# Automatentheorie und ihre Anwendungen
# Teil 4: Automaten auf unendlichen Bäumen (Ausblick)

Thomas Schneider

1. Juli 2013



## Überblick

- Grenzen von LTL – welche Eigenschaften kann man nicht ausdrücken?
- Berechnungsbäume und CTL
- Ausdrucksvermögen von LTL und CTL im Vergleich
- Model-Checking mit CTL
- zugehöriges Automatenmodell: Büchi-Automaten auf unendlichen Bäumen
- äquivalente Automatenmodelle



## Erinnerung an LTL *(Linear Temporal Logic)*

- System gegeben als Kripke-Struktur $\mathcal{S} = (S, S_0, R, L)$
- LTL-Formel $\varphi_E$ beschreibt Pfade, die Eigenschaft $E$ erfüllen
- Beispiel:
  „Wenn Fehler auftritt, ist er nach endlicher Zeit behoben."
  $G(e \rightarrow F\neg e)$ ($e \in$ PROP 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

## Ein Fall für CTL *(Computation Tree Logic)*

**Abhilfe:** Betrachten Berechnungsbäume statt Pfaden

- Idee: Baum enthält *alle* Pfade, die in $s_0$ starten

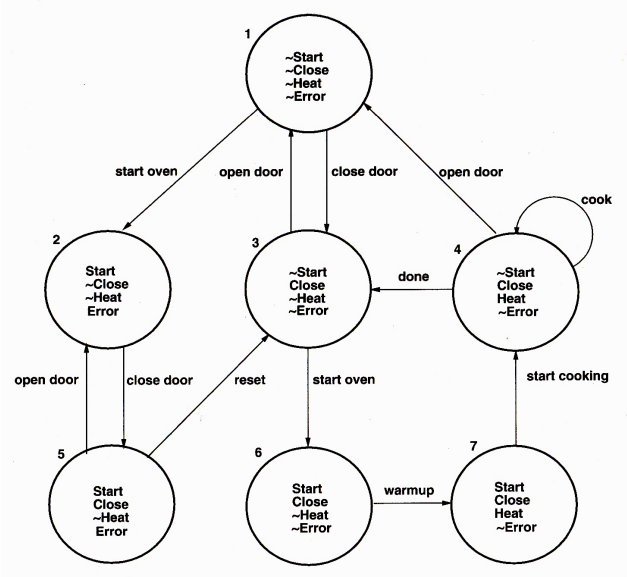
$\rightsquigarrow$ Zustand $s$ hat als Kinder alle seine Nachfolgerzustände in $\mathcal{S}$

$\rightsquigarrow$ Baum entsteht durch „Auffalten“ von $\mathcal{S}$ in $s_0$

- Beispiel: siehe Tafel



## Beispielstruktur Mikrowelle




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



## 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 ($A$)
gibt es immer ($G$)
eine Möglichkeit, die Berechnung fortzusetzen ($E$),
so dass irgendwann ($F$)
kein Fehler auftritt ($\neg e$)

CTL 1981 eingeführt durch
Edmund M. Clarke, *1945, Informatiker, Carnegie Mellon Univ. (Pittsburgh)
E. Allen Emerson, ?, Informatiker, Univ. of Texas, Austin, USA
(beide Turing-Award-Träger 2007)



## CTL formal

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 e. temporalen Operator gefolgt werden
- jeder temporale Operator direkt einem Pfadquantor folgen

## Quiz: zulässige Formeln

Zur Erinnerung:

$$(ZF) \quad \zeta ::= p \mid \zeta_1 \wedge \zeta_2 \mid \zeta_1 \vee \zeta_2 \mid \neg\zeta \mid E\psi \mid A\psi$$
$$(PF) \quad \psi ::= F\zeta \mid G\zeta \mid X\zeta \mid \zeta_1 \, U \, \zeta_2$$

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

- $p \wedge q \quad EFp \quad AXp$ ✓
- $E(p \, U \, q)$ ✓
- $A((p \vee \neg p) \, U \, q)$ ✓ (äquivalent zu $AFq$)
- $E(p \vee AXq)$ ✘
- $EX(p \vee AXq)$ ✓
- $EF(p \, U \, q)$ ✘
- $EFA(p \, U \, q)$ ✓



## Zurück zu unseren Beispielen: Spezifikationen in CTL

**Beispiel Nebenläufigkeit**

- Es kommt nie vor,
  dass beide Teilprog. zugleich im kritischen Bereich sind.
  $AG(\neg(p_{12} \vee \neg p_{22})$ ($p_i \in$ PROP: „Programmzähler in Zeile $i$“)
- Jedes Teilprog. kommt beliebig oft in seinen krit. Bereich.
  $AGAFp_{12} \wedge AGAFp_{22}$
- Jedes Teilprog. *kann* beliebig oft in seinen kB kommen.
  $AGEFp_{12} \wedge AGEFp_{22}$

▶ $AGAF\zeta$ besagt: „$\zeta$ ist auf allen Pfaden unendlich oft wahr“
$AGEF\zeta$ besagt: „jeder Pfad kann so fortgesetzt werden,
dass $\zeta$ irgendwann wahr wird.“

**(liveness properties)**



## Zurück zu unseren Beispielen: Spezifikationen in CTL

**Beispiel Mikrowelle**

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

▶ $AG(\zeta_1 \to AF\zeta_2)$, $AG(\zeta_1 \to EF\zeta_2)$: **progress properties**



## CTL im Kontext anderer Temporallogiken

(Semantik ähnlich wie für LTL definiert)

LTL und CTL sind **bezüglich Ausdrucksstärke unvergleichbar**.

- Bsp.: $\zeta = AFAGp$ und $\varphi = FGp$ **nicht** äquivalent
- es gibt keine zu $\zeta$ äquivalente LTL-Formel (o. Beweis)
- es gibt keine zu $\varphi$ äquivalente CTL-Formel (o. Beweis)

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

eingeführt 1986 von E. Allen Emerson und Joseph Y. Halpern
(J. Y. Halpern, ?, Informatiker, Cornell University (Ithaca, NY, USA))

## Model-Checking für CTL (Skizze)

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

- Gegeben: Kripke-Struktur $\mathcal{S}$, Zustand $s_0$, CTL-Formel $\zeta$
- Frage: ist $\varphi$ in $s_0$ erfüllt?
- Stelle $\zeta_E$ als Baum dar (Bsp. $\zeta_E = p \vee AXE(q\ U\ r)$)
- Gehe Baum von unten nach oben durch
  und markiere Zustände $s$ in $S$ mit der jeweiligen Teilformel,
  wenn sie in $s$ erfüllt ist
- Akzeptiere gdw. $s$ mit $\zeta_E$ markiert ist

**Komplexität:** P-vollständig
(zur Erinnerung: LTL-MC ist PSPACE-vollständig)



## Model-Checking für CTL mit Baumautomaten

**Automatenbasierte Entscheidungsprozedur für CTL**

- basiert auf **alternierenden Baumautomaten**
  (Erweiterung des Begriffs der nichtdeterminist. Baumautomaten)
- hier nicht behandelt
  (siehe V + Ü „Verifikation unendlicher Systeme“, WS 13/14)

**Es folgt**: Überblick „normale“ nichdeterministische Baumautomaten



## Büchi-Baumautomaten: Grundbegriffe

Betrachten **unendlichen Binärbaum** $T$

- Positionen $\{0,1\}^*$
- jeder Knoten $w$ hat linkes und rechtes Kind: $w0, w1$
- **Ebene, Vorgängerknoten** definiert wie üblich

**Pfad** $\pi$ in $T$: Teilmenge $\pi \subseteq T$ mit

- Wurzel $\varepsilon \in \pi$
- wenn $w \in \pi$, dann genau eins der Kinder $w0, w1$ in $\pi$

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

Funktion $t : T \to \Sigma$



## Büchi-Baumautomaten: Definition

**Definition 1**

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
- $\Delta \subseteq Q \times \Sigma \times Q \times Q$ die **Überführungsrelation** ist,
- $I \subseteq Q$ die Menge der **Anfangszustände** ist,
- $F \subseteq Q$ die Menge der **Endzustände** ist.

(entsprechen offenbar Top-down-Automaten)

## Muller- und Paritäts-Baumautomaten

**Definition 2**

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 NBBAs 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 NBBAs sind
- $c : Q \to \mathbb{N}$ die **Akzeptanzkomponente** ist

(Rabin- und Streett-Baumautomaten wie üblich definiert)



## Runs auf Baumautomaten

**Definition 3**

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

- $r(\varepsilon) \in I$;
- für alle $w \in T$ gilt: $(r(w), t(w), r(w0), r(w1)) \in \Delta$

- Run = Markierung der Positionen in $T$ mit Zuständen, verträglich mit Anfangszuständen und Überführungsrelation
- Erfolgreicher Run: verträglich mit Akzeptanzkomponente ($\downarrow$)



## Erfolgreiche Runs

Sei $r$ Run eines N$x$BAs $\mathcal{A}$ und $\pi$ ein Pfad

Betrachten wieder **Unendlichkeitsmenge**

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

**Definition 4**

Run $r$ des N$x$BA $\mathcal{A} = (Q, \Sigma, \Delta, I, Acc)$ ist **erfolgreich**, falls

- $x$ = Büchi; $Acc = F$; für alle Pfade $\pi \subseteq T$: $\mathrm{Inf}(r, \pi) \cap F \neq \emptyset$
- $x$ = Muller; $Acc = \mathcal{F}$; für alle Pfade $\pi \subseteq T$: $\mathrm{Inf}(r, \pi) \in \mathcal{F}$
- $x$ = Parität; $Acc = c$; für alle Pfade $\pi \subseteq T$:

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

$\mathcal{A}$ **akzeptiert** $t$, wenn es einen erfolgreichen Run von $\mathcal{A}$ auf $t$ gibt.

$L_\omega(\mathcal{A}) = \{t \mid \mathcal{A} \text{ akzeptiert } t\}$



## Beispiele (Büchi)

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

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

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

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

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

## Beispiele (Büchi)

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

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

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

- derselbe Büchi-Automat, aber mit $F = \{B, X\}$
  $L_\omega(\mathcal{A}) = \{t \mid t$ hat mind. einen Pfad mit $\infty$ vielen $b$'s$\}$

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

- derselbe Büchi-Automat, aber mit $F = \{A, B\}$: $L_\omega(\mathcal{A}) = \emptyset$



## Beispiele (Muller)

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

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

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

- derselbe Muller-Automat, aber mit $F = \{\{B\}\}$
  $L_\omega(\mathcal{A}) = \{t \mid$ jeder Pfad hat endlich viele $a$'s$\}$

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

- derselbe Muller-Automat, aber mit $F = \{\{A\}, \{B\}\}$
  $L_\omega(\mathcal{A}) = \{t \mid$ jeder Pfad hat endl. viele $b$'s *oder* endl. viele $a$'s$\}$



## Beispiel (Parität)

**Zur Erinnerung:**

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

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

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

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

$c(A) = 1$
$c(B) = 2$

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



## Beziehungen zwischen den Baumsprachenklassen

**Satz 5**

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

**Beweisskizze.**

1. Wie im letzten Kapitel.
2. Nimm erstes Beispiel auf Folie 22
   und zeige, dass diese Sprache nicht Büchi-erkennbar ist.
   Argumente ähnlich zum Pumping-Lemma auf endl. Bäumen.

**Folgerung 6**

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

## Beziehungen zwischen den Baumsprachenklassen

**Satz 7**

- *Jede Muller-erkennbare Sprache ist paritäts-erkennbar.*
- *Jede paritäts-erkennbare Sprache ist Muller-erkennbar.*

**Ohne Beweis.**



## Abschlusseigenschaften

**Satz 8**

*Die Klasse der ...*

1. *Büchi-erkennbaren Sprachen ist abgeschlossen unter $\cup$ und $\cap$, aber nicht unter $^-$.*
2. *Muller-erkennbaren Sprachen ist abgeschlossen unter $\cup, \cap, ^-$.*

**Beweisidee:**

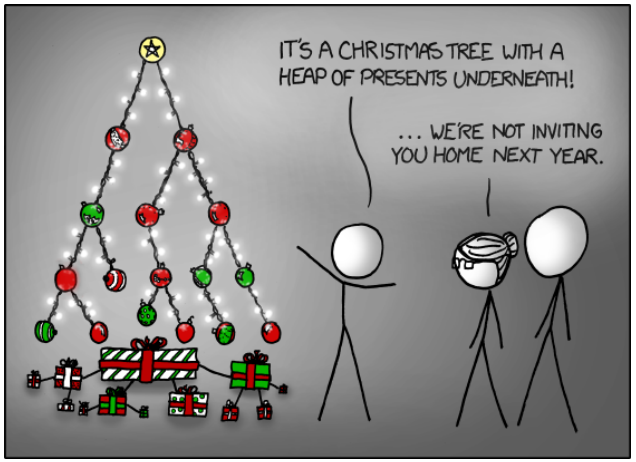
1. $\cup\,\cap$ wie gehabt; $^-$ siehe Folgerung 6.
2. $\cup\,\cap$ wie gehabt;
   $^-$ anspruchsvoller Beweis, Resultat aus der Spieltheorie (•)



## Das war ...

... der Überblick über Automaten auf unendlichen Bäumen.

**Vielen Dank für eure Teilnahme!**




Quelle: `http://www.explainxkcd.com/wiki/index.php?title=835:_Tree`



## Wir sind noch nicht ganz fertig ...

Am Mittwoch:

- Übung
- Nachbesprechung Evaluation
- Hinweise zu Fachgesprächen
- Themenvorschläge für Abschlussarbeiten
- „Eigenwerbung“:
  V + Ü „Verifikation unendlicher Systeme“, WS 13/14

  Vortragender: Dr. Stefan Göller

  `http://www.informatik.uni-bremen.de/tdki/teaching.html`

  `http://www.uni-bremen.de/studium/lehrveranstaltungen/veranstaltungsverzeichnis.html`

## Literatur für diesen Teil (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 (2)


Christel Baier, Joost-Pieter Katoen.

Principles of Model Checking.

MIT Press 2008.

Abschnitt 6 „Computation Tree Logic“.

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 3 „Temporal Logics“,
Abschnitt 4 „Model Checking“.

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