

# Automatentheorie und ihre Anwendungen

## Teil 4: endliche Automaten auf unendlichen Bäumen

Wintersemester 2018/19

Thomas Schneider

AG Theorie der künstlichen Intelligenz (TdKI)


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

# Überblick

## Computation Tree Logic (CTL)

- Grenzen von LTL: kann nicht über Pfade quantifizieren
- Berechnungsbäume und CTL
- Ausdrucksvermögen von LTL und CTL im Vergleich
- Model-Checking mit CTL

## Büchi-Automaten auf unendlichen Bäumen

- Definitionen und Beispiele
- äquivalente Automatenmodelle: Muller-, Paritätsautomaten
- Abschlusseigenschaften;  
Komplementierung von Muller-Automaten 

# Überblick

- 1 *Model-Checking mit CTL*
- 2 Automaten auf unendlichen Bäumen
- 3 Komplementierung

# Und nun ...

- 1 *Model-Checking mit CTL*
- 2 Automaten auf unendlichen Bäumen
- 3 Komplementierung

# Erinnerung an LTL

(Linear Temporal Logic)

- System gegeben als Kripke-Struktur  $\mathcal{S} = (S, S_0, R, \ell)$
- LTL-Formel  $\varphi_E$  beschreibt Pfade, die Eigenschaft  $E$  erfüllen
- Beispiel:  
„Wenn Fehler auftritt, ist er nach endlicher Zeit behoben.“  
 $G(e \rightarrow F\neg e)$  ( $e \in AV$  steht für „Error“)
- Umwandlung  $\varphi_E$  in GNBA  $\mathcal{A}_E$ , der zulässige Pfade beschreibt
- lösen damit Model-Checking-Problem:
  - Gilt  $E$  für *alle* Pfade ab  $S_0$  in  $\mathcal{S}$  ?  
(**universelle Variante**)
  - Gilt  $E$  für *mindestens einen* Pfad ab  $S_0$  in  $\mathcal{S}$  ?  
(**existenzielle Variante**)

LTL 1977 eingeführt durch Amir Pnueli, 1941-2009,  
israelischer Informatiker (Haifa, Weizmann-Inst., Stanford, Tel Aviv, New York)

# Grenzen von LTL

„LTL-Formel  $\varphi_E$  beschreibt Pfade, die Eigenschaft  $E$  erfüllen“

**Nicht ausdrückbar:** zu jedem Zeitpunkt ist es immer *möglich*, die Berechnung auf eine gewisse Weise fortzusetzen

**Beispiel:** „Wenn ein Fehler auftritt, ist es *möglich*, ihn nach endlicher Zeit zu beheben.“

$G(e \rightarrow F\neg e)$  oder  $GF\neg e$  sind

- zu stark in Verbindung mit universellem Model-Checking **T 4.1**
- zu schwach in Verbindung mit existenziellem MC **T 4.1 Forts.**

# Ein Fall für CTL

(Computation Tree Logic)

**Abhilfe:** Betrachten Berechnungsbäume statt Pfade

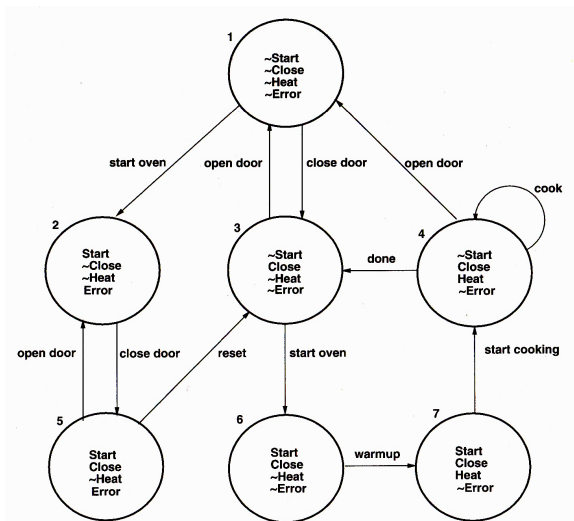
Sei also  $\mathcal{S} = (S, S_0, R, \ell)$  eine Kripke-Struktur

**Berechnungsbaum** für  $s_0 \in S_0$

- entsteht durch „Auffalten“ von  $\mathcal{S}$  in  $s_0$
- enthält *alle unendlichen* Pfade, die in  $s_0$  starten  
d. h.: jeder Zustand  $s \in S$  hat als Kinder  
alle seine Nachfolgerzustände aus  $\mathcal{S}$

$\mathcal{S}$  ist eine endliche Repräsentation aller  $\infty$  Berechnungsbäume

# Beispielstruktur Mikrowelle



T 4.2



# CTL intuitiv

CTL enthält **Pfadquantoren A, E**:

Operatoren, die über **alle** oder **einige** Berechnungen sprechen,  
die in einem bestimmten Zustand beginnen

**Beispiel:**  $AGEF\neg e$

Für alle Berechnungen, die hier starten ( $A$ ),  
gibt es zu jedem Zeitpunkt in der Zukunft ( $G$ )  
eine Möglichkeit, die Berechnung fortzusetzen ( $E$ ),  
so dass irgendwann in der Zukunft ( $F$ )  
kein Fehler auftritt ( $\neg e$ )

CTL 1981 eingeführt durch

Edmund M. Clarke, \*1945, Informatiker, Carnegie Mellon Univ. (Pittsburgh)

E. Allen Emerson, \*1954, Informatiker, Univ. of Texas, Austin, USA

(beide Turing-Award-Träger 2007)

# CTL exakt

Trennung von Zustands- und Pfadformeln:

**Zustandsformeln** drücken Eigenschaften eines Zustandes aus

$$\zeta ::= p \mid \zeta_1 \wedge \zeta_2 \mid \zeta_1 \vee \zeta_2 \mid \neg \zeta \mid E\psi \mid A\psi$$

( $p$ : Aussagenvariable;  $\zeta, \zeta_1, \zeta_2$ : Zustandsformeln;  $\psi$ : Pfadformel)

**Pfadformeln** drücken Eigenschaften eines Pfades aus

$$\psi ::= F\zeta \mid G\zeta \mid X\zeta \mid \zeta_1 U \zeta_2$$

( $\zeta, \zeta_1, \zeta_2$ : Zustandsformeln)

$\rightsquigarrow$  in **zulässigen** CTL-Formeln muss

- jeder Pfadquantor von einem temporalen Operator gefolgt werden
- jeder temporale Operator direkt einem Pfadquantor folgen

# Quiz: zulässige Formeln

Zur Erinnerung:

$$(ZF) \quad \zeta ::= p \mid \zeta_1 \wedge \zeta_2 \mid \zeta_1 \vee \zeta_2 \mid \neg \zeta \mid E\psi \mid A\psi$$

$$(PF) \quad \psi ::= F\zeta \mid G\zeta \mid X\zeta \mid \zeta_1 U \zeta_2$$

**Quizfrage:** Welche der folgenden Formeln sind zulässig?

$$p \wedge q \quad EFp \quad AXp \quad \checkmark$$

$$E(p U q) \quad \checkmark$$

$$A((p \vee \neg p) U q) \quad \checkmark \quad (\text{äquivalent zu } AFq)$$

$$E(p \vee AXq) \quad \times \quad (E \text{ nicht gefolgt von } F, G, X, U)$$

$$EX(p \vee AXq) \quad \checkmark$$

$$EF(p U q) \quad \times \quad (U \text{ folgt nicht } E \text{ oder } A)$$

$$EFA(p U q) \quad \checkmark$$

# CTL-Semantik

CTL-Formeln werden über **Zuständen und Pfaden** von Kripke-Strukturen  $\mathcal{S} = (S, S_0, R, \ell)$  interpretiert

## Schreibweisen

- $s \models \zeta$  für Zustände  $s \in S$  und Zustandsformeln  $\zeta$
- $\pi \models \psi$  für Pfade  $\pi$  und Pfadformeln  $\psi$

## Hilfsbegriffe

- $\text{Paths}(s)$ : Menge aller Pfade, die in Zustand  $s$  beginnen
- $\pi[i]$ :  $i$ -ter Zustand auf dem Pfad  $\pi$   
d. h. wenn  $\pi = s_0s_1s_2 \dots$ , dann  $\pi[i] = s_i$

# CTL-Semantik

Sei  $\mathcal{S} = (S, S_0, R, \ell)$  eine Kripke-Struktur.

## Definition 4.1

**Erfülltheit von Zustandsformeln** in Zuständen  $s \in S$

$s \models p$  falls  $p \in \ell(s)$ , für alle  $p \in AV$

$s \models \neg\zeta$  falls  $s \not\models \zeta$

$s \models \zeta_1 \wedge \zeta_2$  falls  $s \models \zeta_1$  und  $s \models \zeta_2$  (analog für  $\zeta_1 \vee \zeta_2$ )

$s \models E\psi$  falls  $\pi \models \psi$  für ein  $\pi \in \text{Paths}(s)$

$s \models A\psi$  falls  $\pi \models \psi$  für alle  $\pi \in \text{Paths}(s)$

**Erfülltheit von Pfadformeln** in Pfaden  $\pi$  in  $\mathcal{S}$

$\pi \models X\zeta$  falls  $\pi[1] \models \zeta$  (analog für  $F\zeta$  und  $G\zeta$ )

$\pi \models \zeta_1 U \zeta_2$  falls  $\pi[j] \models \zeta_2$  für ein  $j \geq 0$   
und  $\pi[k] \models \zeta_1$  für alle  $k$  mit  $0 \leq k < j$

Schreiben  $\mathcal{S} \models \zeta$  falls  $s_0 \models \zeta$  für alle  $s_0 \in S_0$

# Zurück zu unseren Beispielen: Spezifikationen in CTL

## Beispiel Nebenläufigkeit

- Beide Teilprogramme sind nie zugleich im kritischen Bereich.  

$$AG\neg(p_{12} \wedge p_{22}) \quad (p_i \in AV: \text{„Programmzähler in Zeile } i\text{“})$$
- Jedes Teilprog. kommt beliebig oft in seinen krit. Bereich.  

$$AGAFp_{12} \wedge AGAFp_{22}$$
- Jedes Teilprog. *kann* beliebig oft in seinen kB kommen.  

$$AGEFp_{12} \wedge AGEFp_{22}$$

## Liveness properties:

$AG\zeta$  besagt: „ $\zeta$  ist in allen Berechnungen immer wahr“

$AGAF\zeta$  besagt: „ $\zeta$  ist in allen Berechnungen  $\infty$  oft wahr“

$AGEF\zeta$  besagt: „jede begonnene Berechnung kann so fortgesetzt werden, dass  $\zeta$  irgendwann wahr wird.“

# Zurück zu unseren Beispielen: Spezifikationen in CTL

## Beispiel Mikrowelle

- „Wenn Fehler auftritt, ist er nach endlicher Zeit behoben.“  
 $AG(e \rightarrow AF\neg e)$  ( $e \in AV$  steht für „Error“)
- „Wenn Fehler auftritt, *kann* er nach endl. Z. behoben werden“  
 $AG(e \rightarrow EF\neg e)$
- „Wenn die Mikrowelle gestartet wird, beginnt sie nach endlicher Zeit zu heizen.“  
 $AG(s \rightarrow AFh)$  ( $s, h \in AV$  stehen für „Start“ bzw. „Heat“)
- „Wenn die Mikrowelle gestartet wird, *ist es möglich*, dass sie nach endlicher Zeit zu heizen beginnt.“  
 $AG(s \rightarrow EFh)$

**Progress properties:**  $AG(\zeta_1 \rightarrow AF\zeta_2)$ ,  $AG(\zeta_1 \rightarrow EF\zeta_2)$  bedeuten:  
Wann immer  $\zeta_1$  eintritt, ist nach endlicher Zeit  $\zeta_2$  „garantiert“

# Ausdrucksstärke von CTL versus LTL

## Definition 4.2

Seien  $\zeta$  eine CTL-Zustandsformel und  $\varphi$  eine LTL-Formel.

$\zeta$  und  $\varphi$  sind **äquivalent**, geschrieben  $\zeta \equiv \varphi$ , wenn für alle Kripke-Strukturen  $\mathcal{S} = (S, S_0, R, \ell)$  gilt:

$$\mathcal{S} \models \zeta \quad \text{gdw.} \quad \mathcal{S} \models \varphi$$

### Zur Erinnerung:

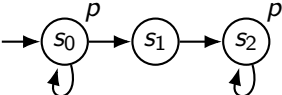
- $\mathcal{S} \models \zeta$ , wenn  $s_0 \models \zeta$  für **alle**  $s_0 \in S_0$
- $\mathcal{S} \models \varphi$ , wenn  $\pi, 0 \models \varphi$  für **alle**  $\pi \in \text{Paths}(s_0)$  und **alle**  $s_0 \in S_0$



## Ausdrucksstärke von CTL versus LTL

## Lemma 4.3

$$AFAGp \not\equiv FGp$$

**Beweis.** Betrachte Kripke-Struktur  $\mathcal{S}$ : 

- alle Pfade  $\pi \in \text{Paths}(s_0)$  erfüllen  $FGp$
- aber  $\mathcal{S} \not\models AFAGp$ :

$$\begin{aligned} & s_0 s_1 s_2^\omega \not\models Gp && \text{wegen } p \notin \ell(s_1) \\ \Rightarrow & s_0 \not\models AGp && \text{weil } s_0 s_1 s_2^\omega \in \text{Paths}(s_0) \\ \Rightarrow & s_0^\omega \not\models FAGp && \text{weil } s_0^\omega \text{ nur aus } s_0 \text{ besteht} \\ \Rightarrow & s_0 \not\models AFAGp && \text{weil } s_0^\omega \in \text{Paths}(s_0) \end{aligned}$$



# Ausdrucksstärke von CTL versus LTL

## Lemma 4.4

Sei  $\zeta$  eine CTL-Zustandsformel und  $\zeta'$  die LTL-Formel, die man durch Entfernen aller Pfadquantoren aus  $\zeta$  erhält. Dann gilt:

$\zeta \equiv \zeta'$  oder es gibt keine zu  $\zeta$  äquivalente LTL-Formel.

**Ohne Beweis.** (Clarke, Draghicescu 1988)

## Lemma 4.5

- (1) Es gibt keine zu  $AFAGp$  äquivalente LTL-Formel.
- (2) Es gibt keine zu  $FGp$  äquivalente CTL-Zustandsformel.

**Beweis.**

(1) folgt aus Lemmas 4.3 und 4.4

(2) siehe Tafel

**T 4.3**  $\square$

# Ausdrucksstärke von CTL versus LTL

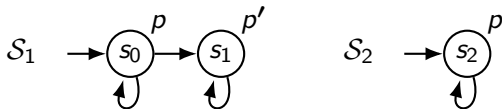
Auch **progress properties** sind **nicht** in LTL ausdrückbar:

## Lemma 4.6

Sei  $\zeta = AG(p \rightarrow Efp')$ . Es gibt keine zu  $\zeta$  äquivalente LTL-Formel.

**Beweis.** Angenommen, es gebe LTL-Formel  $\varphi \equiv \zeta$ .

Betrachte Kripke-Strukturen



Dann gilt  $\mathcal{S}_1 \models \zeta$ .

Also auch  $\mathcal{S}_1 \models \varphi$ .

Da  $\text{Paths}(s_2) \subseteq \text{Paths}(s_0)$ , gilt auch  $\mathcal{S}_2 \models \varphi$ .

Aber offensichtlich  $\mathcal{S}_2 \not\models \zeta$ . ⚡

□

# Ausdrucksstärke von CTL versus LTL

Erweiterung von LTL und CTL: **CTL\***

CTL\*: 1986 von E. A. Emerson und J. Y. Halpern (\*1953, Inform., Cornell)

# Model-Checking für CTL (Skizze)

**Standard-Algorithmus** („bottom-up labelling“, ohne Automaten):

Eingabe: Kripke-Str.  $\mathcal{S}$ , Zust.  $s_0$ , CTL-Zustandsformel  $\zeta$

Frage:  $s_0 \models \zeta$ ?

Vorgehen:

- Stelle  $\zeta$  als Baum dar (Bsp. siehe Tafel) **T 4.4**
- Gehe Baum von unten nach oben durch und markiere Zustände  $s$  in  $\mathcal{S}$  mit der jeweiligen Teilformel, wenn sie in  $s$  erfüllt ist **T 4.4 Forts.**
- Akzeptiere gdw.  $s_0$  mit  $\zeta$  markiert ist

**Komplexität:** P-vollständig (LTL-MC: PSpace-vollständig)

Dafür ist CTL-SAT ExpTime-vollständig (LTL-SAT: PSpace-vollst.).

# Model-Checking für CTL mit Baumautomaten

## Automatenbasierte Entscheidungsprozedur für CTL

... basiert auf **alternierenden Baumautomaten**

(Erweiterung des Begriffs der nichtdeterminist. Baumautomaten;  
siehe Teil 5 der Vorlesung)

## Verwandt:

Automatenbasierte Entscheidungsprozedur für CTL\*-**Erfüllbarkeit**

- basiert auf nichtdeterministischen Rabin-Baumautomaten
- technisch aufwändige Konstruktion
- hier nicht behandelt

## Es folgt:

Überblick „klassische“ nichtdeterministische Baumautomaten

# Und nun ...

- 1 *Model-Checking mit CTL*
- 2 Automaten auf unendlichen Bäumen
- 3 Komplementierung

# Baumautomaten: Grundbegriffe

## Betrachten unendlichen vollständigen Binärbaum

- Positionen: *alle* Wörter aus  $\{0, 1\}^*$
- jeder Knoten  $p$  hat linkes und rechtes Kind:  $p0, p1$
- **Tiefe** von Knoten  $p$ :  $|p|$
- **Ebene**  $k$ : alle Knoten der Tiefe  $k$
- $p_2$  ist **Nachfolger** von  $p_1$ , geschrieben  $p_1 \sqsubseteq p_2$ ,  
wenn  $p_2 = p_1p$  für ein  $p \in \{0, 1\}^*$

**T 4.5**

**Pfad:** Teilmenge  $\pi \subseteq \{0, 1\}^*$  mit  $\varepsilon \in \pi$  und:

- wenn  $p \in \pi$ , dann genau eins der Kinder  $p0, p1$  in  $\pi$
- $\forall k$ : von allen Knoten der Ebene  $k$  ist genau einer in  $\pi$

**T 4.5 Forts.**

**$\Sigma$ -Baum**  $t$  (Alphabet  $\Sigma$  ohne Stelligkeit):

Funktion  $t : \{0, 1\}^* \rightarrow \Sigma$

**T 4.5 Forts.**

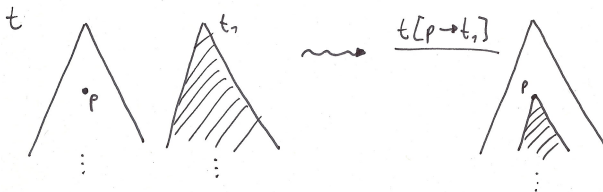


# Baumautomaten: etwas mehr Notation (1)

$$\hat{t} = t[p \rightarrow t_1]:$$

der Baum, den man aus  $t$  erhält, wenn man den Teilbaum, der in  $p$  wurzelt, durch  $t_1$  ersetzt

Skizze:



exakte Beschreibung:

$$\hat{t}(p') = \begin{cases} t_1(p'') & \text{wenn } p' = pp'' \\ t(p') & \text{wenn } p \not\sqsubseteq p' \end{cases}$$

# Baumautomaten: etwas mehr Notation (2)

$$\hat{t} = a(t_0, t_1):$$

der Baum mit Wurzel  $a$  und Teilbäumen  $t_0, t_1$  in den Wurzelkindern  $0, 1$ :

Skizze:



exakte Beschreibung:

$$\hat{t}(p) = \begin{cases} a & \text{wenn } p = \varepsilon \\ t_0(p') & \text{wenn } p = 0p' \\ t_1(p') & \text{wenn } p = 1p' \end{cases}$$

# Baumautomaten: Definition

## Definition 4.7

Ein **nichtdeterministischer Büchi-Baumautomat (NBBA)** über  $\Sigma$  ist ein 5-Tupel  $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ , wobei

- $Q$  eine endliche nichtleere **Zustandsmenge** ist,
- $\Sigma$  ein Alphabet ist
- $\Delta \subseteq Q \times \Sigma \times \underbrace{Q \times Q}$  die **Überföhrungsrelation** ist,
- $I \subseteq Q$  die Menge der **Anfangszustände** ist,
- $F \subseteq Q$  die Menge der **akzeptierenden Zustände** ist.

(entsprechen offenbar Top-down-Automaten)

# Muller- und Paritäts-Baumautomaten

## Definition 4.8

Ein **nichtdeterministischer Muller-Baumautomat (NMBA)** über  $\Sigma$  ist ein 5-Tupel  $\mathcal{A} = (Q, \Sigma, \Delta, I, \mathcal{F})$ , wobei

- $Q, \Sigma, \Delta, I$  wie für NBBA's sind
- $\mathcal{F} \subseteq 2^Q$  die **Akzeptanzkomponente** ist

Ein **nichtdeterministischer Paritäts-Baumautomat (NPBA)** über  $\Sigma$  ist ein 5-Tupel  $\mathcal{A} = (Q, \Sigma, \Delta, I, c)$ , wobei

- $Q, \Sigma, \Delta, I$  wie für NBBA's sind
- $c : Q \rightarrow \mathbb{N}$  die **Akzeptanzkomponente** ist

(Rabin- und Streett-Baumautomaten wie üblich definiert)

# Runs auf Baumautomaten

Run = Markierung der Positionen in  $\{0, 1\}^*$  mit Zuständen, verträglich mit Anfangszuständen und Überföhrungsrelation

## Definition 4.9

Ein **Run** eines NBBA (NMBA, NPBA)  $\mathcal{A}$  auf einem  $\Sigma$ -Baum  $t$  ist eine Funktion  $r : \{0, 1\}^* \rightarrow Q$ , so dass

- $r(\varepsilon) \in I$ ;
- für alle  $p \in \{0, 1\}^*$  gilt:  $(r(p), t(p), r(p0), r(p1)) \in \Delta$

Erfolgreicher Run: verträglich mit Akzeptanzkomponente

# Erfolgreiche Runs

Sei  $r$  Run eines NxBAs  $\mathcal{A}$  und  $\pi$  ein Pfad

Betrachten wieder **Unendlichkeitsmenge**

$$\text{Inf}(r, \pi) = \{q \in Q \mid r(p) = q \text{ für unendlich viele } p \in \pi\}$$

## Definition 4.10

- Run  $r$  des NBBA  $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$  ist **erfolgreich**, falls **für alle Pfade**  $\pi$  gilt:  $\text{Inf}(r, \pi) \cap F \neq \emptyset$
- Run  $r$  des NMBA  $\mathcal{A} = (Q, \Sigma, \Delta, I, \mathcal{F})$  ist **erfolgreich**, falls **für alle Pfade**  $\pi$  gilt:  $\text{Inf}(r, \pi) \in \mathcal{F}$
- Run  $r$  des NPBA  $\mathcal{A} = (Q, \Sigma, \Delta, I, c)$  ist **erfolgreich**, falls **für alle Pfade**  $\pi$  gilt:  $\min\{c(q) \mid q \in \text{Inf}(r, \pi)\}$  ist gerade

# Akzeptanz und erkannte Sprache

... sind wie üblich definiert:

## Definition 4.11

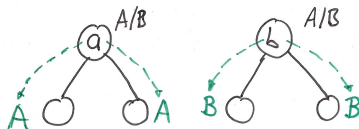
Sei  $\mathcal{A}$  ein NBBA, NMBA oder NPBA,  
sei  $t$  ein  $\Sigma$ -Baum und  $L$  eine Menge von  $\Sigma$ -Bäumen.

- $\mathcal{A}$  **akzeptiert**  $t$ ,  
wenn es einen erfolgreichen Run von  $\mathcal{A}$  auf  $t$  gibt.
- $L_\omega(\mathcal{A}) = \{t \mid \mathcal{A} \text{ akzeptiert } t\}$
- $L$  heißt **Büchi-erkennbar**,  
wenn es einen NBBA  $\mathcal{A}$  gibt mit  $L_\omega(\mathcal{A}) = L$ .
- Analog: **Muller-erkennbar** und **paritäts-erkennbar**

# Beispiele (Büchi)

- NBBA  $\mathcal{A} = (\{A, B\}, \{a, b\}, \Delta, \{A\}, \{A\})$  mit  
 $\Delta = \{ (A, a, A, A), (B, a, A, A), (A, b, B, B), (B, b, B, B) \}$

Skizze:



$$L_\omega(\mathcal{A}) = \{t \mid \text{jeder Pfad hat } \infty \text{ viele } a\text{'s}\}$$

- derselbe NBBA, aber mit  $F = \{B\}$   
 $L_\omega(\mathcal{A}) = \{t \mid \text{jeder Pfad hat } \infty \text{ viele } b\text{'s}\}$

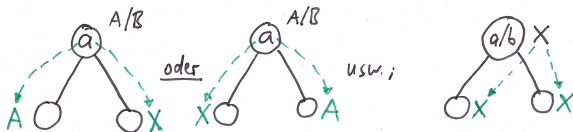
- derselbe NBBA, aber mit  $F = \{A, B\}$   
 $L_\omega(\mathcal{A}) = \{t \mid t \text{ ist ein } \Sigma\text{-Baum}\}$



# Beispiele (Büchi)

- NBBA  $\mathcal{A} = (\{A, B, X\}, \{a, b\}, \Delta, \{A\}, \{A, X\})$  mit  $\Delta = \{ (A, a, A, X), (B, a, A, X), (A, b, B, X), (B, b, B, X), (X, a, X, X), (A, a, X, A), (B, a, X, A), (A, b, X, B), (B, b, X, B), (X, b, X, X) \}$

Skizze:



$$L_\omega(\mathcal{A}) = \{t \mid t \text{ hat mind. einen Pfad mit } \infty \text{ vielen } a\text{'s}\}$$

- derselbe NBBA, aber mit  $F = \{B, X\}$

$$L_\omega(\mathcal{A}) = \{t \mid t \text{ hat mind. einen Pfad mit } \infty \text{ vielen } b\text{'s}\}$$

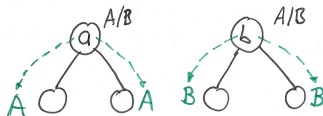
- derselbe NBBA, aber mit  $F = \{X\}$ :  $L_\omega(\mathcal{A}) = \emptyset$

- derselbe NBBA, aber mit  $F = \{A, B\}$

# Beispiele (Muller)

- NMBA  $\mathcal{A} = (\{A, B\}, \{a, b\}, \Delta, \{A\}, \{\{A\}\})$  mit  
 $\Delta = \{ (A, a, A, A), (B, a, A, A), (A, b, B, B), (B, b, B, B) \}$

Skizze:



$L_\omega(\mathcal{A}) = \{t \mid \text{jeder Pfad hat endlich viele } b\text{'s}\} (!)$

- derselbe NMBA, aber mit  $F = \{\{B\}\}$   
 $L_\omega(\mathcal{A}) = \{t \mid \text{jeder Pfad hat endlich viele } a\text{'s}\}$
- derselbe NMBA, aber mit  $F = \{\{A, B\}\}$   
 $L_\omega(\mathcal{A}) = \{t \mid \text{jeder Pfad hat } \infty \text{ viele } a\text{'s und } \infty \text{ viele } b\text{'s}\}$
- derselbe NMBA, aber mit  $F = \{\{A\}, \{B\}\}$   
 $L_\omega(\mathcal{A}) = \{t \mid \text{jeder Pfad hat endl. viele } b\text{'s oder endl. viele } a\text{'s}\}$

# Beispiel (Parität)

## Zur Erinnerung:

Run  $r$  ist erfolgreich, wenn für alle Pfade  $\pi \subseteq T$  gilt:

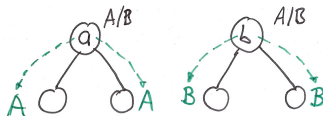
$$\min\{c(q) \mid q \in \text{Inf}(r, \pi)\} \text{ ist gerade}$$

NPBA  $\mathcal{A} = (\{A, B\}, \{a, b\}, \Delta, \{A\}, c)$  mit

$$\Delta = \{ (A, a, A, A), (B, a, A, A), (A, b, B, B), (B, b, B, B) \}$$

$$c(A) = 1$$

$$c(B) = 2$$



$$L_\omega(\mathcal{A}) = \{t \mid \text{jeder Pfad hat endlich viele } a\text{'s}\}$$

# Büchi- versus Muller-Erkenbarkeit

## Satz 4.12

- 1 Jede Büchi-erkennbare Sprache ist Muller-erkennbar.
- 2 **Nicht jede** Muller-erkennbare Sprache ist Büchi-erkennbar.

## Beweis.

- 1 Wie im letzten Kapitel.
- 2 Betrachten  $L = \{t \mid \text{jeder Pfad in } t \text{ hat endlich viele } a\text{'s}\}$   
 $L$  ist Muller-erkennbar (siehe Bsp. auf Folie 34)  
Müssen zeigen:  $L$  nicht Büchi-erkennbar

# Büchi- versus Muller-Erkennbarkeit

**Zu zeigen:**  $L = \{t \mid \text{jeder Pfad in } t \text{ hat endlich viele } a\text{'s}\}$   
nicht Büchi-erkennbar.

Nehmen an, es gebe NBBA  $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$  mit  $L_\omega(\mathcal{A}) = L$ .

O. B. d. A. sei  $I = \{q_0\}$ . Sei  $n := |F|$ .

**Idee:**

- Bestimme Baum  $t \in L$  mit Run  $r$  und Pfad, auf dem zwischen 2 Besuchen *desselben* akzeptierenden Zustandes ein  $a$  auftritt
- „Pumpe“  $t, r$  so auf, dass dieser Teilpfad sich  $\infty$  oft wiederholt
- ⚡ Neuer Baum wird akzeptiert, aber neuer Pfad hat  $\infty$  viele  $a$ 's

# Büchi- versus Muller-Erkennbarkeit

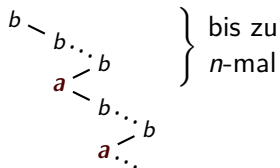
Betrachte Baum  $t \in L$  mit  $t(p) = a$  gdw.  $p \in \bigcup_{i=1, \dots, n} (1^+0)^i$ ,

d. h.  $t$  enthält ein  $a$  an allen Positionen, die man erreicht, wenn man bei der Wurzel startet und bis zu  $n$ -mal wie folgt läuft:

- einmal oder mehrmals zum rechten Kind (beliebig oft)
- einmal zum linken Kind („Linksschritt“)

An den übrigen Positionen enthält  $t$  ein  $b$ .

**Skizze:**



Klar:  $t \in L$ . Sei  $r$  ein erfolgreicher Run.

Details des Pumpens: s. Tafel

**T 4.6**  $\square$

# Folgerung aus dem Beweis von Satz 4.12

## Folgerung 4.13

Die Klasse der Büchi-erkennbaren Baumsprachen ist **nicht** abgeschlossen unter Komplement.

Man kann Satz 4.12 stärker formulieren (ohne Beweis):

## Satz 4.14

Die Menge der Baumsprachen, die Muller-, aber nicht Büchi-erkennbar sind, ist

$$\{L_{\Delta} \mid L \text{ ist NBA-erkennbar, aber nicht DBA-erkennbar}\}.$$

( $L \subseteq \Sigma^{\omega}$  ist eine  $\omega$ -Sprache;

$L_{\Delta}$  = Menge aller  $\Sigma$ -Bäume, deren Beschriftung entlang *jedes* Pfades in  $L$  liegt)

# Paritäts- versus Muller-Erkennbarkeit

## Satz 4.15

- 1 Jede paritäts-erkennbare Sprache ist Muller-erkennbar.
- 2 Jede Muller-erkennbare Sprache ist paritäts-erkennbar.

### Beweis.

- 1 Wie im letzten Kapitel.
- 2 Sei  $\mathcal{A} = (Q, \Sigma, \Delta, I, \mathcal{F})$  NMBA, mit  $I = \{q_0\}$  und  $\mathcal{F} = \{F\}$  (o. B. d. A.) sowie  $n := |Q|$ .

**Gesucht:** äquivalenter NPBA  $\mathcal{A}'$

**Idee:**  $\mathcal{A}'$  soll ...

- „sich merken“, in welcher Reihenfolge die  $n$  Zustände zuletzt gesehen wurden (Permutation  $q_1 \cdots q_n$  von  $Q$ )
- sicherstellen, dass ab einem gewissen Zeitpunkt genau die Zustände aus  $F$  dauerhaft am Ende der Permutation stehen



# Details der Konstruktion (1)

Sei  $\mathcal{A} = (Q, \Sigma, \Delta, \{q_0\}, F)$  NMBA mit  $|Q| = n$ .

Konstruieren NPBA  $\mathcal{A}' = (Q', \Sigma, \Delta', I', c)$  mit Zuständen

$$Q' = \{ \langle q_1 \cdots q_n, \ell \rangle \mid q_1 \cdots q_n \text{ ist Permutation von } Q, \\ \ell \in \{1, \dots, n\} \}$$

**Idee:**

- $q_n$  ist der zuletzt besuchte Zustand auf dem aktuellen Pfad,  $q_{n-1}$  der zuletzt besuchte Zustand  $\neq q_n$  usw.
- $\ell$  ist Position von  $q_n$  in der vorangehenden Permutation

Skizze: siehe Tafel

**T 4.7**

## Details der Konstruktion (2)

Zeigen zunächst folgende **Hilfsaussage (HA)** über Zustände von  $\mathcal{A}'$

Sei  $q_1 q_2 q_3 \dots$  eine Folge von Zuständen aus  $Q$ ;  
 sei  $s_1 s_2 s_3 \dots$  die zugehörige Folge von Zuständen aus  $Q'$   
 mit  $s_1 = \langle t_1 \dots t_{n-1} q_1, 1 \rangle$  und  $s_i = \langle \text{perm}_i, \ell_i \rangle$  für alle  $i \geq 0$ .

Dann gilt  $\text{Inf}(q_1 q_2 q_3 \dots) = S$  mit  $|S| = k$  gdw.

- ① Für endlich viele  $i$  ist  $\ell_i \leq n - k$  und
- ② Für unendlich viele  $i$  gilt:
  - (a)  $\ell_i = n - k + 1$  und
  - (b) Die Menge der Zustände in den Positionen  $\underbrace{n - k + 1, \dots, n}_{\text{letzte } k \text{ Positionen}}$  in  $\text{perm}_i$  ist  $S$

**Beweis der Hilfsaussage:** siehe Tafel

**T 4.8**

# Details der Konstruktion (3)

Können nun Konstruktion fortsetzen:

$$I' = \left\{ \langle t_1 \cdots t_{n-1} q_0, 1 \rangle \mid t_1 \cdots t_{n-1} \text{ ist Perm. von } Q \setminus \{q_0\} \right\}$$

$$\Delta' = \left\{ \left( \langle i_1 \cdots i_{n-1} i, \ell \rangle, a, \langle i'_1 \cdots i'_{n-1} i', \ell' \rangle, \langle i''_1 \cdots i''_{n-1} i'', \ell'' \rangle \right) \mid \right.$$

- $(i, a, i', i'') \in \Delta$
- $i'_1 \cdots i'_{n-1}$  entsteht aus  $i_1 \cdots i_{n-1} i$  durch Löschen von  $i'$
- $i''_1 \cdots i''_{n-1}$  entsteht aus  $i_1 \cdots i_{n-1} i$  durch Löschen von  $i''$
- $\ell' =$  Position von  $i'$  in  $i_1 \cdots i_{n-1} i$
- $\ell'' =$  Position von  $i''$  in  $i_1 \cdots i_{n-1} i$

$$\left. \right\}$$

$$c(s) = \begin{cases} 2\ell & \text{falls } s = \langle q_1 \cdots q_n, \ell \rangle \text{ und } \{q_\ell, \dots, q_n\} = F \\ 2\ell + 1 & \text{falls } s = \langle q_1 \cdots q_n, \ell \rangle \text{ und } \{q_\ell, \dots, q_n\} \neq F \end{cases}$$

Beweis der Korrektheit: siehe Tafel

**T 4.9**  $\square$

# Abschlusseigenschaften

## Satz 4.16

Die Klasse der ...

- 1 Büchi-erkennbaren Sprachen ist abgeschlossen unter  $\cup$  und  $\cap$ , aber nicht unter  $\bar{\phantom{x}}$ .
- 2 Muller-erkennbaren Sprachen ist abgeschlossen unter  $\cup, \cap, \bar{\phantom{x}}$ .

**Beweis:**

- 1  $\cup \cap$  wie gehabt;  $\bar{\phantom{x}}$  siehe Folgerung 4.13.
- 2  $\cup \cap$  wie gehabt;  
 $\bar{\phantom{x}}$  siehe nächsten Abschnitt □

# Und nun ...

- 1 *Model-Checking mit CTL*
- 2 Automaten auf unendlichen Bäumen
- 3 Komplementierung**

# Überblick

## Ziel dieses Abschnitts:

Lösen Komplementierung mit Hilfe eines bekannten Resultates über Gewinnstrategien in einer bestimmten Art (abstrakter) Spiele

## Vorgehen:

- Ordnen jedem NPBA  $\mathcal{A}$  und Baum  $t$  ein 2-Personen-Spiel  $G_{\mathcal{A},t}$  zu (Beschränkung auf NPBAs ist unerheblich, siehe Satz 4.15)
- Dann wird leicht zu sehen sein:  
 $\mathcal{A}$  akzeptiert  $t \Leftrightarrow$  Spielerin 1 hat Gewinnstrategie in  $G_{\mathcal{A},t}$
- Ein Resultat aus der Spieltheorie impliziert:  
In  $G_{\mathcal{A},t}$  hat immer **genau eine Spielerin eine Gewinnstrategie, die nicht vom bisherigen Spielverlauf abhängt**
- Konstruieren  $\mathcal{A}'$ , so dass gilt:  
 $\mathcal{A}'$  akzeptiert  $t \Leftrightarrow$  **Spielerin 2 hat Gewinnstrategie in  $G_{\mathcal{A},t}$**   
Dann folgt  $L_\omega(\mathcal{A}') = \overline{L_\omega(\mathcal{A})}$

# Intuitive Beschreibung des Spiels $G_{\mathcal{A},t}$

Zwei Spielerinnen **Aut** (Automat), **PF** (Pfadfinderin)

- sind abwechselnd an der Reihe
- bewegen sich pro Runde 1 Schritt im Baum durch Markieren von Positionen mit Zuständen; zu Beginn:  $(\varepsilon, q_I)$ ,  $q_I \in I$

In jeder Runde wählt

- **Aut** eine Transition, die auf die markierte Position anwendbar ist
- **PF** einen Kindknoten und verschiebt Markierung dorthin

Spiel läuft  $\infty$  lange, erzeugt  $\infty$  Folge  $r = q_0q_1q_2$  von Zuständen (bestimmt durch die gewählten Transitionen)

**Aut** gewinnt, wenn  $r$  der Akzeptanzbedingung von  $\mathcal{A}$  entspricht;  
sonst gewinnt **PF**

(d. h. **Aut** versucht,  $\mathcal{A}$  zum Akzeptieren zu bringen; **PF** versucht das zu verhindern)

Skizze: s. Tafel

**T 4.10**

# Genauere Beschreibung des Spiels $G_{A,t}$

Spiel ist ein unendlicher Graph

- Knoten sind die **Spielpositionen**:
  - für **Aut**:  $\{(p, q) \mid p \in \{0, 1\}^*, q \in Q\}$  (Positionen im Baum)
  - für **PF**:  $\{(q, t(p), q_0, q_1) \in \Delta \mid p \in \{0, 1\}^*\}$  (Transitionen)
- Kanten sind die möglichen **Spielzüge**:
  - $(p, q) \rightarrow (q, t(p), q_0, q_1)$
  - $(q, t(p), q_0, q_1) \rightarrow (p0, q_0)$   
 $\quad \quad \quad \rightarrow (p1, q_1)$
- Startknoten:  $(\varepsilon, q_I)$  für  $q_I \in I$  (o. B. d. A.  $I = \{q_I\}$ )

Jede mögliche  $\infty$  Folge von Spielzügen entspricht einem  $\infty$  Pfad im Spielbaum  $G_{A,t}$

Knoten  $v'$  **erreichbar** von Knoten  $v$ :

es gibt endliche Folge von Spielzügen von  $v$  nach  $v'$



# Spielstrategien

**Strategie ab Spielposition  $v$**  für Spielerin  $X \in \{\mathbf{Aut}, \mathbf{PF}\}$ :

Funktion, die jeder Zugfolge  $v \dots v'$  mit  $v'$  Spielposition für  $X$  einen in  $v'$  möglichen Zug zuweist

(legt fest, welchen Zug  $X$  in jeder von  $v$  aus erreichbaren Spielposition macht)

**Gewinnstrategie** für Spielerin  $X \in \{\mathbf{Aut}, \mathbf{PF}\}$ :

Strategie, die sicherstellt, dass  $X$  gewinnt,  
*unabhängig* von den Zügen der Gegenspielerin

**T 4.11**

**gedächtnislose Strategie:**

Strategie, die nur von  $v'$  abhängt, nicht von den vorigen Positionen

# Akzeptanz mittels Gewinnstrategien

## Lemma 4.17

Seien  $\mathcal{A} = (Q, \Sigma, \Delta, \{q_I\}, c)$  ein NPBA und  $t$  ein  $\Sigma$ -Baum.  
Dann gilt:

$t \in L_\omega(\mathcal{A}) \Leftrightarrow \text{Aut hat Gewinnstrategie in } G_{\mathcal{A},t} \text{ ab Position } (\varepsilon, q_I)$

## Beweis:

Konstruiere Gewinnstrategie direkt aus einem erfolgreichen Run  
und umgekehrt

# Akzeptanz mittels Gewinnstrategien

„ $t \in L_\omega(\mathcal{A}) \Rightarrow \mathbf{Aut}$  hat Gewinnstrategie in  $G_{\mathcal{A},t}$  ab Position  $(\varepsilon, q_I)$ “

Gelte  $t \in L_\omega(\mathcal{A})$  und sei  $r$  akzeptierender Run von  $\mathcal{A}$  auf  $t$ .

Konstruiere Gewinnstrategie für  $\mathbf{Aut}$  wie folgt aus  $r$ .

- in Startposition  $(\varepsilon, q_I)$  wähle  $(r(\varepsilon), t(\varepsilon), r(0), r(1))$
- in allen anderen Spielpos.  $(p, q)$  wähle  $(q, t(p), r(p0), r(p1))$

Wenn  $\mathbf{Aut}$  diese Strategie befolgt, dann entspricht die im Spiel erzeugte Zustandsmenge einem Pfad in  $r$ .

Da  $r$  akzeptierend, gewinnt  $\mathbf{Aut}$  nach Definition von  $G_{\mathcal{A},t}$ .

„ $\mathbf{Aut}$  hat Gewinnstrategie in  $G_{\mathcal{A},t}$  ab Position  $(\varepsilon, q_I) \Rightarrow t \in L_\omega(\mathcal{A})$ “

T 4.12

# Determiniertheit von Paritätsspielen

Klassisches Resultat aus der Spieltheorie, hier nicht bewiesen:

Satz 4.18 (Emerson & Jutla 1991, Mostowski 1991)

Alle Paritätsspiele sind **gedächtnislos determiniert**:  
genau eine der Spielerinnen hat eine gedächtnislose Gewinnstrategie.

„Paritätsspiel“ bezeichnet dabei 2-Personen-Spiele, die

- auf Graphen gespielt werden,  
deren Knoten mit natürlichen Zahlen markiert sind;
- als Gewinnbedingung für unendliche Spielverläufe  
die Paritätsbedingung verwenden.

Für alle  $\mathcal{A}$  und  $t$  ist  $G_{\mathcal{A},t}$  ein Paritätsspiel.

# Determiniertheit von Paritätsspielen

Folgerung aus Satz 4.18:

## Folgerung 4.19

Seien  $\mathcal{A} = (Q, \Sigma, \Delta, \{q_I\}, c)$  ein NPBA und  $t$  ein  $\Sigma$ -Baum.

Dann gibt es für jede Spielposition  $v$  in  $G_{\mathcal{A},t}$  — und insbesondere für  $(\varepsilon, q_I)$  — eine gedächtnislose Gewinnstrategie für **Aut** oder **PF**.

## Folgerung 4.20 (aus Lemma 4.17 und Folgerung 4.19)

$t \in \overline{L_\omega(\mathcal{A})} \Leftrightarrow$  **PF** hat gedächtnislose GS ab  $(\varepsilon, q_I)$  in  $G_{\mathcal{A},t}$

**Ziel:** konstruieren NPBA, um deren Existenz zu testen

# Gewinnbäume

Betrachten gedächtnislose Gewinnstrategien für **PF** als Menge von Funktionen

$$f_p : \Delta \rightarrow \{0, 1\} \quad \text{für jede Baumposition } p \in \{0, 1\}^*$$

**Idee:**  $f_p$  weist jeder Transition, die **Aut** in Baumposition  $p$  wählt, einen Spielzug (Richtung 0/1) zu

- Sei  $F$  die Menge dieser Funktionen
- Ordnen die  $f_p$  in einem **F-Baum**  $s$  an (Strategiebaum)

**PF-Gewinnbaum** für  $t$ :

ein  $F$ -Baum, der eine Gewinnstrategie für **PF** in  $G_{\mathcal{A}, t}$  kodiert

Folgerung 4.21 (aus Folgerung 4.20)

$t \in \overline{L_\omega(\mathcal{A})} \Leftrightarrow$  es gibt einen **PF-Gewinnbaum** für  $t$

**Neues Ziel:** bauen NPBA, um Existenz **PF-Gewinnbaum** zu testen

# Existenz von PF-Gewinnbäumen (PF-GB)

Sei  $\mathcal{A} = (Q, \Sigma, \Delta, \{q_I\}, c)$  ein NPBA und  $t$  ein  $\Sigma$ -Baum

**Zwischenziel:** Prüfen, ob gegebener  $F$ -Baum  $s$  **kein** PF-GB ist

Idee:

- Benutzen NPA  $\mathcal{A}'$  ( $\omega$ -Wortautomat)
- $\mathcal{A}'$  prüft für jeden Pfad  $\pi$  in  $t$  und jeden möglichen Spielzug von **Aut** separat, ob Akzeptanzbedingung von  $\mathcal{A}$  erfüllt ist

$\rightsquigarrow \mathcal{A}'$  akzeptiert  $\geq 1$  Pfad  $\Leftrightarrow s$  ist kein PF-Gewinnbaum für  $t$

Sei  $\pi \in \{0, 1\}^\omega$  ein Pfad mit  $\pi = \pi_1\pi_2\pi_3 \dots$

$\mathcal{A}'$  arbeitet auf Wörtern der folgenden Form:

$$\langle s(\varepsilon), t(\varepsilon), \pi_1 \rangle \quad \langle s(\pi_1), t(\pi_1), \pi_2 \rangle \quad \langle s(\pi_1\pi_2), t(\pi_1\pi_2), \pi_3 \rangle \quad \dots$$

Sei  $L_{s,t}$  die Sprache aller dieser Wörter

Beispiel: s. Tafel

**T 4.13**

# Konstruktion des Wortautomaten für Gewinnbäume

Sei  $\mathcal{A} = (Q, \Sigma, \Delta, \{q_I\}, c)$  ein NPBA und  $t$  ein  $\Sigma$ -Baum

Konstruieren NPA  $\mathcal{A}' = (Q, \Sigma', \Delta', \{q_I\}, c)$  wie folgt:

- $\Sigma' = \{ \langle f, a, i \rangle \mid f \in F, a \in \Sigma, i \in \{0, 1\} \}$
- $Q, c$  wie in  $\mathcal{A}$  (wollen Akzeptanz von  $\mathcal{A}$  prüfen)
- $\Delta' = \{ (q, \langle f, a, i \rangle, q'_i) \mid \langle f, a, i \rangle \in \Sigma', i \in \{0, 1\}, \text{ es gibt } \delta = (q, a, q'_0, q'_1) \in \Delta \text{ mit } f(\delta) = i \}$

$\mathcal{A}'$  prüft für *jeden möglichen* Zug von **Aut**, ob **Aut** gewinnen kann

## Lemma 4.22

$s$  ist ein **PF**-Gewinnbaum für  $t \Leftrightarrow L_{s,t} \cap L_\omega(\mathcal{A}') = \emptyset$

**Beweis:** s. Tafel

**T 4.14**  $\square$



# Konstruktion des Komplementautomaten für $\mathcal{A}$

**Gesucht:** (siehe Folgerung 4.21)

NPBA  $\mathcal{B}$ , der  $t$  akzeptiert gdw. es einen **PF**-Gewinnbaum für  $t$  gibt

Wegen Lemma 4.22 muss  $\mathcal{B}$  akzeptieren gdw.  $L_{s,t} \subseteq \overline{L_\omega(\mathcal{A}' )}$

Konstruktion von  $\mathcal{B}$  in 2 Schritten:

## Schritt 1

- Sei  $\mathcal{A}'' = (Q'', \Sigma', \Delta'', q_l'', c'')$  der **DPA** mit  $L_\omega(\mathcal{A}'') = \overline{L_\omega(\mathcal{A}' )}$
- $\mathcal{A}''$  ist **deterministisch**: Safra-Konstruktion  
(+ Umwandlung zwischen den Automatentypen)

## Schritt 2

$\mathcal{B}$  soll auf jedem Pfad von  $t$

- $\mathcal{A}''$  laufen lassen
- und „parallel“ dazu eine Strategie für **PF** raten

# Konstruktion des Komplementautomaten für $\mathcal{A}$

**Idee:** NPBA  $\mathcal{B}$  soll  $\mathcal{A}''$  auf **jedem** Pfad simulieren, indem  $\mathcal{B}$

- $s$  rät (also pro Position  $p$  ein  $f_p$ )
- sich ansonsten wie  $\mathcal{A}''$  verhält,  
(also pro Position die Folgezustände  $q_0, q_1$  gemäß  $\Delta''$  setzt)

$\mathcal{A}''$  deterministisch  $\Rightarrow$  Zustand pro Position  $p$  eindeutig bestimmt

**Konstruktion von  $\mathcal{B} = (Q'', \Sigma, \Delta^{\text{neu}}, q_1'', c'')$ :**

- $Q'', q_1'', c''$  werden von  $\mathcal{A}''$  übernommen
- $\Delta^{\text{neu}} = \left\{ (q, a, q_0, q_1) \mid \text{es gibt } f \in F \text{ mit} \right.$   
 $\left. (q, \langle f, a, i \rangle, q_i) \in \Delta'' \text{ f\"ur } i = 0, 1 \right\}$

**Lemma 4.23**

$t \in L_\omega(\mathcal{B})$  gdw. es gibt  $F$ -Baum  $s$  mit  $L_{s,t} \subseteq L_\omega(\mathcal{A}'')$

**Beweis:** siehe Tafel

**T 4.15**  $\square$

... Es darf aufgeatmet werden ... 😊

# Das Resultat

## Satz 4.24 (Rabin 1969)

Für jeden NPBA  $\mathcal{A}$  gibt es einen NPBA  $\mathcal{B}$  mit  $L_\omega(\mathcal{B}) = \overline{L_\omega(\mathcal{A})}$ .

### Beweis:

Für den bisher konstruierten NPBA  $\mathcal{B}$  gilt:

- $t \in L_\omega(\mathcal{B})$  gdw.  $\exists s . L_{s,t} \subseteq L_\omega(\mathcal{A}'')$  (Lemma 4.23)
- gdw.  $\exists s . L_{s,t} \subseteq \overline{L_\omega(\mathcal{A}'')}$  (Konstr.  $\mathcal{A}''$ )
- gdw.  $\exists s . L_{s,t} \cap L_\omega(\mathcal{A}') = \emptyset$  (Mengenlehre)
- gdw.  $\exists$  PF-Gewinnbaum  $s$  für  $t$  (Lemma 4.22)
- gdw.  $t \in \overline{L_\omega(\mathcal{A})}$  (Folg. 4.21)



# Bemerkungen zur Komplexität der Konstruktion

Sei  $n = |Q|$  (Anzahl der Zustände des NBPA  $\mathcal{A}$ ).

Dann hat der NPA  $\mathcal{A}'$  dieselben  $n$  Zustände.

DPA  $\mathcal{A}''$  kann so konstruiert werden, dass  $|Q''| \in O(2^{n \log n})$ .

$\rightsquigarrow$  NBPA  $\mathcal{B}$  hat  $O(2^{n \log n})$  Zustände.

# Literatur für diesen Teil (Basis, 1)



E. Grädel, W. Thomas, T. Wilke (Hrsg.).

**Automata, Logics, and Infinite Games.**

LNCS 2500, Springer, 2002, S. 43–60.

Kapitel 6–9 über Paritätsspiele und Baumautomaten.

<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 4.

<http://www.informatik.uni-bremen.de/tdki/lehre/ws09/automata/automata-notes.pdf>

# Literatur für diesen Teil (Basis, 2)



Christel Baier, Joost-Pieter Katoen.

Principles of Model Checking.

MIT Press 2008.

Abschnitt 6 „Computation Tree Logic“.

SUB, Zentrale:  $a \inf 440 \text{ ver}/782$ ,  $a \inf 440 \text{ ver}/782a$



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

Model Checking.

MIT Press 1999.

Abschnitt 3 „Temporal Logics“,

Abschnitt 4 „Model Checking“.

SUB, Zentrale:  $a \inf 440 \text{ ver}/780(6)$ ,  $a \inf 440 \text{ ver}/780(6)a$

# Literatur für diesen Teil (weiterführend)



Edmund M. Clarke, I. A. Draghicescu

Expressibility Results for Linear-Time and Branching-Time Logics

REX Workshop 1988, S. 428–437.

<https://doi.org/10.1007/BFb0013029>