q \dots p$ wie in [HA]

$\Rightarrow \infty$ viele Knoten, Verzweigungsgrad $\leq |Q|$, Wurzel $(q_l, 0)$

Nach Lemma von König (nächste Folie) folgt:

- \mathcal{T} hat einen ∞ Pfad $(q_l, 0), (q_1, 1), (q_2, 2), \dots$;
- Verkettung aller Teilruns entlang dieses Pfades ist ein Run von \mathcal{A} auf α , der ∞ oft einen akzZ besucht

$\Rightarrow \alpha \in L_\omega(\mathcal{A})$



Im Korrektheitsbeweise benutztes Werkzeug

Lemma 3.24 (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)

Vollständigkeit

Lemma 3.25

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.

- Sei $\alpha \in L_\omega(\mathcal{A})$ und $r = q_0q_1q_2 \dots$ erfolgr. Run von \mathcal{A} auf α
- \mathcal{A}^d hat *eindeutigen* Run $s = S_0S_1S_2 \dots$ auf α
- Zu zeigen: s ist erfolgreich, d. h.:

Es gibt einen Knotennamen v , für den gilt:

- $\exists m \geq 0 : S_i$ enthält Knoten v für alle $i \geq m$
- v ist in ∞ vielen S_i mit $\textcircled{!}$ markiert

Beweis dieser Aussage: s. Tafel

T 3.17 \square

Konsequenz aus Safras Konstruktion

Satz 3.26 (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 Lemma 3.22.

Folgerung 3.27

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

Beweis. Über folgende Transformationskette:

NBA für L \rightarrow DRA für L (gemäß Satz 3.26)
 \rightarrow DMA für L (gemäß Satz 3.21)
 \rightarrow DMA für \bar{L} (wie gehabt)
 \rightarrow NBA für \bar{L} (gemäß Satz 3.16) \square

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 mögliche Makrozustände
 - und $2^{O(n \log n)}$ mögliche Safrabäume \rightsquigarrow DRA hat maximal $m := 2^{O(n \log n)}$ Zustände
- Das ist optimal (siehe Roggenbachs Kapitel in LNCS 2500)

Komplementierung beinhaltet auch den Schritt DMA \rightarrow NBA

- liefert einen nochmal **exponentiell** größeren DBA:
 - 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*
- 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 80)
↪ höhere Komplexität

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

Das Leerheitsproblem

Zur Erinnerung:

Gegeben: NBA \mathcal{A}

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

Satz 3.28

Das Leerheitsproblem für NBAs ist entscheidbar.

Quiz: Welche Komplexität hat es? **NL** ... **P** ... höher?

Beweis. $L_\omega(\mathcal{A}) \neq \emptyset$ genau dann, wenn gilt:

Es gibt $q_0 \in I$ und $q_f \in F$

und einen Pfad von q_0 zu q_f in \mathcal{A}

und einen Pfad von q_f zu q_f in \mathcal{A}

\Rightarrow Reduktion zum Leerheitsproblem für NEAs:

Das Leerheitsproblem

Bezeichne $L(\mathcal{A}_{q_1, q_2})$ die von \mathcal{A} als **NEA** erkannte Sprache, wenn $\{q_1\}$ Anfangs- und $\{q_2\}$ Endzustandsmenge ist

Folgender Algorithmus entscheidet das Leerheitsproblem:

```

Rate nichtdeterministisch  $q_0 \in I$  und  $q_F \in F$ 
if  $L(\mathcal{A}_{q_0, q_f}) \subseteq \{\varepsilon\}$  oder  $L(\mathcal{A}_{q_f, q_f}) \subseteq \{\varepsilon\}$  then return „leer“
return „nicht leer“
  
```

Dabei ist $L(\mathcal{A}_{...}) \subseteq \{\varepsilon\}$ gdw. $L(\mathcal{A}_{...}) \cap \underbrace{(\Sigma \setminus \{\varepsilon\})}_{\text{konst. NEA}} = \emptyset$

(„ $L(\mathcal{A}_{...}) = \emptyset$ “ genügt nicht, denn $L_\omega(\rightarrow \odot) = \emptyset$.) □

Das ist ein **NL**-Algorithmus

(eigentlich **coNL**, aber **NL** = **coNL** ist bekannt, Immerman–Szelepcsényi 1987)

Leerheit für NBAs ist **NL**-vollständig

Überblick Entscheidungsprobleme für NBAs

Problem	entscheidbar?	Komplexität	effizient lösbar?
LP	✓	NL -vollständig	✓
WP	— macht keinen Sinn, da Eingabewort ∞ —		
ÄP	✓	PSpace -vollst.	✗*
UP	✓	PSpace -vollst.	✗*

* unter den üblichen komplexitätstheoretischen Annahmen
(z. B. **PSpace** \neq **P**)

Und nun ...

- 1 *Motivation*
- 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 3.29

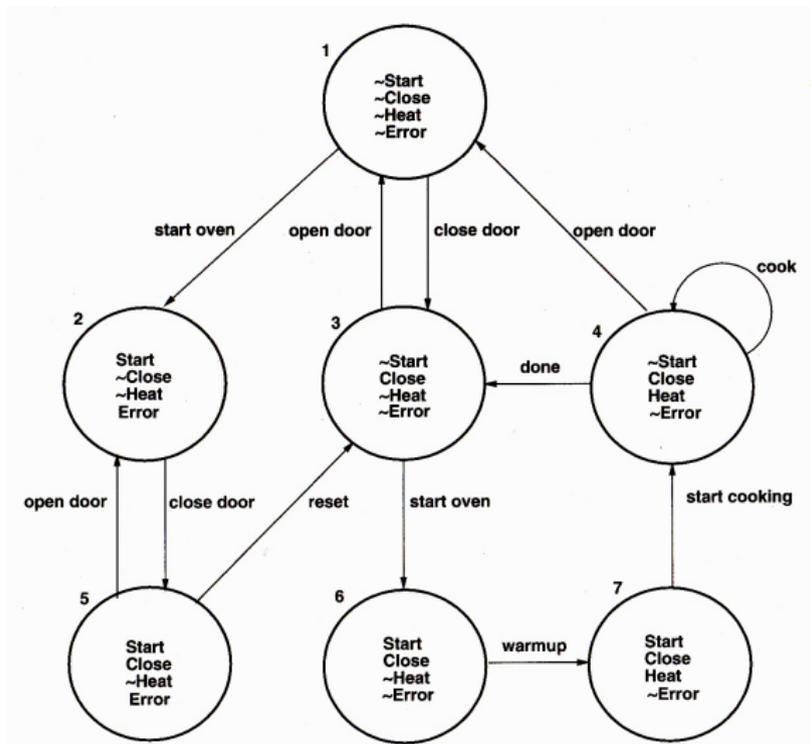
Sei AV eine Menge von Aussagenvariablen. Eine **Kripke-Struktur** \mathcal{S} über AV ist ein Quadrupel $\mathcal{S} = (S, S_0, R, \ell)$, wobei

- 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**: $\forall s \in S \exists s' \in S : sRs'$
- $\ell : S \rightarrow 2^{AV}$ eine Funktion ist, die **Markierungsfunktion**.
 $\ell(s) = \{p_1, \dots, p_m\}$ bedeutet: in s sind genau p_1, \dots, p_m wahr

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

* Saul Kripke, geb. 1940, Philosoph und Logiker, Princeton und New York, USA

Beispiel 1: Mikrowelle



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

Beispiel 2: nebenläufiges Programm

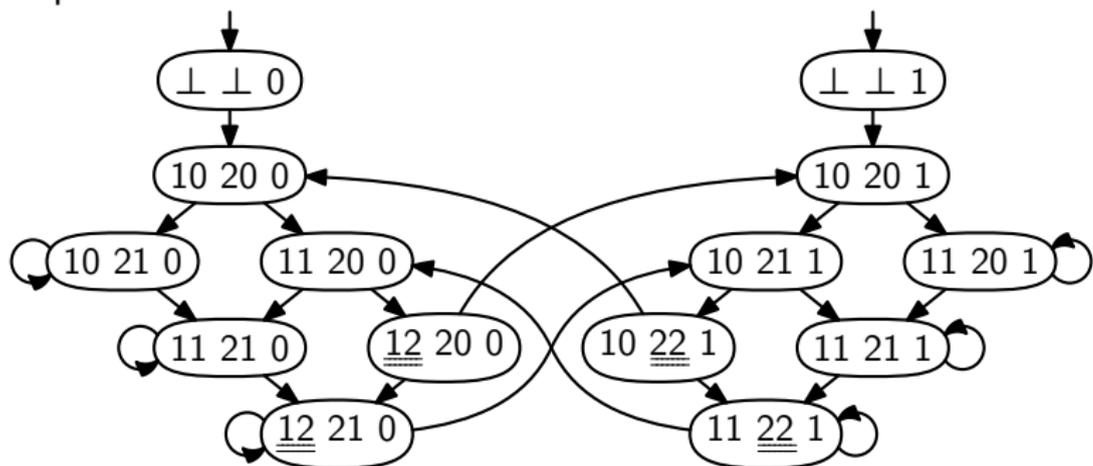
```
P    0  cobegin  
      1     $P_0 \parallel P_1$   
      2  coend  
  
P0 10  while(true) do  
      11    wait(turn = 0)  
      12    turn ← 1      kritischer Bereich  
      13  end while  
  
P1 20  while(true) do  
      21    wait(turn = 1)  
      22    turn ← 0      kritischer Bereich 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

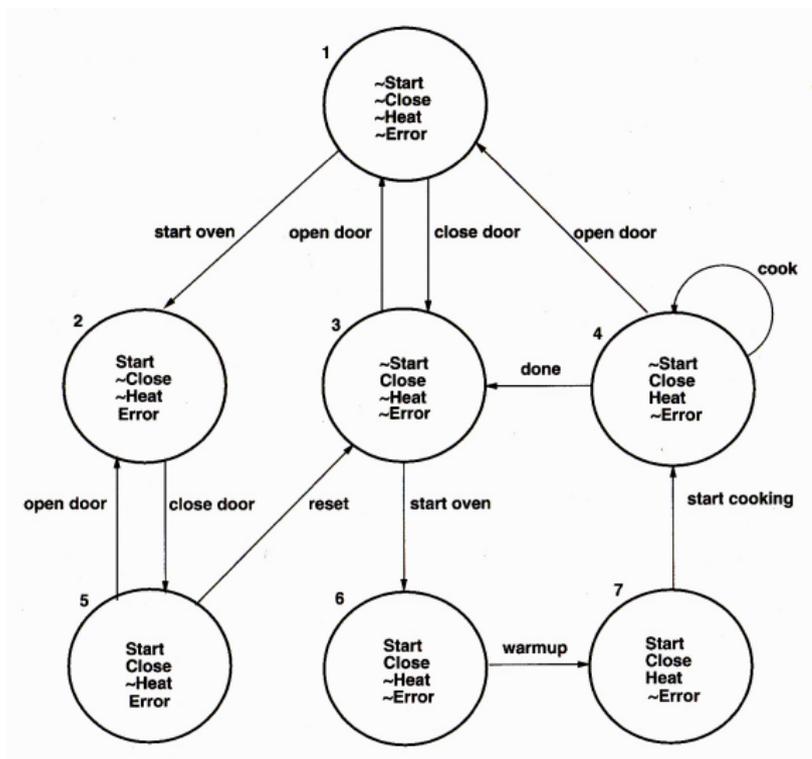


Spezifikationen

... 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

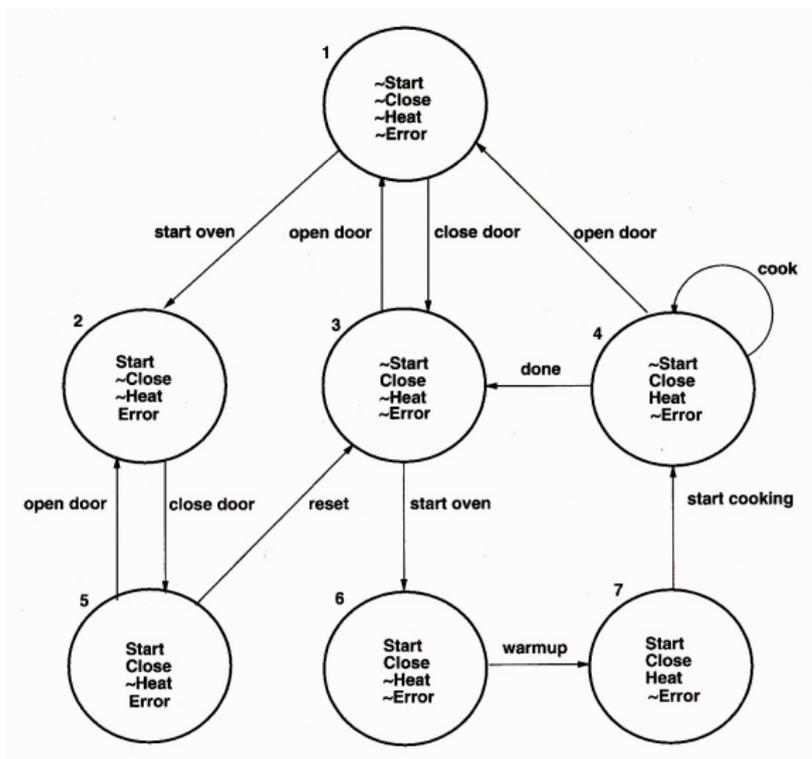


aus:

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

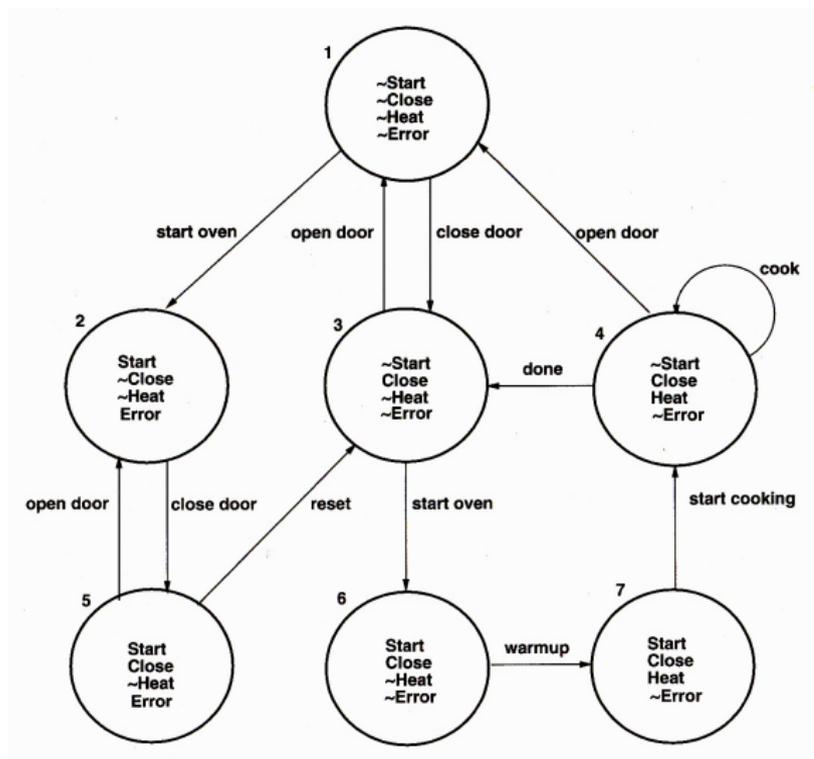


aus:

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

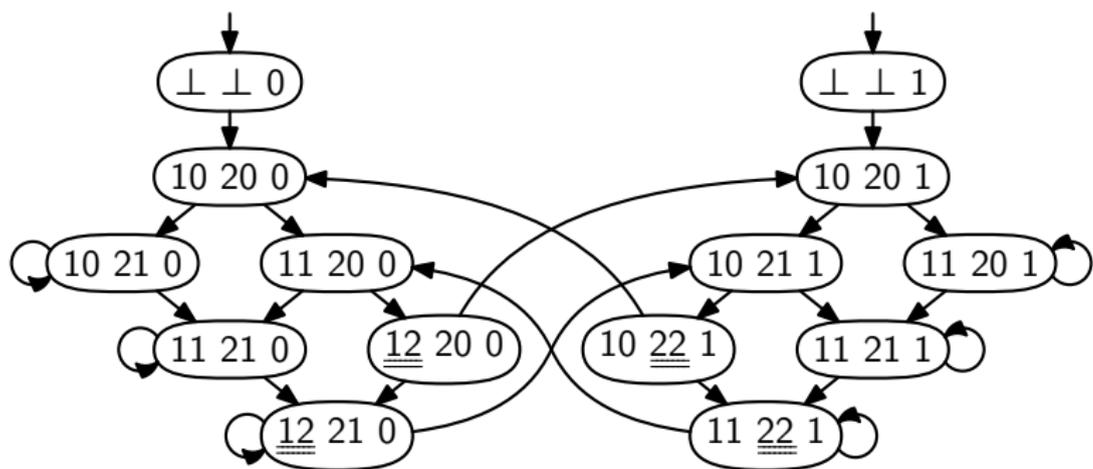


aus:

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

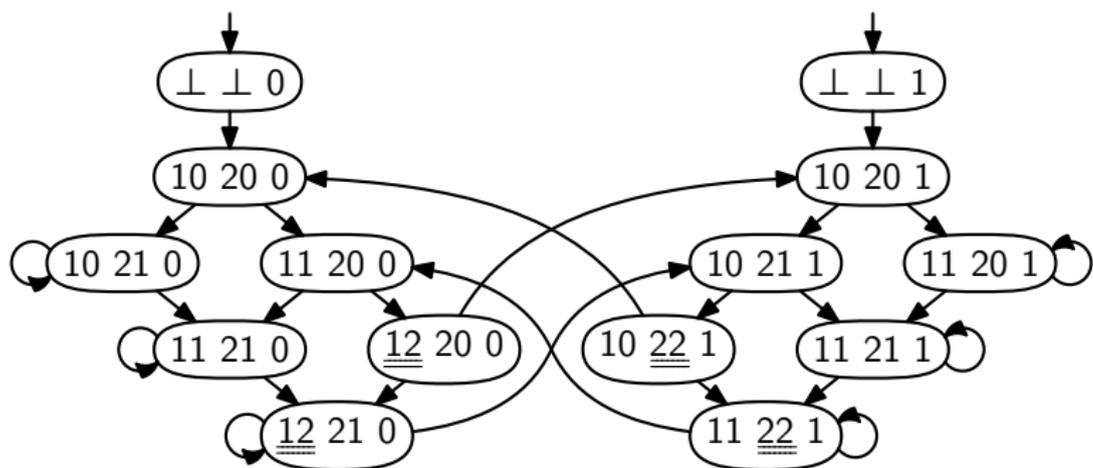
„Wenn MW gestartet, ist es *möglich*, danach zu heizen.“ ✓

Spezifikationen für das Beispiel Nebenläufigkeit



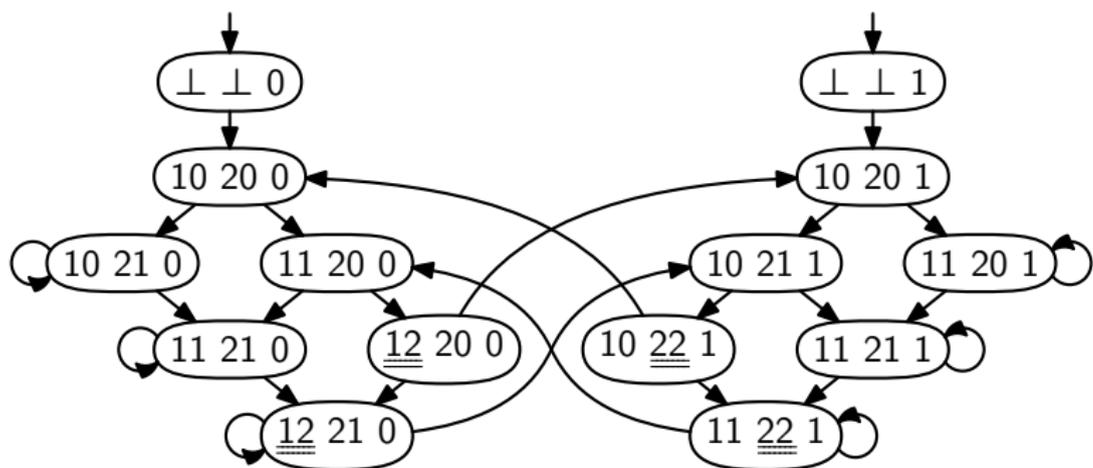
„Es kommt nie vor,
dass beide Teilprogramme zugleich im kritischen Bereich sind.“ ✓

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 3.30 (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)

Frage: Wie kann man Model-Checking

- exakt beschreiben und
- algorithmisch lösen?

Model-Checking mittels Büchi-Automaten!

Schritt 1

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

(beide reduzierbar zum Leerheitsproblem, benutzt Abschlusseigenschaften)

Schritt 2

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

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

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

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

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

Beispiel: siehe Tafel.

T 3.18

Beschreibung von E durch NBA \mathcal{A}_E

Beispiel Mikrowelle (siehe Bild auf Folie 90)

- (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 (siehe Bild auf Folie 92)

- (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.“

T 3.19

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** (exponentielle 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)

Bemerkung zur Implementierung

Praktisches Problem

- Komplexität von MCP wird **bezüglich** $|\mathcal{A}_S| + |\mathcal{A}_E|$ gemessen
 - $|\mathcal{S}|$ und damit $|\mathcal{A}_S|$ ist **exponentiell** in der Anzahl der Variablen:
State space explosion problem
- ↪ universelles bzw. existenzielles MCP sind eigentlich in **ExpSpace** bzw. in **PSpace** bezüglich Anz. der Variablen

Abhilfe:

- „On-the-fly model checking“
- Zustände von \mathcal{A}_S werden während des Leerheitstests nur bei Bedarf erzeugt

Spezifikationen mittels Linearer Temporallogik (LTL)

Nun zu Schritt 2. Ziele:

- 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 **AV** abzählbare Menge von **Aussagenvariablen**.

Definition 3.31 (LTL-Formeln)

- Jede Aussagenvariable $p \in AV$ 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 $\varphi \vee \psi = \neg(\neg\varphi \wedge \neg\psi)$,
 $\varphi \rightarrow \psi = \neg\varphi \vee \psi$, $\varphi \leftrightarrow \psi = (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$

LTL-Semantik

Pfad: Abbildung $\pi : \mathbb{N} \rightarrow 2^{AV}$ Schreiben $\pi_0\pi_1 \dots$ statt $\pi(0)\pi(1) \dots$

Definition 3.32

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

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

- $\pi, i \models p$, falls $p \in \pi_i$, für alle $p \in AV$
- $\pi, i \models \neg\psi$, falls $\pi, i \not\models \psi$
- $\pi, i \models \varphi \wedge \psi$, falls $\pi, i \models \varphi$ und $\pi, i \models \psi$
- $\pi, i \models F\varphi$, falls $\pi, j \models \varphi$ für ein $j \geq i$
- $\pi, i \models G\varphi$, falls $\pi, j \models \varphi$ für alle $j \geq i$
- $\pi, i \models X\varphi$, falls $\pi, i+1 \models \varphi$
- $\pi, i \models \varphi U \psi$, falls $\pi, j \models \psi$ für ein $j \geq i$
und $\pi, k \models \varphi$ für alle k mit $i \leq k < j$

T 3.20

Beispiel-Spezifikationen als LTL-Formeln

Beispiel Mikrowelle (siehe Bild auf Folie 90)

- „Wenn ein Fehler auftritt, ist er nach endlicher Zeit behoben.“
 $G(e \rightarrow F\neg e)$ ($e \in AV$ steht für „Error“)
- „Wenn die Mikrowelle gestartet wird, fängt sie nach endlicher Zeit an zu heizen.“
 $G(s \rightarrow Fh)$ ($s, h \in AV$ stehen für „Start“ bzw. „Heat“)
- „Irgendwann ist für genau einen Zeitpunkt die Tür geöffnet.“
 $F(c \wedge X(\neg c \wedge Xc))$ ($c \in AV$ 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 (siehe Bild auf Folie 92)

- Es kommt nie vor,
dass beide Teilprog. zugleich im kritischen Bereich sind.

$$G\neg(p_{12} \wedge p_{22}) \quad (p_i \in AV \text{ stehen für „Programnzähler in Zeile } i\text{“})$$

- Jedes Teilprog. kommt beliebig oft in seinen krit. Bereich.

$$GFp_{12} \wedge GFp_{22}$$

Model-Checking mit LTL-Formeln

Zur Erinnerung:

Definition 3.30: 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**)

Model-Checking mit LTL-Formeln

Für LTL:

(jedem Pfad $s_0s_1s_2\dots$ in einer Kripke-Struktur $\mathcal{S} = (S, S_0, R, \ell)$ entspricht ein LTL-Pfad $\pi_0\pi_1\pi_2\dots$ mit $\pi_i = \ell(s_i)$)

Definition 3.33 (Model-Checking-Problem)

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

- gilt $\pi, 0 \models \varphi$ für alle Pfade π , die in einem $s_0 \in S_0$ starten?
(**universelle Variante**)
- gibt es Pfad π , der in einem $\pi_0 \in S_0$ startet, mit $\pi, 0 \models \varphi$?
(**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 = „ $L(\mathcal{A}_S) \cap L(\mathcal{A}_E) \neq \emptyset$?“

Noch zu klären: Wie wandeln wir φ_E in \mathcal{A}_E um?

Umwandlung von LTL-Formeln in Automaten (Überblick)

Wandeln φ_E in **generalisierten 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 (nur quadratische Vergrößerung)

Vorbetrachtungen

Sei φ_E eine LTL-Formel, in der o. B. d. A.

- nur die Operatoren \neg, \wedge, X, U vorkommen

Die anderen kann man mit diesen ausdrücken:

$$F\varphi \equiv (\neg(p \wedge \neg p)) U \varphi \quad G\varphi \equiv \neg F\neg\varphi$$

- keine doppelte Negation vorkommt

natürlich gilt $\neg\neg\psi \equiv \psi$ für alle Teilformeln ψ

(Hier steht $\alpha \equiv \beta$ für $\forall \pi \forall i : \pi, i \models \alpha$ gdw. $\pi, i \models \beta$)

Etwas Notation

- $\sim\psi = \begin{cases} \vartheta & \text{falls } \psi = \neg\vartheta \\ \neg\psi & \text{sonst} \end{cases}$

- $\text{cl}(\varphi_E) = \{\psi, \sim\psi \mid \psi \text{ ist Teilformel von } \varphi_E\}$

- $\Sigma = 2^{\text{AV}}$

Intuitionen

Erweiterung von Pfaden

- Betrachten Pfade $\pi = s_0 s_1 s_2 \dots$ mit $s_i \subseteq AV$
- Erweitern jedes s_i mit den $\psi \in \text{cl}(\varphi_E)$, für die $\pi, i \models \psi$ gilt
- Resultat: Folge $\bar{\pi} = t_0 t_1 t_2 \dots$ mit $t_i \subseteq \text{cl}(\varphi_E)$

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

 Skizze: s. Tafel T 3.21

- Zustände: \approx alle t_i
- $\bar{\pi} = t_0 t_1 t_2 \dots$ wird ein Run von \mathcal{A}_{φ_E} für $s_0 s_1 s_2 \dots$ sein
- Run $\bar{\pi}$ wird erfolgreich sein gdw. $\pi, 0 \models \varphi_E$
- 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,
wobei $t \subseteq \text{cl}(\varphi_E)$ **elementar** ist, wenn gilt:

- 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 $\sim\psi \notin t$
- 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 $\sim\psi \in t$

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

T 3.22

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

Seien $t, t' \in Q$ (elementare Formelmengen) und $s \in \Sigma$ ($\Sigma = 2^{AV}$)

Δ besteht aus allen Tripeln (t, s, t') mit

- ① $s = t \cap AV$ (d. h. s besteht aus allen Aussagevariablen in t)
- ② für alle $X\psi \in \text{cl}(\varphi_E)$: $X\psi \in t$ gdw. $\psi \in t'$
- ③ 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 $\psi_1 U \psi_2$) 

Skizzen: siehe Tafel

T 3.23

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

Menge der Anfangszustände

alle elementaren Formelmengen, die φ_E enthalten

$$I = \{t \in Q \mid \varphi_E \in t\}$$

Menge der akzeptierenden Zustände

stellen sicher, dass kein $\psi_1 U \psi_2$ für immer “aufgeschoben” wird

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

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

Intuition: Ein $t \in M_{\psi_1 U \psi_2}$ kommt unendlich oft vor gdw.
 $\psi_1 U \psi_2$ immer nur höchstens endlich lange “aufgeschoben” wird

Beispiel: Xa , siehe Tafel

T 3.24

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

T 3.25

Abschließende Betrachtungen

- $|Q|$ ist exponentiell in $|\varphi_E|$
- Dafür kann man jetzt beim universellen MCP auf Komplementierung \mathcal{A}_{φ_E} verzichten:
Wandle $\neg\varphi_E$ in Automaten um

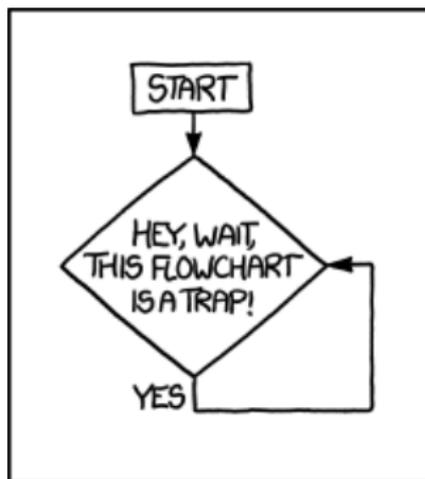
↪ beide MCP-Varianten in **PSpace**

- beide MCP-Varianten sind **PSpace**-vollständig
(aber für bestimmte LTL-Fragmente **NP**- oder **NL**-vollständig)

A. Prasad Sistla, Edmund M. Clarke: *The Complexity of Propositional Linear Temporal Logics*. Journal of the ACM 32(3): 733-749 (1985)

Michael Bauland, Martin Mundhenk, Thomas Schneider, Henning Schnoor, Ilka Schnoor, Heribert Vollmer: *The Tractability of Model Checking for LTL: the Good, the Bad, and the Ugly Fragments*. ACM Trans. Comput. Log. 12(2): 13 (2011)

Damit sind wir am Ende dieses Kapitels.



<http://xkcd.com/1195> (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“.

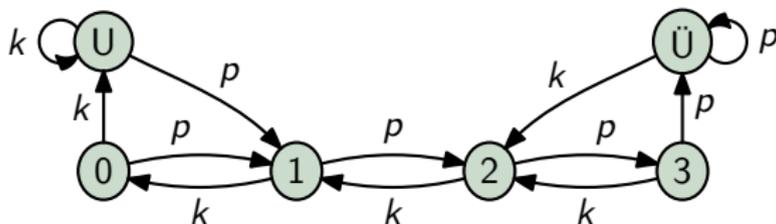
SUB, Zentrale: `a inf 440 ver/780(6)`, `a inf 440 ver/780(6)a`

- 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 $0, 1, 2, 3, \ddot{U}, U$
 - $0,1,2,3$: im Lager liegen $0,1,2,3$ Stück
 - \ddot{U} berschuss: P will ein Stück im vollen Lager ablegen
 - U nterversorgung: 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 $\Sigma = \{p, k\}$
(Läufe)

Zufriedenheit: P (K) möchte ...

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

Lauf, der P und K zufrieden stellt:

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

Lauf, der weder P noch K zufrieden stellt: $p^4 k^4 p^4 k^4 \dots$