Automatentheorie und ihre Anwendungen
Teil 5: Alternierung

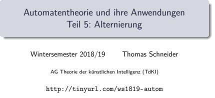
Wintersemester 2018/19 Thomas Schneider

AG Theorie der künstlichen Intelligenz (TdKI)

`http://tinyurl.com/ws1819-autom`

2019-02-01

Teil 5: Alternierung


Automatentheorie und ihre Anwendungen
Teil 5: Alternierung
Wintersemester 2018/19 Thomas Schneider
AG Theorie der künstlichen Intelligenz (TdKI)
http://tinyurl.com/ws1819-autom

8:30

TODO:
- Übersetzung ABA $\longrightarrow$ NBA besser verstehen (v. a. Beweisdetails in T5.8, 5.10 und Bsp.-Automat T5.11).
- Beweise sauberer führen (Finkbeiner-Skript z. T. lakonisch & hat Fehler)
- Intuition hinter Automatenkonstruktion klarer fassen (X, W)
- konkreten Automaten mit Bsp.-Run, dem kein Run-DAG entspricht, angeben

TODO: Weitere Themen siehe TODO.txt.

Warum Alternierung?

- Starke Beziehungen zwischen Logik und Automaten, z. B.:
  - NBAs $\leftrightarrow$ LTL (Teil 3 dieser Vorlesung)
  - NEAs $\leftrightarrow$ S1S (Satz von Büchi-Elgot-Trakhtenbrot, VL Logik)
- In Logiken kann man aber Sprachen oder Eigenschaften oft deutlich kürzer ausdrücken, z. B.:
  - LTL-Formel $\to$ NBA: exponentielle Explosion
  - S1S-Formel $\to$ NEA: sogar nicht-elementare Explosion
- Verkleinern dieser Lücke:

  Erlaube in Automaten nicht nur **existenzielle** (= nichtdeterm.) „Verzweigungen“, sondern auch **universelle**.

Warum Alternierung?
Starke Beziehungen zwischen Logik und Automaten, z. B.:
NBAs $\leftrightarrow$ LTL (Teil 3 dieser Vorlesung)
NEAs $\leftrightarrow$ S1S (Satz von Büchi-Elgot-Trakhtenbrot, VL Logik)
In Logiken kann man aber Sprachen oder Eigenschaften oft deutlich kürzer ausdrücken, z. B.:
LTL-Formel $\to$ NBA: exponentielle Explosion
S1S-Formel $\to$ NEA: sogar nicht-elementare Explosion
Verkleinern dieser Lücke:
Erlaube in Automaten nicht nur existenzielle (= nichtdeterm.) „Verzweigungen“, sondern auch universelle.

8:30

Büchi-Elgot-Trakhtenbrot: Logik-VL. S1S = monad. SO auf lin. Strukt.

„nicht-elementar“: jede Negation erfordert Potenzmengenkonstruktion, vergrößert Automaten exp.
$\to$ „exp. Turm“ unbeschränkt (Verschachtelungstiefe $\neg$ (und $\exists$))

Um die Lücke zu verkleinern, erweitert man das Automatenmodell so, dass es der Logik ähnlicher wird.

Warum Alternierung?

- „Alternierung“ heißt also, dass ein Maschinenmodell (abwechselnd) existenzielle und universelle Entscheidungen treffen kann.
- Alternierende Varianten gibt es für alle Automatentypen aus dieser Vorlesung (auf endlichen oder unendlichen Objekten, Wörtern oder Bäumen) und für andere Maschinenmodelle (z. B. Turingmaschinen).
- Für alternierende Automaten ist Komplementierung besonders leicht zu erreichen.
- Wir beschränken uns im Folgenden auf ω-**Wort**automaten, also auf alternierende Büchi-Automaten.

Warum Alternierung?

- „Alternierung“ heißt also, dass ein Maschinenmodell (abwechselnd) existenzielle und universelle Entscheidungen treffen kann.
- Alternierende Varianten gibt es für alle Automatentypen aus dieser Vorlesung (auf endlichen oder unendlichen Objekten, Wörtern oder Bäumen) und für andere Maschinenmodelle (z. B. Turingmaschinen).
- Für alternierende Automaten ist Komplementierung besonders leicht zu erreichen.
- Wir beschränken uns im Folgenden auf ω-**Wort**automaten, also auf alternierende Büchi-Automaten.

8:32

„Abwechselnd“ ist hier wichtig. Nur ex./nur univ. ist witzlos.

Für TMs: erlaubt z. B. feinere Komplexitätsanalyse

Überblick

1. Einführung und Grundbegriffe
2. Von LTL zu alternierenden Automaten
3. Komplementierung

Überblick
1 Einführung und Grundbegriffe
2 Von LTL zu alternierenden Automaten
3 Komplementierung

8:33

Und nun ...

Und nun ...
1 Einführung und Grundbegriffe
2 Von LTL zu alternierenden Automaten
3 Komplementierung

Alternierung: Grundidee

- Nichtdeterministischer Automat $\mathcal{A}$ akzeptiert eine Eingabe, wenn ein erfolgreicher Run **existiert**.

  d. h.: falls $(q, a, q'),\ (q, a, q'') \in \Delta$, kann $\mathcal{A}$ in Situation (q, a) „entscheiden", wie der Run fortgesetzt wird.

  Mindestens eine dieser Entscheidungen muss zum Ziel führen.

- Alternierung erlaubt auch **universelle Entscheidungen**, in beliebiger Kombination mit existenziellen.

- „Beliebige Kombination" wird realisiert durch **positive Boolesche Formel**, d. h. aussagenlogische Formel ohne $\neg$.

- Statt eines Runs (Zustandsfolge) gibt es nun einen **Run-Baum,** der alle universellen Entscheidungen berücksichtigt.

Alternierung: Grundidee

- Nichtdeterministischer Automat $\mathcal{A}$ akzeptiert eine Eingabe, wenn ein erfolgreicher Run **existiert**. d. h.: falls $(q, a, q'),\ (q, a, q'') \in \Delta$, kann $\mathcal{A}$ in Situation (q, a) „entscheiden", wie der Run fortgesetzt wird. **Mindestens eine** dieser Entscheidungen muss zum Ziel führen.
- Alternierung erlaubt auch **universelle Entscheidungen**, in beliebiger Kombination mit existenziellen.
- „Beliebige Kombination" wird realisiert durch **positive Boolesche Formel**, d. h. aussagenlogische Formel ohne $\neg$.
- Statt eines Runs (Zustandsfolge) gibt es nun einen **Run-Baum**, der alle universellen Entscheidungen berücksichtigt.

8:33

Positive Formeln werden manchmal auch monoton genannt. Hier egal.
Wichtig: keine Negation!

Positive Boolesche Formeln

Definition 5.1 (Syntax)

Die Menge der **positiven Booleschen Formeln (PBFs)** über einer Menge X, geschrieben $\mathbf{B}^+(X)$, ist die kleinste Menge, für die gilt:

- Jedes Element $x \in X$ ist eine PBF.
- Die Konstanten $0, 1$ sind PBFs.
- Wenn φ, ψ PBFs sind, dann auch $\varphi \wedge \psi$ und $\varphi \vee \psi$.

Positive Boolesche Formeln

Definition 5.1 (Syntax)

Die Menge der positiven Booleschen Formeln (PBFs) über einer Menge X, geschrieben $\mathbf{B}^+(X)$, ist die kleinste Menge, für die gilt:

- Jedes Element $x \in X$ ist eine PBF.
- Die Konstanten $0, 1$ sind PBFs.
- Wenn φ, ψ PBFs sind, dann auch $\varphi \wedge \psi$ und $\varphi \vee \psi$.

8:35

Positive Boolesche Formeln

Definition 5.1 (Syntax)

Die Menge der **positiven Booleschen Formeln (PBFs)** über einer Menge X, geschrieben $\mathbf{B^+(X)}$, ist die kleinste Menge, für die gilt:

- Jedes Element $x \in X$ ist eine PBF.
- Die Konstanten $0, 1$ sind PBFs.
- Wenn φ, ψ PBFs sind, dann auch $\varphi \wedge \psi$ und $\varphi \vee \psi$.

Definition 5.2 (Semantik)

Jede Menge $Y \subseteq X$ definiert eine **Belegung** $\boldsymbol{V_Y} : X \to \{0, 1\}$:

$$V_Y(x) = 1, \text{ falls } x \in Y; \quad V_Y(x) = 0 \text{ sonst.}$$

Eine Menge $Y \subseteq X$ **erfüllt** eine PBF $\varphi \in B^+(X)$, geschrieben $\boldsymbol{Y \models \varphi}$, wenn $V_Y \models \varphi$ (nach Standard-Semantik AL).

T 5.1

Positive Boolesche Formeln

Definition 5.1 (Syntax)

Die Menge der positiven Booleschen Formeln (PBFs) über einer Menge X, geschrieben $B^+(X)$, ist die kleinste Menge, für die gilt:

- Jedes Element $x \in X$ ist eine PBF.
- Die Konstanten $0, 1$ sind PBFs.
- Wenn φ, ψ PBFs sind, dann auch $\varphi \wedge \psi$ und $\varphi \vee \psi$.

Definition 5.2 (Semantik)

Jede Menge $Y \subseteq X$ definiert eine Belegung $V_Y : X \to \{0, 1\}$: $V_Y(x) = 1$, falls $x \in Y$; $V_Y(x) = 0$ sonst.

Eine Menge $Y \subseteq X$ erfüllt eine PBF $\varphi \in B^+(X)$, geschrieben $Y \models \varphi$, wenn $V_Y \models \varphi$ (nach Standard-Semantik AL).

T 5.1

8:35

Alternierende Automaten

Definition 5.3

Ein **alternierender Büchi-Automat** auf ω-Wörtern **(ABA)** ist ein 5-Tupel $\mathcal{A} = (Q, \Sigma, \delta, I, F)$, wobei

- Q eine endliche nichtleere **Zustandsmenge** ist,
- Σ eine **Alphabet** (endliche nichtleere Menge von Zeichen) ist,
- $\delta : Q \times \Sigma \to \mathrm{B}^{+}(Q)$ die **Überführungsfunktion** ist,
- $I \subseteq Q$ die Menge der **Anfangszustände** ist,
- $F \subseteq Q$ die Menge der **akzeptierenden Zustände** ist.

Alternierende Automaten
Definition 5.3
Ein alternierender Büchi-Automat auf ω-Wörtern (ABA) ist ein 5-Tupel $\mathcal{A} = (Q, \Sigma, \delta, I, F)$, wobei
- Q eine endliche nichtleere Zustandsmenge ist,
- Σ eine Alphabet (endliche nichtleere Menge von Zeichen) ist,
- $\delta : Q \times \Sigma \to \mathrm{B}^{+}(Q)$ die Überführungsfunktion ist,
- $I \subseteq Q$ die Menge der Anfangszustände ist,
- $F \subseteq Q$ die Menge der akzeptierenden Zustände ist.

8:41

Überführungs**funktion:** weil die PBF bereits die nichtdeterministischen Entscheidungen enthält

Alternierende Automaten

Definition 5.3

Ein **alternierender Büchi-Automat** auf ω-Wörtern **(ABA)** ist ein 5-Tupel $\mathcal{A} = (Q, \Sigma, \delta, I, F)$, wobei

- Q eine endliche nichtleere **Zustandsmenge** ist,
- Σ eine **Alphabet** (endliche nichtleere Menge von Zeichen) ist,
- $\delta : Q \times \Sigma \to \mathrm{B}^+(Q)$ die **Überführungsfunktion** ist,
- $I \subseteq Q$ die Menge der **Anfangszustände** ist,
- $F \subseteq Q$ die Menge der **akzeptierenden Zustände** ist.

Wir nehmen wieder o. B. d. A. $I = \{q_I\}$ an.

Alternative Akzeptanzbedingungen (Muller, Parität usw.) sind auch möglich.

Alternierende Automaten

Definition 5.3

Ein alternierender Büchi-Automat auf ω-Wörtern (ABA) ist ein 5-Tupel $\mathcal{A} = (Q, \Sigma, \delta, I, F)$, wobei

- Q eine endliche nichtleere Zustandsmenge ist,
- Σ eine Alphabet (endliche nichtleere Menge von Zeichen) ist,
- $\delta : Q \times \Sigma \to \mathrm{B}^+(Q)$ die Überführungsfunktion ist,
- $I \subseteq Q$ die Menge der Anfangszustände ist,
- $F \subseteq Q$ die Menge der akzeptierenden Zustände ist.

Wir nehmen wieder o. B. d. A. $I = \{q_I\}$ an.

Alternative Akzeptanzbedingungen (Muller, Parität usw.) sind auch möglich.

8:41

Überführungs**funktion**: weil die PBF bereits die nichtdeterministischen Entscheidungen enthält

Run-Bäume

Betrachten **Baum mit Verzweigungsgrad $\leq n$,** für festes $n \in \mathbb{N}$

- Positionen: Menge $P \subseteq \{1, \dots, n\}^*$, präfix-abgeschlossen
- **Kinder** eines Knotens p: $\textbf{Kinder}(p) \subseteq \{p1, \dots, pn\}$
- **Tiefe, Ebene, Nachfolger, Pfad:** wie gehabt

Run-Bäume

Betrachten **Baum mit Verzweigungsgrad $\leq n$,** für festes $n \in \mathbb{N}$

- Positionen: Menge $P \subseteq \{1, \dots, n\}^*$, präfix-abgeschlossen
- **Kinder** eines Knotens p: $\textbf{Kinder}(p) \subseteq \{p1, \dots, pn\}$
- **Tiefe, Ebene, Nachfolger, Pfad:** wie gehabt

8:43

Runs sind jetzt Bäume mit endlichem Verzweigungsgrad; Bäume müssen nicht vollständig sein.

Präfix-Abg.: wie bei endlichen Bäumen; jedes Kind braucht sein Elter!

Run-Bäume

Betrachten **Baum mit Verzweigungsgrad $\leq n$**, für festes $n \in \mathbb{N}$

- Positionen: Menge $P \subseteq \{1, \ldots, n\}^*$, präfix-abgeschlossen
- **Kinder** eines Knotens p: **Kinder**$(p) \subseteq \{p1, \ldots, pn\}$
- **Tiefe, Ebene, Nachfolger, Pfad:** wie gehabt

Pfad in P: endliche oder unendliche Folge $\pi = \pi_0\pi_1\pi_2\cdots$ von **Positionen** $\pi_i \in P$ mit

- $\pi_0 = \varepsilon$ und
- $\pi_{i+1} \in \text{Kinder}(\pi_i)$ für alle $i \geq 0$

Run-Bäume
Betrachten Baum mit Verzweigungsgrad $\leq n$, für festes $n \in \mathbb{N}$
Positionen: Menge $P \subseteq \{1, \ldots, n\}^*$, präfix-abgeschlossen
Kinder eines Knotens p: Kinder$(p) \subseteq \{p1, \ldots, pn\}$
Tiefe, Ebene, Nachfolger, Pfad: wie gehabt
Pfad in P: endliche oder unendliche Folge $\pi = \pi_0\pi_1\pi_2\cdots$ von Positionen $\pi_i \in P$ mit
$\pi_0 = \varepsilon$ und
$\pi_{i+1} \in \text{Kinder}(\pi_i)$ für alle $i \geq 0$

8:43

Runs sind jetzt Bäume mit endlichem Verzweigungsgrad; Bäume müssen nicht vollständig sein.

Präfix-Abg.: wie bei endlichen Bäumen; jedes Kind braucht sein Elter!

Run-Bäume

Betrachten **Baum mit Verzweigungsgrad $\leq n$**, für festes $n \in \mathbb{N}$

- Positionen: Menge $P \subseteq \{1, \ldots, n\}^*$, präfix-abgeschlossen
- **Kinder** eines Knotens p: **Kinder**$(p) \subseteq \{p1, \ldots, pn\}$
- **Tiefe, Ebene, Nachfolger, Pfad:** wie gehabt

Pfad in P: endliche oder unendliche Folge $\pi = \pi_0\pi_1\pi_2\cdots$ von **Positionen** $\pi_i \in P$ mit

- $\pi_0 = \varepsilon$ und
- $\pi_{i+1} \in$ Kinder(π_i) für alle $i \geq 0$

Σ-Baum (P, t) (Alphabet Σ):

- P wie oben
- $t : P \to \Sigma$ ist Markierungsfunktion **T 5.2**

Run-Bäume

Betrachten Baum mit Verzweigungsgrad $\leq n$, für festes $n \in \mathbb{N}$
- Positionen: Menge $P \subseteq \{1, \ldots, n\}^*$, präfix-abgeschlossen
- Kinder eines Knotens p: Kinder$(p) \subseteq \{p1, \ldots, pn\}$
- Tiefe, Ebene, Nachfolger, Pfad: wie gehabt

Pfad in P: endliche oder unendliche Folge $\pi = \pi_0\pi_1\pi_2\cdots$ von Positionen $\pi_i \in P$ mit
- $\pi_0 = \varepsilon$ und
- $\pi_{i+1} \in$ Kinder(π_i) für alle $i \geq 0$

Σ-Baum (P, t) (Alphabet Σ):
- P wie oben
- $t : P \to \Sigma$ ist Markierungsfunktion T 5.2

8:43

Runs sind jetzt Bäume mit endlichem Verzweigungsgrad; Bäume müssen nicht vollständig sein.

Präfix-Abg.: wie bei endlichen Bäumen; jedes Kind braucht sein Elter!

Berechnungen und Akzeptanz

Definition 5.4

Ein **Run** eines ABA $\mathcal{A} = (Q, \Sigma, \delta, \{q_i\}, F)$ auf einem Wort $\alpha = \alpha_0\alpha_1\alpha_2 \cdots \in \Sigma^\omega$ ist ein ***Q*-Baum** (P, r), so dass:

- $r(\varepsilon) = q_I$
- für alle $p \in P$: wenn $r(p) = q$, dann

$$\left\{ r(p') \mid p' \in \text{Kinder}(p) \right\} \models \delta(q, \alpha_{|p|}) .$$

T 5.3

(für andere Akzeptanzbedingungen analog)

2019-02-01

Teil 5: Alternierung
└Einführung und Grundbegriffe
└Berechnungen und Akzeptanz

Berechnungen und Akzeptanz

Definition 5.4

Ein Run eines ABA $\mathcal{A} = (Q, \Sigma, \delta, \{q_i\}, F)$ auf einem Wort $\alpha = \alpha_0\alpha_1\alpha_2 \cdots \in \Sigma^\omega$ ist ein *Q*-Baum (P, r), so dass:

- $r(\varepsilon) = q_I$
- für alle $p \in P$: wenn $r(p) = q$, dann $\left\{ r(p') \mid p' \in \text{Kinder}(p) \right\} \models \delta(q, \alpha_{|p|})$. T 5.3

(für andere Akzeptanzbedingungen analog)

8:48 bis 8:59

Bedingung 2, Run:
die Kinder eines Knotens p enthalten **eine** Menge von Zuständen,
die die PBF erfüllen, die δ dem Paar $\big(r(p), |\text{p}|\text{-tes Zeichen von } \alpha\big)$ zuweist

Ein Beispiel an Tafel; weiteres Beispiel im nächsten Abschnitt (LTL)!

Berechnungen und Akzeptanz

Definition 5.4

Ein **Run** eines ABA $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, F)$ auf einem Wort $\alpha = \alpha_0\alpha_1\alpha_2 \cdots \in \Sigma^\omega$ ist ein **Q-Baum** (P, r), so dass:

- $r(\varepsilon) = q_I$
- für alle $p \in P$: wenn $r(p) = q$, dann

$$\left\{ r(p') \mid p' \in \text{Kinder}(p) \right\} \models \delta(q, \alpha_{|p|}) \,.$$ **T 5.3**

Run (P, r) ist **erfolgreich**, wenn für **jeden unendlichen** Pfad $\pi = \pi_0\pi_1\pi_2 \ldots$ in P gilt:

$$\text{Inf}(r, \pi) \cap F \neq \emptyset$$ **T 5.3 Forts.**

(für andere Akzeptanzbedingungen analog)

Berechnungen und Akzeptanz
Definition 5.4
Ein Run eines ABA $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, F)$ auf einem Wort $\alpha = \alpha_0\alpha_1\alpha_2 \cdots \in \Sigma^\omega$ ist ein Q-Baum (P, r), so dass:
$r(\varepsilon) = q_I$
für alle $p \in P$: wenn $r(p) = q$, dann
$\left\{ r(p') \mid p' \in \text{Kinder}(p) \right\} \models \delta(q, \alpha_{|p|})$. T 5.3
Run (P, r) ist erfolgreich, wenn für jeden unendlichen Pfad $\pi = \pi_0\pi_1\pi_2 \ldots$ in P gilt:
$\text{Inf}(r, \pi) \cap F \neq \emptyset$ T 5.3 Forts.
(für andere Akzeptanzbedingungen analog)

8:48 bis 8:59

Bedingung 2, Run:
die Kinder eines Knotens p enthalten **eine** Menge von Zuständen,
die die PBF erfüllen, die δ dem Paar $\big(r(p), |p|\text{-tes Zeichen von } \alpha\big)$ zuweist

Ein Beispiel an Tafel; weiteres Beispiel im nächsten Abschnitt (LTL)!

Berechnungen und Akzeptanz

Definition 5.4

Ein **Run** eines ABA $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, F)$ auf einem Wort $\alpha = \alpha_0\alpha_1\alpha_2 \dots \in \Sigma^\omega$ ist ein ***Q*-Baum** (P, r), so dass:

- $r(\varepsilon) = q_I$
- für alle $p \in P$: wenn $r(p) = q$, dann

$$\left\{ r(p') \mid p' \in \text{Kinder}(p) \right\} \models \delta(q, \alpha_{|p|}) \,. \qquad \textbf{T 5.3}$$

Run (P, r) ist **erfolgreich**, wenn für **jeden unendlichen** Pfad $\pi = \pi_0\pi_1\pi_2 \dots$ in P gilt:

$$\text{Inf}(r, \pi) \cap F \neq \emptyset \qquad \textbf{T 5.3 Forts.}$$

$\boldsymbol{L_\omega(\mathcal{A})} = \{\alpha \in \Sigma^\omega \mid \mathcal{A} \text{ hat einen erfolgr. Run auf } \alpha\}$ **T 5.3 Forts.**

(für andere Akzeptanzbedingungen analog)

Berechnungen und Akzeptanz
Definition 5.4
Ein Run eines ABA $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, F)$ auf einem Wort $\alpha = \alpha_0\alpha_1\alpha_2 \dots \in \Sigma^\omega$ ist ein Q-Baum (P, r), so dass:
- $r(\varepsilon) = q_I$
- für alle $p \in P$: wenn $r(p) = q$, dann $\left\{ r(p') \mid p' \in \text{Kinder}(p) \right\} \models \delta(q, \alpha_{|p|})$. T 5.3
Run (P, r) ist erfolgreich, wenn für jeden unendlichen Pfad $\pi = \pi_0\pi_1\pi_2 \dots$ in P gilt: $\text{Inf}(r, \pi) \cap F \neq \emptyset$ T 5.3 Forts.
$L_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \mathcal{A} \text{ hat einen erfolgr. Run auf } \alpha\}$ T 5.3 Forts.
(für andere Akzeptanzbedingungen analog)

8:48 bis 8:59

Bedingung 2, Run:
die Kinder eines Knotens p enthalten **eine** Menge von Zuständen,
die die PBF erfüllen, die δ dem Paar $\big(r(p), |p|\text{-tes Zeichen von } \alpha\big)$ zuweist

Ein Beispiel an Tafel; weiteres Beispiel im nächsten Abschnitt (LTL)!

Und nun ...

1. Einführung und Grundbegriffe
2. Von LTL zu alternierenden Automaten
3. Komplementierung

Teil 5: Alternierung
└Von LTL zu alternierenden Automaten
└Und nun ...

2019-02-01

Und nun ...

1. Einführung und Grundbegriffe
2. Von LTL zu alternierenden Automaten
3. Komplementierung

Vorbetrachtungen

Übersetzung logischer Formeln in alternierende Automaten ist oft einfacher als in nichtdeterministische Automaten.

Hier am Beispiel LTL $\rightarrow$ ABA

Vorbetrachtungen
Übersetzung logischer Formeln in alternierende Automaten ist oft einfacher als in nichtdeterministische Automaten.
Hier am Beispiel LTL → ABA

8:59

Formeln hier als Grammatik angegeben.

- Variablen als x, nicht p (schon für Positionen vergeben).
- $\vee$ ist eigentlich überflüssig, kann man aber gleich ganz bequem behandeln.
- F, G sind nur Abkürzungen mittels U.

Expansionsgesetz: kann man leicht semantisch überprüfen.
Haben wir damals auch im (G)NBA kodiert.
Allerdings jetzt einfacher mit ABA modellierbar!

Vorbetrachtungen

Übersetzung logischer Formeln in alternierende Automaten
ist oft einfacher als in nichtdeterministische Automaten.

Hier am Beispiel LTL $\to$ ABA

Erinnerung an LTL: $\varphi ::= x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi\, U\, \varphi$
mit $x \in$ AV (Aussagenvariablen)

Vorbetrachtungen
Übersetzung logischer Formeln in alternierende Automaten
ist oft einfacher als in nichtdeterministische Automaten.
Hier am Beispiel LTL $\to$ ABA
Erinnerung an LTL: $\varphi ::= x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi\, U\, \varphi$
mit $x \in$ AV (Aussagenvariablen)

8:59
Formeln hier als Grammatik angegeben.
- Variablen als x, nicht p (schon für Positionen vergeben).
- $\vee$ ist eigentlich überflüssig, kann man aber gleich ganz bequem behandeln.
- F, G sind nur Abkürzungen mittels U.

Expansionsgesetz: kann man leicht semantisch überprüfen.
Haben wir damals auch im (G)NBA kodiert.
Allerdings jetzt einfacher mit ABA modellierbar!

Vorbetrachtungen

Übersetzung logischer Formeln in alternierende Automaten ist oft einfacher als in nichtdeterministische Automaten.

Hier am Beispiel LTL $\rightarrow$ ABA

Erinnerung an LTL: $\varphi ::= x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi\, U\, \varphi$
mit $x \in$ AV (Aussagenvariablen)

$s, i \models \varphi\, U\, \psi$, falls $s, j \models \psi$ für ein $j \geqslant i$
und $s, k \models \varphi$ für alle k mit $i \leqslant k < j$

Vorbetrachtungen
Übersetzung logischer Formeln in alternierende Automaten ist oft einfacher als in nichtdeterministische Automaten.
Hier am Beispiel LTL → ABA
Erinnerung an LTL: $\varphi ::= x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi\, U\, \varphi$
mit $x \in$ AV (Aussagenvariablen)
$s, i \models \varphi\, U\, \psi$, falls $s, j \models \psi$ für ein $j \geqslant i$
und $s, k \models \varphi$ für alle k mit $i \leqslant k < j$

8:59
Formeln hier als Grammatik angegeben.
- Variablen als x, nicht p (schon für Positionen vergeben).
- $\vee$ ist eigentlich überflüssig, kann man aber gleich ganz bequem behandeln.
- F, G sind nur Abkürzungen mittels U.

Expansionsgesetz: kann man leicht semantisch überprüfen.
Haben wir damals auch im (G)NBA kodiert.
Allerdings jetzt einfacher mit ABA modellierbar!

Vorbetrachtungen

Übersetzung logischer Formeln in alternierende Automaten
ist oft einfacher als in nichtdeterministische Automaten.

Hier am Beispiel LTL $\to$ ABA

Erinnerung an LTL: $\varphi ::= x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi\, U\, \varphi$
mit $x \in$ AV (Aussagenvariablen)

$s, i \models \varphi\, U\, \psi$, falls $s, j \models \psi$ für ein $j \geqslant i$
und $s, k \models \varphi$ für alle k mit $i \leqslant k < j$

$F\varphi \equiv (x \vee \neg x)\, U\, \varphi$

$G\varphi \equiv \neg F \neg \varphi$

Vorbetrachtungen
Übersetzung logischer Formeln in alternierende Automaten ist oft einfacher als in nichtdeterministische Automaten.
Hier am Beispiel LTL $\to$ ABA
Erinnerung an LTL: $\varphi ::= x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi\, U\, \varphi$ mit $x \in$ AV (Aussagenvariablen)
$s, i \models \varphi\, U\, \psi$, falls $s, j \models \psi$ für ein $j \geqslant i$ und $s, k \models \varphi$ für alle k mit $i \leqslant k < j$
$F\varphi \equiv (x \vee \neg x)\, U\, \varphi$
$G\varphi \equiv \neg F \neg \varphi$

8:59

Formeln hier als Grammatik angegeben.

- Variablen als x, nicht p (schon für Positionen vergeben).
- $\vee$ ist eigentlich überflüssig, kann man aber gleich ganz bequem behandeln.
- F, G sind nur Abkürzungen mittels U.

Expansionsgesetz: kann man leicht semantisch überprüfen.
Haben wir damals auch im (G)NBA kodiert.
Allerdings jetzt einfacher mit ABA modellierbar!

Vorbetrachtungen

Übersetzung logischer Formeln in alternierende Automaten
ist oft einfacher als in nichtdeterministische Automaten.

Hier am Beispiel LTL $\to$ ABA

Erinnerung an LTL: $\varphi ::= x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi\, U\, \varphi$
mit $x \in$ AV (Aussagenvariablen)

$s, i \models \varphi\, U\, \psi$, falls $s, j \models \psi$ für ein $j \geqslant i$
und $s, k \models \varphi$ für alle k mit $i \leqslant k < j$

$F\varphi \equiv (x \vee \neg x)\, U\, \varphi$

$G\varphi \equiv \neg F \neg \varphi$

Expansionsgesetz:
$s, i \models \varphi\, U\, \psi$ **gdw.** $s, i \models \psi$ oder $(s, i \models \varphi$ und $s, i{+}1 \models \varphi\, U\, \psi)$

Vorbetrachtungen
Übersetzung logischer Formeln in alternierende Automaten ist oft einfacher als in nichtdeterministische Automaten.
Hier am Beispiel LTL $\to$ ABA
Erinnerung an LTL: $\varphi ::= x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi\, U\, \varphi$ mit $x \in$ AV (Aussagenvariablen)
$s, i \models \varphi\, U\, \psi$, falls $s, j \models \psi$ für ein $j \geqslant i$ und $s, k \models \varphi$ für alle k mit $i \leqslant k < j$
$F\varphi \equiv (x \vee \neg x)\, U\, \varphi$
$G\varphi \equiv \neg F \neg \varphi$
Expansionsgesetz:
$s, i \models \varphi\, U\, \psi$ gdw. $s, i \models \psi$ oder $(s, i \models \varphi$ und $s, i{+}1 \models \varphi\, U\, \psi)$

8:59

Formeln hier als Grammatik angegeben.

- Variablen als x, nicht p (schon für Positionen vergeben).
- $\vee$ ist eigentlich überflüssig, kann man aber gleich ganz bequem behandeln.
- F, G sind nur Abkürzungen mittels U.

Expansionsgesetz: kann man leicht semantisch überprüfen.
Haben wir damals auch im (G)NBA kodiert.
Allerdings jetzt einfacher mit ABA modellierbar!

Intuitionen der Konstruktion

Seien φ eine LTL-Formel und ψ eine beliebige Teilformel.

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

$$\mathbf{cl}(\varphi) = \{\psi, \sim\!\psi \mid \psi \text{ ist Teilformel von } \varphi\}$$

Intuitionen der Konstruktion
Seien φ eine LTL-Formel und ψ eine beliebige Teilformel.
$\sim\!\psi = \begin{cases} \vartheta & \text{falls } \psi = \neg\vartheta \\ \neg\psi & \text{sonst} \end{cases}$
$\mathbf{cl}(\varphi) = \{\psi, \sim\!\psi \mid \psi \text{ ist Teilformel von } \varphi\}$

9:03

„Negation der PBF“: geht natürlich nicht, weil PBFs keine ¬ haben. Gleich mehr.

Intuitionen der Konstruktion

Seien φ eine LTL-Formel und ψ eine beliebige Teilformel.

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

$\mathbf{cl}(\varphi) = \{\psi, \sim\!\psi \mid \psi \text{ ist Teilformel von } \varphi\}$

Bestandteile des ABA $\mathcal{A}_\varphi$

- Eingabealphabet: $\Sigma = 2^{\mathsf{AV}}$ wie gehabt
- Zustände: für jede Formel $\psi \in \mathsf{cl}(\varphi)$ ein q_ψ; Startzustand q_φ

Intuitionen der Konstruktion
Seien φ eine LTL-Formel und ψ eine beliebige Teilformel.
$\sim\!\psi = \begin{cases} \vartheta & \text{falls } \psi = \neg\vartheta \\ \neg\psi & \text{sonst} \end{cases}$
$\mathbf{cl}(\varphi) = \{\psi, \sim\!\psi \mid \psi \text{ ist Teilformel von } \varphi\}$
Bestandteile des ABA $\mathcal{A}_\varphi$
- Eingabealphabet: $\Sigma = 2^{\mathsf{AV}}$ wie gehabt
- Zustände: für jede Formel $\psi \in \mathsf{cl}(\varphi)$ ein q_ψ; Startzustand q_φ

9:03

„Negation der PBF“: geht natürlich nicht, weil PBFs keine $\neg$ haben. Gleich mehr.

Intuitionen der Konstruktion

Seien φ eine LTL-Formel und ψ eine beliebige Teilformel.

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

$\mathbf{cl}(\boldsymbol{\varphi}) = \{\psi, {\sim}\psi \mid \psi \text{ ist Teilformel von } \varphi\}$

Bestandteile des ABA $\mathcal{A}_\varphi$

- Eingabealphabet: $\Sigma = 2^{\mathsf{AV}}$ wie gehabt
- Zustände: für jede Formel $\psi \in \mathrm{cl}(\varphi)$ ein q_ψ; Startzustand q_φ
- Übergänge:
  - für $\wedge, \vee$: mittels PBF
  - für $\neg$: per „Negation“ der PBF
  - für $X\psi$: schicke q_ψ zur nächsten Position
  - für U: per Expansionsgesetz

9:03

„Negation der PBF“: geht natürlich nicht, weil PBFs keine $\neg$ haben.
Gleich mehr.

Intuitionen der Konstruktion

Seien φ eine LTL-Formel und ψ eine beliebige Teilformel.

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

$\mathbf{cl}(\varphi) = \{\psi, \sim\!\psi \mid \psi \text{ ist Teilformel von } \varphi\}$

Bestandteile des ABA $\mathcal{A}_\varphi$

- Eingabealphabet: $\Sigma = 2^{\mathsf{AV}}$ wie gehabt
- Zustände: für jede Formel $\psi \in \mathsf{cl}(\varphi)$ ein q_ψ; Startzustand q_φ
- Übergänge:
  - für $\wedge, \vee$: mittels PBF
  - für $\neg$: per „Negation“ der PBF
  - für $X\psi$: schicke q_ψ zur nächsten Position
  - für U: per Expansionsgesetz
- F verhindert unendliches „Aufschieben“ von U-Teilformeln!

Intuitionen der Konstruktion
Seien φ eine LTL-Formel und ψ eine beliebige Teilformel.
$\sim\!\psi = \begin{cases} \vartheta & \text{falls } \psi = \neg\vartheta \\ \neg\psi & \text{sonst} \end{cases}$
$\mathsf{cl}(\varphi) = \{\psi, \sim\!\psi \mid \psi \text{ ist Teilformel von } \varphi\}$
Bestandteile des ABA $\mathcal{A}_\varphi$
- Eingabealphabet: $\Sigma = 2^{\mathsf{AV}}$ wie gehabt
- Zustände: für jede Formel $\psi \in \mathsf{cl}(\varphi)$ ein q_ψ; Startzustand q_φ
- Übergänge:
  - für $\wedge, \vee$: mittels PBF
  - für $\neg$: per „Negation“ der PBF
  - für $X\psi$: schicke q_ψ zur nächsten Position
  - für U: per Expansionsgesetz
- F verhindert unendliches „Aufschieben“ von U-Teilformeln!

9:03

„Negation der PBF“: geht natürlich nicht, weil PBFs keine $\neg$ haben. Gleich mehr.

„Negation von PBFs“

Idee: Nutzen stattdessen Dualität von $\wedge, \vee$ (de Morgan), um Negation nach innen zu ziehen.

Negation eines Atoms q_ψ ist dann $q_{\sim\psi}$.

Genauer: mittels Operator $\overline{}$ wie folgt:

$$\overline{\zeta_1 \wedge \zeta_2} = \overline{\zeta_1} \vee \overline{\zeta_2}$$
$$\overline{\zeta_1 \vee \zeta_2} = \overline{\zeta_1} \wedge \overline{\zeta_2}$$
$$\overline{q_\psi} = q_{\sim\psi}$$
$$\overline{1} = 0$$
$$\overline{0} = 1$$

„Negation von PBFs“

Idee: Nutzen stattdessen Dualität von $\wedge, \vee$ (de Morgan), um Negation nach innen zu ziehen.
Negation eines Atoms q_ψ ist dann $q_{\sim\psi}$.

Genauer: mittels Operator $\overline{}$ wie folgt:

$\overline{\zeta_1 \wedge \zeta_2} = \overline{\zeta_1} \vee \overline{\zeta_2}$
$\overline{\zeta_1 \vee \zeta_2} = \overline{\zeta_1} \wedge \overline{\zeta_2}$
$\overline{q_\psi} = q_{\sim\psi}$
$\overline{1} = 0$
$\overline{0} = 1$

9:07

Achtung: wir wollen *nur PBFs* „negieren“, nicht LTL-Formeln.
Deshalb brauchen wir nur Fälle für $\wedge$, $\vee$ und Atome.

Aussagenvariablen sind aber hier immer nur die q_ψ;
und die können wir negieren, indem wir ψ negieren.

Konstruktion des ABA

- $Q = \{q_\psi \mid \psi \in \mathrm{cl}(\varphi)\}, \quad q_I = q_\varphi$
- $\Sigma = 2^{\mathrm{AV}}$
- $\delta : Q \times \Sigma \to \mathrm{B}^+(Q)$ wie folgt:

$$\delta(q_x, a) = \begin{cases} 1 & \text{falls } x \in a \\ 0 & \text{sonst} \end{cases}$$
$$\delta(q_{\sim\psi}, a) = \overline{\delta(q_\psi, a)}$$
$$\delta(q_{\psi\wedge\vartheta}, a) = \delta(q_\psi, a) \wedge \delta(q_\vartheta, a)$$
$$\delta(q_{\psi\vee\vartheta}, a) = \delta(q_\psi, a) \vee \delta(q_\vartheta, a)$$
$$\delta(q_{X\psi}, a) = q_\psi$$
$$\delta(q_{\psi U\vartheta}, a) = \delta(q_\vartheta, a) \vee \big(\delta(q_\psi, a) \wedge q_{\psi U\vartheta}\big)$$

- $F = \{q_{\neg(\psi U \vartheta)} \mid \neg(\psi\ U\ \vartheta) \in \mathrm{cl}(\varphi)\}$ **T 5.4**

Konstruktion des ABA
- $Q = \{q_\psi \mid \psi \in \mathrm{cl}(\varphi)\}, \quad q_I = q_\varphi$
- $\Sigma = 2^{\mathrm{AV}}$
- $\delta : Q \times \Sigma \to \mathrm{B}^+(Q)$ wie folgt:
$$\delta(q_x, a) = \begin{cases} 1 & \text{falls } x \in a \\ 0 & \text{sonst} \end{cases}$$
$$\delta(q_{\sim\psi}, a) = \overline{\delta(q_\psi, a)}$$
$$\delta(q_{\psi\wedge\vartheta}, a) = \delta(q_\psi, a) \wedge \delta(q_\vartheta, a)$$
$$\delta(q_{\psi\vee\vartheta}, a) = \delta(q_\psi, a) \vee \delta(q_\vartheta, a)$$
$$\delta(q_{X\psi}, a) = q_\psi$$
$$\delta(q_{\psi U\vartheta}, a) = \delta(q_\vartheta, a) \vee \big(\delta(q_\psi, a) \wedge q_{\psi U\vartheta}\big)$$
- $F = \{q_{\neg(\psi U \vartheta)} \mid \neg(\psi\ U\ \vartheta) \in \mathrm{cl}(\varphi)\}$ T 5.4

9:09

Erklären, 5min Pause, dann Beispiel vorrechnen $\to$ bis 9:30 ?

Man kann natürlich für die Zustände direkt ψ statt q_ψ schreiben.
Das sorgt aber für Verwirrung in der Definition von $\overline{}$,
weil man bei $\wedge, \vee$ nicht mehr sieht, ob PBFs oder LTL-Formeln verknüpft werden.

Vergleich mit Konstruktion LTL $\longrightarrow$ (G)NBA aus Teil 3

Auffällige Unterschiede

- ABA hat linear viele Zustände, GNBA exponentiell viele.
- Hier wird die Bedeutung **aller** Operatoren in δ kodiert.

Gemeinsamkeiten

- Beide Konstruktionen verwenden das Expansionsgesetz.
- Beide Akzeptanzbedingungen verfolgen denselben Zweck: verbieten, die Erfüllung von U-Formeln ∞ weit hinauszuzögern.

1. Punkt bedeutet natürlich, dass es zu einem ABA im Allg. keinen polynomiell großen äquivalenten NBA geben kann.

Vergleich mit Konstruktion LTL $\longrightarrow$ (G)NBA aus Teil 3

Auffällige Unterschiede
- ABA hat linear viele Zustände, GNBA exponentiell viele.
- Hier wird die Bedeutung aller Operatoren in δ kodiert.

Gemeinsamkeiten
- Beide Konstruktionen verwenden das Expansionsgesetz.
- Beide Akzeptanzbedingungen verfolgen denselben Zweck: verbieten, die Erfüllung von U-Formeln ∞ weit hinauszuzögern.

1. Punkt bedeutet natürlich, dass es zu einem ABA im Allg. keinen polynomiell großen äquivalenten NBA geben kann.

9:30

Letzter Satz: wenn ABA $\to$ NBA wieder gemacht wird, dann „Mehr dazu später“ in Folie wieder einkommentieren.

Und nun ...

Und nun ...

1. Einführung und Grundbegriffe
2. Von LTL zu alternierenden Automaten
3. Komplementierung

Abschluss unter Komplement

... ist für ABA-erkennbare Sprachen besonders leicht zu zeigen.

Für eine PBF φ definieren wir **dual(φ)**
als die PBF, die durch „Umdrehen" von $\wedge$ und $\vee$ entsteht,
z. B.: $\text{dual}\big((q_1 \wedge q_2) \vee q_3\big) = (q_1 \vee q_2) \wedge q_3$

Wir betrachten zur weiteren Erleichterung jetzt AMAs
(alternierende **Muller**-Aut., Akzeptanzkomp. $\mathcal{F} \subseteq 2^Q$ wie gehabt)

Abschluss unter Komplement
... ist für ABA-erkennbare Sprachen besonders leicht zu zeigen.
Für eine PBF φ definieren wir dual(φ)
als die PBF, die durch „Umdrehen" von $\wedge$ und $\vee$ entsteht,
z. B.: $\text{dual}((q_1 \wedge q_2) \vee q_3) = (q_1 \vee q_2) \wedge q_3$
Wir betrachten zur weiteren Erleichterung jetzt AMAs
(alternierende Muller-Aut., Akzeptanzkomp. $\mathcal{F} \subseteq 2^Q$ wie gehabt)

9:31
Grund für Muller:
Akzeptanzbedingung des Komplementautomaten wird einfacher.

Abschluss unter Komplement

Satz 5.5

Die Klasse der AMA-erkennbaren ω-Sprachen ist unter Komplement abgeschlossen.

Abschluss unter Komplement
Satz 5.5
Die Klasse der AMA-erkennbaren ω-Sprachen ist unter Komplement abgeschlossen.

9:32 bis 9:45 (nur Bsp. und kein Beweis)
Konstruktion: vergleiche mit Safra-Konstruktion

Beweis: braucht Spiele (sehr ähnlich zu denen aus Kap. 4)
und alternative Def. alternierender Automaten $\rightsquigarrow$ hier nicht.

TODO: Wenn Satz bewiesen werden soll, dann Beweis ganz neu ausarbeiten; man muss wohl Alternierung mittels existenzieller und universeller Zust. definieren;
siehe Notizen zwischen T5.5 und T5.6.

Abschluss unter Komplement

Satz 5.5

Die Klasse der AMA-erkennbaren ω-Sprachen ist unter Komplement abgeschlossen.

Beweis. Sei $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, \mathcal{F})$ ein AMA.

Abschluss unter Komplement
Satz 5.5
Die Klasse der AMA-erkennbaren ω-Sprachen ist unter Komplement abgeschlossen.
Beweis. Sei $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, \mathcal{F})$ ein AMA.

9:32 bis 9:45 (nur Bsp. und kein Beweis)
Konstruktion: vergleiche mit Safra-Konstruktion

Beweis: braucht Spiele (sehr ähnlich zu denen aus Kap. 4) und alternative Def. alternierender Automaten $\rightsquigarrow$ hier nicht.

TODO: Wenn Satz bewiesen werden soll, dann Beweis ganz neu ausarbeiten; man muss wohl Alternierung mittels existenzieller und universeller Zust. definieren;
siehe Notizen zwischen T5.5 und T5.6.

Abschluss unter Komplement

Satz 5.5

Die Klasse der AMA-erkennbaren ω-Sprachen ist unter Komplement abgeschlossen.

Beweis. Sei $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, \mathcal{F})$ ein AMA.

Konstruiere AMA $\mathcal{A}' = (Q, \Sigma, \delta', \{q_I\}, \mathcal{F}')$ wie folgt:

- Für alle $q \in Q$ und $a \in \Sigma$, setze $\delta'(q, a) = \text{dual}(\delta(q, a))$.

Abschluss unter Komplement

Satz 5.5
Die Klasse der AMA-erkennbaren ω-Sprachen ist unter Komplement abgeschlossen.

Beweis. Sei $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, \mathcal{F})$ ein AMA.
Konstruiere AMA $\mathcal{A}' = (Q, \Sigma, \delta', \{q_I\}, \mathcal{F}')$ wie folgt:
- Für alle $q \in Q$ und $a \in \Sigma$, setze $\delta'(q, a) = \text{dual}(\delta(q, a))$.

9:32 bis 9:45 (nur Bsp. und kein Beweis)
Konstruktion: vergleiche mit Safra-Konstruktion

Beweis: braucht Spiele (sehr ähnlich zu denen aus Kap. 4)
und alternative Def. alternierender Automaten $\leadsto$ hier nicht.

TODO: Wenn Satz bewiesen werden soll, dann Beweis ganz neu ausarbeiten; man muss wohl Alternierung mittels existenzieller und universeller Zust. definieren;
siehe Notizen zwischen T5.5 und T5.6.

Abschluss unter Komplement

Satz 5.5

Die Klasse der AMA-erkennbaren ω-Sprachen ist unter Komplement abgeschlossen.

Beweis. Sei $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, \mathcal{F})$ ein AMA.

Konstruiere AMA $\mathcal{A}' = (Q, \Sigma, \delta', \{q_I\}, \mathcal{F}')$ wie folgt:

- Für alle $q \in Q$ und $a \in \Sigma$, setze $\delta'(q, a) = \text{dual}(\delta(q, a))$.
- $\mathcal{F}' = 2^Q \setminus \mathcal{F}$ **T 5.5**

Abschluss unter Komplement
Satz 5.5
Die Klasse der AMA-erkennbaren ω-Sprachen ist unter Komplement abgeschlossen.
Beweis. Sei $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, \mathcal{F})$ ein AMA.
Konstruiere AMA $\mathcal{A}' = (Q, \Sigma, \delta', \{q_I\}, \mathcal{F}')$ wie folgt:
Für alle $q \in Q$ und $a \in \Sigma$, setze $\delta'(q, a) = \text{dual}(\delta(q, a))$.
$\mathcal{F}' = 2^Q \setminus \mathcal{F}$ T 5.5

9:32 bis 9:45 (nur Bsp. und kein Beweis)
Konstruktion: vergleiche mit Safra-Konstruktion

Beweis: braucht Spiele (sehr ähnlich zu denen aus Kap. 4) und alternative Def. alternierender Automaten $\rightsquigarrow$ hier nicht.

TODO: Wenn Satz bewiesen werden soll, dann Beweis ganz neu ausarbeiten; man muss wohl Alternierung mittels existenzieller und universeller Zust. definieren;
siehe Notizen zwischen T5.5 und T5.6.

Abschluss unter Komplement

Satz 5.5

Die Klasse der AMA-erkennbaren ω-Sprachen ist unter Komplement abgeschlossen.

Beweis. Sei $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, \mathcal{F})$ ein AMA.

Konstruiere AMA $\mathcal{A}' = (Q, \Sigma, \delta', \{q_I\}, \mathcal{F}')$ wie folgt:

- Für alle $q \in Q$ und $a \in \Sigma$, setze $\delta'(q, a) = \mathrm{dual}(\delta(q, a))$.
- $\mathcal{F}' = 2^Q \setminus \mathcal{F}$ **T 5.5**

Dann gilt: $L_\omega(\mathcal{A}') = \overline{L_\omega(\mathcal{A})}$ (Beweis mittels Spielen) □

2019-02-01

Teil 5: Alternierung
└Komplementierung
└Abschluss unter Komplement

Abschluss unter Komplement

Satz 5.5
Die Klasse der AMA-erkennbaren ω-Sprachen ist unter Komplement abgeschlossen.

Beweis. Sei $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, \mathcal{F})$ ein AMA.
Konstruiere AMA $\mathcal{A}' = (Q, \Sigma, \delta', \{q_I\}, \mathcal{F}')$ wie folgt:
- Für alle $q \in Q$ und $a \in \Sigma$, setze $\delta'(q, a) = \mathrm{dual}(\delta(q, a))$.
- $\mathcal{F}' = 2^Q \setminus \mathcal{F}$ T 5.5

Dann gilt: $L_\omega(\mathcal{A}') = \overline{L_\omega(\mathcal{A})}$ (Beweis mittels Spielen) □

9:32 bis 9:45 (nur Bsp. und kein Beweis)
Konstruktion: vergleiche mit Safra-Konstruktion

Beweis: braucht Spiele (sehr ähnlich zu denen aus Kap. 4)
und alternative Def. alternierender Automaten $\rightsquigarrow$ hier nicht.

TODO: Wenn Satz bewiesen werden soll, dann Beweis ganz neu ausarbeiten;
man muss wohl Alternierung mittels existenzieller und universeller Zust. definieren;
siehe Notizen zwischen T5.5 und T5.6.

Abschluss unter Komplement

Satz 5.5

Die Klasse der AMA-erkennbaren ω-Sprachen ist unter Komplement abgeschlossen.

Beweis. Sei $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, \mathcal{F})$ ein AMA.

Konstruiere AMA $\mathcal{A}' = (Q, \Sigma, \delta', \{q_I\}, \mathcal{F}')$ wie folgt:

- Für alle $q \in Q$ und $a \in \Sigma$, setze $\delta'(q, a) = \mathrm{dual}(\delta(q, a))$.
- $\mathcal{F}' = 2^Q \setminus \mathcal{F}$ **T 5.5**

Dann gilt: $L_\omega(\mathcal{A}') = \overline{L_\omega(\mathcal{A})}$ (Beweis mittels Spielen) $\square$

Insbesondere ist $\mathcal{A}'$ (bis auf $\mathcal{F}'$) nicht größer als $\mathcal{A}$!

Abschluss unter Komplement
Satz 5.5
Die Klasse der AMA-erkennbaren ω-Sprachen ist unter Komplement abgeschlossen.
Beweis. Sei $\mathcal{A} = (Q, \Sigma, \delta, \{q_I\}, \mathcal{F})$ ein AMA.
Konstruiere AMA $\mathcal{A}' = (Q, \Sigma, \delta', \{q_I\}, \mathcal{F}')$ wie folgt:
Für alle $q \in Q$ und $a \in \Sigma$, setze $\delta'(q, a) = \mathrm{dual}(\delta(q, a))$.
$\mathcal{F}' = 2^Q \setminus \mathcal{F}$ T 5.5
Dann gilt: $L_\omega(\mathcal{A}') = \overline{L_\omega(\mathcal{A})}$ (Beweis mittels Spielen) $\square$
Insbesondere ist $\mathcal{A}'$ (bis auf $\mathcal{F}'$) nicht größer als $\mathcal{A}$!

9:32 bis 9:45 (nur Bsp. und kein Beweis)
Konstruktion: vergleiche mit Safra-Konstruktion

Beweis: braucht Spiele (sehr ähnlich zu denen aus Kap. 4) und alternative Def. alternierender Automaten $\rightsquigarrow$ hier nicht.

TODO: Wenn Satz bewiesen werden soll, dann Beweis ganz neu ausarbeiten; man muss wohl Alternierung mittels existenzieller und universeller Zust. definieren;
siehe Notizen zwischen T5.5 und T5.6.

Alternierende vs. nichtdeterministische Automaten

Satz 5.6 (Miyano & Hayashi 1984)

Für jeden ABA $\mathcal{A}$ gibt es einen NBA $\mathcal{A}'$ mit $L_\omega(\mathcal{A}) = L_\omega(\mathcal{A}')$.

Alternierende und nichtdeterministische Büchi-Automaten sind also **gleichmächtig**.

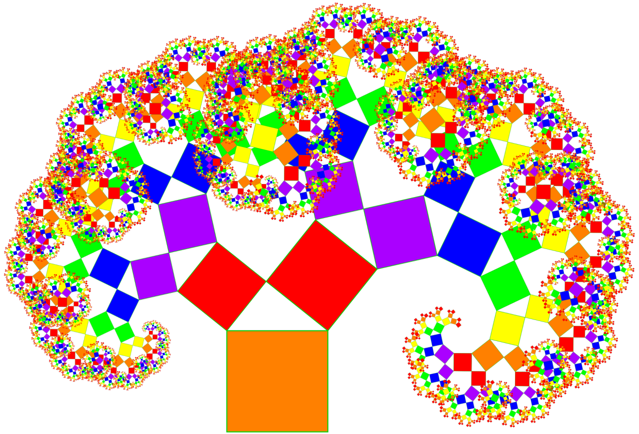
Beweisskizze: Siehe Folien aus dem letzten Jahr
`http://tinyurl.com/ws1718-automaten`

Alternierende vs. nichtdeterministische Automaten
Satz 5.6 (Miyano & Hayashi 1984)
Für jeden ABA $\mathcal{A}$ gibt es einen NBA $\mathcal{A}'$ mit $L_\omega(\mathcal{A}) = L_\omega(\mathcal{A}')$.
Alternierende und nichtdeterministische Büchi-Automaten sind also gleichmächtig.
Beweisskizze: Siehe Folien aus dem letzten Jahr
http://tinyurl.com/ws1718-automaten

9:45

Noch ein wichtiges Resultat, aber ohne Beweis

Fast fertig für dieses Semester . . .



Pythagoras-Baum. Quelle: Wikipedia, User Gjacquenot (Lizenz CC BY-SA 3.0)

Danke für Eure Aufmerksamkeit!

9:46

- Literatur
- Eval.
- Prüfungshinweise?

Literatur für diesen Teil

Bernd Finkbeiner.
Automata, Games, and Verification.
Vorlesungsskript, Universität des Saarlandes, SoSe 2015.
Kap. 8: Alternating Büchi Automata.
https://www.react.uni-saarland.de/teaching/automata-games-verification-15/lecture-notes.html
https://www.react.uni-saarland.de/teaching/automata-games-verification-15/downloads/notes.pdf

Literatur für diesen Teil
Bernd Finkbeiner.
Automata, Games, and Verification.
Vorlesungsskript, Universität des Saarlandes, SoSe 2015.
Kap. 8: Alternating Büchi Automata.
https://www.react.uni-saarland.de/teaching/automata-games-verification-15/lecture-notes.html
https://www.react.uni-saarland.de/teaching/automata-games-verification-15/downloads/notes.pdf

2019-02-01

Teil 5: Alternierung