Automaten auf unendlichen Wörtern

Michael Neuendorf

3. Februar 2005

Dieses Skript habe ich für mich als Lernhilfe bei der Vorbereitung auf die Diplomprüfungen geschrieben in der Hoffnung, daß es mir dann hilfreich sein wird (sobald ich denn endlich mal soweit sein werde); da ich verschiedentlich darauf angesprochen wurde, ob ich dieses Skript denn auch öffentlich zur Verfügung stellen würde, entspreche hiermit diesem Wunsch.
Ich werde die jeweils aktuelle Version in Abständen wieder ins Netz stellen, sodaß sie mit etwas Abstand zur Vorlesung verfügbar sein wird.
Solltet ihr Fehler jeglicher Art (z.B. Tippfehler oder auch inhaltliche Fehler) finden, so schickt mir doch bitte eine kurze eMail mit Abschnittname und Fehlerbeschreibung an die folgende Adresse: *michael.neuendorf@rwth-aachen.de*

Michael Neuendorf

Motivation der Vorlesung:

1. Erweiterung der Automaten- und Sprachentheorie auf den Bereich der unendlichen Wörter
2. Lösung logischer Entscheidungsprobleme (vor allem im Bereich der Arithmetik)
3. Verifikation nichtterminierender Programme (Model Checking)
4. Constraint Satisfaction über reellen Zahlen

1 Büchi-Automaten und reguläre ω-Sprachen

1.1 Elementare Sachverhalte

Ein (nichtdeterministischer) Büchi-Automat (NBA) hat die Form $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$ mit endlicher Zustandsmenge Q, Eingabealphabet Σ, Anfangszustand $q_0 \in Q$, Transitionsrelation $\Delta \subseteq Q \times \Sigma \times Q$, Endzustandsmenge $F \subseteq Q$. Sei $\alpha = \alpha(0)\alpha(1)\ldots$ unendliches Wort (ω-Wort) über Σ, d.h. $\alpha(i) \in \Sigma$. Ein Lauf von $\mathfrak{A}$ auf α ist eine Folge $\rho = \rho(0)\rho(1)\ldots$ über Q mit $rho(0) = \rho_0$, $(\rho(i), \alpha(i), \rho(i+1)) \in \Delta$ für $i \geq 0$.
Ein Lauf ρ heißt (Büchi-) akzeptierend, falls $\rho(i) \in F$ für unendlich viele $i[\exists^\omega i \rho(i) \in F$. $\mathfrak{A}$ akzeptiert α, gdw. ein akzeptierender Lauf von $\mathfrak{A}$ auf α existiert. $L(\mathfrak{A}) = \{\alpha \in \Sigma^\omega | \mathfrak{A}$ akzeptiert $\alpha\}$, $\Sigma^\omega =$ Menge der ω-Wörter über Σ

Beispiel: $\mathfrak{A}$ akzeptiert $\alpha \Leftrightarrow \alpha$ hat die Form $uaaa\ldots$ mit $u \in \Sigma^*$ oder α hat die Form $uabab\ldots$ mit $u \in \Sigma^*$.

Deterministischer Büchi-Automat In einem deterministischen Büchi-Automaten (DBA) ist Δ ersetzt durch die Transitionsfunktion $\delta : Q \times \Sigma \to Q$. In einem Lauf ρ ergibt sich jeweils $\rho(i+1) = \delta(\rho(i), \alpha(i))$ (außerdem $\rho(0) = q_0$). Das Akzeptieren des DBA geschieht analog zum Akzeptieren eines NBA.
Bemerkung: Jeder DBA ist (auffaßbar als) NBA; setze $\Delta = \{(p, a, q) | \delta(p, a) = q\}$.

Satz: Sei $L_1 = \{\alpha \in \{a, b\}^\omega | \alpha = uaaa...$ mit $u \in \{a, b\}^*\}$. L_1 ist NBA-erkennbar, aber nicht DBA erkennbar.

Beweis: L_1 NBA-erkennbar, wie zuvor. Annahme: DBA $\mathfrak{A}_1$ erkennt L_1. $\mathfrak{A}_1$ akzeptiert $aaa\ldots$, $\mathfrak{A}_1 = (Q, \{a, b\}, q_0, \delta, F)$ nimmt unendlich oft F-Zustand an, erstmalig etwa nach Präfig a^{n_0}. $\mathfrak{A}_1$ akzeptiert $a^{n_0}baaa\ldots$, erster Endzustand nach b etwa nach $a^{n_0}ba^{n_1}$. $\mathfrak{A}_1$ akzeptiert $a^{n_0}ba^{n_1}baaa\ldots$, erhalte $\beta = a^{n_0}ba^{n_1}ba^{n_2}b\ldots$, $\mathfrak{A}_1$ besucht Endzustand vor jedem b, aber $\beta \notin L_1$, Widerspruch zur Wahl von $\mathfrak{A}_1$.

Sprachoperationen

- $U, V, W, \ldots$ stehen für Mengen endlicher Wörter ($\subseteq \Sigma^*$)
- $K, L, M, \ldots$ stehen für Mengen unendlicher Wörter ($\subseteq \Sigma^\omega$)
- $\alpha = u_0u_1u_2, \ldots$ mit $u_i \in U$ für $i \geq 0$
- $\lim U = \{\alpha \in \Sigma^\omega | \alpha$ hat unendlich viele Präfixe in $U\}$
- $\exists^\omega i \alpha(0)\ldots\alpha(i) \in U$
- Entsprechend wird eingeführt für $U \subseteq \Sigma^*, L \subseteq \Sigma^\omega$:

  $$U \cdot L = \{\alpha \in \Sigma^\omega | \alpha) u\beta \text{ mit } u \in U, \beta \in L \text{ geeignet}\}$$

- Beispiel: $u = abba^* + aa$; Beispielwörter in U^ω: $aaaaaaaa\ldots, aaabbabbaabbaaabbaaa\ldots$
  - Behauptung: $\alpha in U^\omega \Leftrightarrow \alpha$ hat ungeradzahlig viele a vor erstem b (sofern vorhanden) und b immer paarweise mit vorangehendem a.
  - $\lim U = \{abbaaaa\ldots\}$.
  - Präfixe in U sind $abb, abba, abbaa, \ldots$.
- Ziel:
  1. L DBA-erkennbar $\Leftrightarrow L = \lim U, U$ regulär
  2. L NBA-erkennbar $\Leftrightarrow L = \bigcup_{i=1}^n U_i \cdot V_i^\omega$; U_i, V_i regulär

Satz:

1. $L \subseteq \Sigma^\omega$ ist durch einen DBA erkennbar gdw. $L = \lim U$ für ein U regulär.
2. $L \subseteq \Sigma^\omega$ ist NBA-erkennbar gdw. $L = \bigcup_{i=1}^{n} U_i \cdot V_i^\omega$ für geeignete $U_i, V_i \subseteq \Sigma^*$ regulär.

Beweis:

1. - $\Rightarrow$: Sei $\mathfrak{B} = (Q, \Sigma, q_0, \delta, F)$ DBA, $U :=$ die durch $\mathfrak{B}$ als Standardautomaten-Sprachen (regulär!). Dann gilt: $\mathfrak{B}$ akzeptiert $\alpha \Leftrightarrow \exists^\omega i \ \delta^*(q_0, \alpha[0,i]) \in F \Leftrightarrow \exists^\omega i \alpha[0,i] \in U \Leftrightarrow \alpha \in \lim U$

   - $\Leftarrow$: Sei $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ Standard-DBA, der U erkenne. Zeige $\lim U$ DBA-erkennbar. Definiere DBA durch Übernahme von $\mathfrak{A}$ als Büchi-Automat. Dann gelten obige Äquivalenzen mit $\mathfrak{A}$ statt $\mathfrak{B}$.

2. - $\Rightarrow$: L werde erkannt durch NBA $\mathfrak{B} = (Q, \Sigma, q_0, \Delta, F)$. Für $p, q \in Q$ definiere $W(p,q) :=$ die durch $(Q, \Sigma, p, \Delta, \{q\})$ erkannte Sprache endlicher Wörter.
     Behauptung: $L = \bigcup_{q \in F} W(q_0, p) \cdot W(q,q)$, $W(p,q)$ regulär.
     - $\subseteq$: Sei $\alpha \in L$, d.h. durch $\mathfrak{B}$ akzeptiert, etwa mit Lauf der Form $\alpha = u_0 u_1 u_2 \ldots$ folgt $\alpha \in W(q_0, q) \cdot W(q,q)^\omega$
     - $\supseteq$: $A \in R.S. \curvearrowright$ für geeignetes $q \in F$ gilt $\alpha \in u_0 u_1 u_2 \ldots$ mit $u_0 \in W(q_0, q)$, $u_i \in W(q,q), \quad i > 0$. $\mathfrak{B}$ akzeptiert α, durch Kombinieren der Läufe über $u_0 \in W(q_0, q), u_i \in W(q,q)$
   - $\Leftarrow$: **Lemma:**
     a) $V \subseteq \Sigma^*$ regulär $\Rightarrow V^\omega$ NBA-erkennbar
     b) $U \subseteq \Sigma^*$ regulär, K NBA-erkennbar $\Rightarrow U \cdot K$ NBA-erkennbar
     c) K_1, K_2 NBA-erkennbar $\Rightarrow K_1 \cup K_2$ NBA-erkennbar

     Beweis:
     a) Sei V erkannt durch EA $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$. Modifiziere $\mathfrak{A}$ durch Einfügen von Rücktransitionen; dieses ist korrekt, falls keine andren Transitionen nach q_0 führen. Hierzu, wenn nötig, Vorbehandlung von $\mathfrak{A}$: Einführung eines neuen Anfangszustandes
     b) Gegeben EA $\mathfrak{A} = (Q_\mathfrak{A}, \Sigma, q_{0\mathfrak{A}} \Delta_\mathfrak{A}, F_\mathfrak{A})$ für U und $\mathfrak{B} = (Q_\mathfrak{B}, \Sigma, q_{0\mathfrak{B}} \Delta_\mathfrak{B}, F_\mathfrak{B})$ für K. Suche NBA für $U \cdot K$. NBA $\mathfrak{C}$ hat Zustandsmenge $Q_\mathfrak{A} \cup Q_\mathfrak{B}$, $q_{0\mathfrak{A}}$ als Anfangszustand, als Transitionsmenge $\Delta_\mathfrak{A} \cup \Delta_\mathfrak{B} \cup \{(q, a, q_{0\mathfrak{B}}) | \exists (q, a, q') \in \Delta_\mathfrak{A}$ mit $q' \in F\} \cup \{(q_{0\mathfrak{A}}, b, p) | q_{0\mathfrak{A}} \in F_\mathfrak{A} (q_{0\mathfrak{C}}, b, p) \in \Delta_\mathfrak{C}\}$, Endzustandsmenge $F_\mathfrak{B}$.
     c) K_1 werde durch den NBA $\mathfrak{A}_1 = (Q_1, \Sigma, q_{01}, \Delta_1, F_1)$ erkannt, K_2 durch $\mathfrak{A}_2 = (Q_2, \Sigma, q_{02}, \Delta_2, F_2)$. Bilde Produktautomaten $\mathfrak{B} = (Q_1 \times Q_2, \Sigma, (q_{01}, q_{02}), \Delta, F_\cup)$. $((p_1, p_2) a, (q_1, q_2)) \in \Delta :\Leftrightarrow (p_1, a, q_1) \in \Delta_1 \wedge (p_2, a, q_2) \in \Delta_2$. $F_\cup = (F_1 \times Q_2) \cup (Q_1 \times F_2)$. Nötig ist eine Vorbehandlung von $\mathfrak{A}_1, \mathfrak{A}_2$: Vervollständigung der Transitionsrelation, sodaß für jeden Zustand und jeden Buchstaben eine Transition existiert; es genügt zu diesem Zwecke einen Senkenzustand hinzuzufügen.

Anmerkung:

- Der zweite Teil des Satzes motiviert reguläre Ausdrücke der Form $r_1 \cdot s_1^\omega + \ldots + r_n \cdot s_n^\omega$, wobei r_i, s_i jeweils Standardausdrücke sind (mit $+, \cdot, *$).
- Produkt in $r \cdot s^\omega$ und Summe in $r_1 s_1^\omega + r_2 s_2^\omega$ sind nötig.
  Beweis:
  - Summe: $L = \{a^\omega, b^\omega\}$ definierbar $a^\omega + b^\omega$. Annahme: Es genügt ein Ausdruck $r \cdot s^\omega$. In der Sprache zu s ist ein Wort a^i und ein Wort b^i enthalten. Es existiert ein Wort in r, etwa w; dann $w(uv)^\omega$ in Sprache zu $r \cdot s^\omega$, aber $w(uv)^\omega \notin L$.
  - Produkt: $L = \{ab^\omega\}$. Annahme: Es genügt s^ω. Dann gibt es in der Sprache zu s ein Wort u mit a und ein Wort v mit b. $(uv)^\omega$ gehört zu s, hat aber unendlich viele a.

Satz (Entscheidbarkeit des Nicht-Leerheitsproblemes für NBA): Es existiert ein Algorithmus, der zu gegebenem NBA $\mathfrak{A}$ entscheidet, ob $L(\mathfrak{A}) \neq \emptyset$.

Beweis: Teste für jeden der Endzustände, etwa für $q \in F$, ob in $\mathfrak{A}$ ein Pfad von q_0 nach q und ein nichtleerer Pfad von q nach q existieren.
Folgerung aus dem Darstellungssatz: L NBA-erkennbar, $L \neq \emptyset \Rightarrow L$ enthält ein schließlich periodisches ω-Wort (der Form $\alpha = uvvvv \ldots$)

2 Deterministische ω-Automaten

2.1 Mullerautomaten, Rabinautomaten

- Muller 1963: "Mullerautomten"
- McNaughton 1966: "Beweis der Äquivalenz zu NBAs", Muller-Schupp-Beweis hier
- Rabin $\sim$ 1968

Beispiel:

$L = \{\alpha \in \{a,b\}^\omega |$ in α kommt b nur endlich oft vor (also nur a von gewisser Stelle an) $\}$

Es ist bekannt, daß die Büchi-Bedingung nicht für den deterministischen Automaten reicht. Dazu ist eine Festlegung über das nur endlich häufige Besuchen von Zuständen nötig.

Ansatz: Spezifiziere die Menge aller Zustände, die im Lauf unendlich oft auftreten sollen.

Definition: Zu einem Lauf $\rho \in Q^\omega$ definiere $\mathrm{Inf}(\rho) := \{q \in Q | \exists^\omega i : \rho(i) = q\}$
Bemerkung: Jeder Lauf ρ läßt sich zerlegen als $\rho = u\rho'$ mit:

- In u treten auf Zustände auf aus $\mathrm{Inf}(\rho)$ und solche, die nur endlich oft auftreten,
- in ρ' treten genau die Zustände in $\mathrm{Inf}(\rho)$ immer wieder auf.

Definition (Muller-Automat) Ein Muller-Automat hat die Form $\mathfrak{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ mit $\mathcal{F} = \mathcal{P}(Q))$, etwa $\mathcal{F} = \{F_1, \ldots, F_k\}$, wobei $F_i \subseteq Q$, sonst wie Standard-DEA.
$\mathfrak{A}$ akzeptiert $\alpha \in \Sigma^\omega$, falls für den eindeutigen Lauf ρ auf α gilt: $\mathrm{Inf}(\rho) \in \mathcal{F}$, d.h. $\mathrm{Inf}(\rho) = F_i$ für ein $i \in \{1, \ldots, k\}$.
$L(\mathfrak{A}) = \{\alpha \in \Sigma^\omega | \mathfrak{A}$ akzeptiert $\alpha\}$; L Muller-erkennbar (genauer deterministisch Muller-erkennbar).
Bemerkung: Büchi-Bedingung für Lauf ρ besagt: $\mathrm{Inf}(\rho) \cap F \neq \emptyset$

Beispiel: $L = \{\alpha \in \{a,b,c\}^\omega | \exists^\omega i : \alpha(i) = a \rightarrow \exists^\omega i : \alpha(i) = b\}$ - "kommt a unendlich oft vor, so auch b unendlich oft ".

- Ansatz: Überprüfe in einer Menge $\mathrm{Inf}(\rho)$: $q_a \in \mathrm{Inf}(\rho) \rightarrow q_b \in \mathrm{Inf}(\rho)$.
- Umsetzung: Liste alle Mengen $P \in Q$ auf, sodaß gilt: $q_a \in P \rightarrow q_b \in P$
- P-Kandidaten: $\{q_a\}, \{q_b\}, \{q_c\}, \{q_a, q_b\}, \{q_b, q_c\}, \{q_a, q_c\}, \{q_a, q_b, q_c\}$, $q_a \notin P \vee q_b \in P$.
- Also nimm $\mathcal{F} = \{\{q_b\}, \{q_c\}, \{q_a, q_b\}, \{q_b, q_c\}, \{q_a, q_b, q_c\}\}$.

Satz: $L \subseteq \Sigma^*$ ist Muller-erkennbar genau dann, wenn L Boolesche Kombination (mit Komplement, Durchschnitt, Vereinigung) von deterministisch Büchi-erkennbaren ω-Sprachen.

Beweis:

- $\Rightarrow$: Gegeben ist ein Muller-Automat $\mathfrak{A} = (Q, \Sigma, q_9, \delta, \mathcal{F}$.
  $\mathfrak{A}$ akzeptiert α
  $\Leftrightarrow$ ein $F \in \mathcal{F}$ existiert, sodaß für den Lauf ρ auf α gilt: $\mathrm{Inf}(\rho) = F$
  $\Leftrightarrow \bigvee_{F\in\mathcal{F}} \begin{cases} q \in F & q \text{ wird unendlich oft besucht in } \rho \\ q \in Q \setminus F & q \text{ wird nicht unendlich oft besucht in } \rho \end{cases}$
  $\Leftrightarrow \bigvee_{F\in\mathcal{F}}(\bigwedge_{q\in F} \exists^\omega i : \rho(i) = q \wedge \bigvee_{q\in Q\setminus F} \neg\exists^\omega i : \rho(i) = q) \quad (*)$.
  Definiere zu $q \in Q$ einen deterministischen Büchi-Automaten $\mathfrak{A}_Q = (Q, \Sigma, q_0, \delta, \{q\})$; L_q sei die durch $\mathfrak{A}_q$ erkannte Sprache. **Beachte:** $\mathfrak{A}_q$ akzeptiert $\alpha \Leftrightarrow$ der eindeutige Lauf ρ auf α erfüllt: $\exists^\omega i : \rho(i) = q$.
  Also $(*) \Leftrightarrow \bigvee_{F\in\mathcal{F}}(\bigwedge_{q\in F} \alpha \in L_q \wedge \bigwedge_{q\in Q\setminus F} \alpha \notin L_q)$, $L(\mathfrak{A}) = \bigcup_{F\in\mathcal{F}}(\bigcap_{q\in F} L_q \cap \bigcap_{q\in Q\setminus F}(\Sigma^\omega \setminus L_q))$
- $\Leftarrow$: Zeige:

  1. L Büchi-erkennbar $\Rightarrow$ L Muller-erkennbar
  2. L Muller-erkennbar $\Rightarrow$ $\Sigma^\omega \setminus L$ Muller-erkennbar
  3. L_1, L_2 Muller-erkennbar $\Rightarrow$ $L_1 \cap L_2$ Muller-erkennbar

  - ad 1: Gegeben: Deterministischer Büchi-Automat $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$, $F \subseteq Q$. Definiere Muller-Automat $\mathfrak{A}' = (Q, \Sigma, q_0, \delta, F)$ mit $\mathcal{F} = \{P \subseteq Q | P \cap F \neq \emptyset\}$
  - ad 2: Zu $\mathfrak{A} = (Q, \Sigma, q, \delta, \mathcal{F})$ nimm $\bar{\mathfrak{A}} = (Q, \Sigma, q_0, \delta, \bar{\mathcal{F}})$ mit $\bar{\mathcal{F}} = \mathcal{P}(Q) \setminus \mathcal{F}$.
  - ad 3: Gegeben: $\mathfrak{A}_1 = (Q_1, \Sigma, q_{01}, \delta_1, \mathcal{F}_1)$, $\mathfrak{A}_2 = (Q_2, \Sigma, q_{02}, \delta_2, \mathcal{F}_2)$. Bilde Produktautomat $\mathfrak{A} = (Q_1 \times Q_2, \Sigma, (q_{01}, q_{02}), \delta, \mathcal{F})$ mit

    $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$ und
    $\{(p_1, q_2), \ldots, (p_m, q_m)\} \subseteq Q_1 \times Q_2 \in \mathcal{F}$
    $\Leftrightarrow \{p_1, \ldots, p_m\} \in \mathcal{F}_1 \wedge \{q_1, \ldots, q_m\} \in \mathcal{F}_2$

Satz: Ist L deterministisch Muller-erkennbar $\Rightarrow$ L Nichtdeterministisch Büchi-erkennbar.

Beweis:

- Gegeben: $\mathfrak{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$, etwa $\mathcal{F} = \{\mathcal{F}_1, \ldots, \mathcal{F}_k\}$
- Der nichtdeterministische Büchi-Automat rät die Stelle des Laufes ρ (von $\mathfrak{A}$), von der an nur Zustände aus $\mathrm{Inf}(\rho)$ auftreten, und rät dasjenige F_j, für das $F_j = \mathrm{Inf}(\rho)$ sein soll. Dann Aufsammeln besuchter Zustände, sofern diese in F_j sind, und Neuinitialisierung mit $\emptyset$, wenn F_j erreicht ist ("Akzeptierzustand").

Definition: $\mathfrak{B} = ((Q \times \mathcal{P}(Q) \times \{1, \ldots, k\}) \cup Q, \Sigma, q_0, \Delta, F)$ mit

$$\Delta = \begin{cases} (p, a, q), (p, a, (p, \emptyset, j)) & \text{falls } \delta(p, a) = q \quad (j \in \{1, \ldots, k\}) \\ ((p, P, j), a, (q, P \cup \{q\}, j)) & \text{falls } \delta(p, a) = q \quad P \cup \{q\} \subset F_j \\ ((p, P, j), a, (q, \emptyset, j)) & \text{falls } \delta(p, a) = q \quad P \cup \{q\} = F_j \end{cases}$$

und

$$F = \{(p, \emptyset, j) | p \in Q, j \in \{1, \ldots, k\}\}.$$

Rabin-Automat Rabin-Automat analog Muller-Automat, mit Akzeptierkomponente $\Omega = ((E_1, F_1), \ldots, (E_r, F_r))$. Akzeptieren durch Lauf ρ bedeutet:

$$\exists i \in \{1, \ldots, r\} : \mathrm{Inf}(\rho) \cap F_i \neq \emptyset \wedge \mathrm{Inf}(\rho) \cap E_i = \emptyset$$

Bemerkung: Jeder Rabin-Automat (mit Akzeptierkomponente Ω) ist äquivalent zu einer Muller-Automaten durch Übergang zu

$$\mathcal{F} = \{P \subset Q | \exists i (P \cap F_i \neq \emptyset \wedge P \cap E_i = \emptyset)\}$$

Satz (McNaughton) Zu jedem nichtdeterministischen Büchi-Automaten kann man einen äquivalenten deterministischen Rabin- (und somit auch Muller-) Automaten konstruieren: NBüchi → DRabin → DMuller → NBüchi

Beweis: NBüchi → DRabin

1. Ansatz: Potenzmengenkonstruktion
2. Verbesserter Ansatz: Verfolge das Wachsen des unendlichen Berechnungsbaumes des NBA auf seinem Inputwort und akzeptiere, wenn darin ein unendlicher Pfad existiert, der immer wieder einen Endzustand hat.

   Konstruktion von Muller/Schupp anhand des NBA $(a+b)^*a(b+a)^\omega$
   Berechnungsbaum auf $abbabab\ldots$

   a) Fasse bei der Fortsetzung Nichtendzustände und Endzustände jeweils zusammen: (nach oben bzw. nach unten)

      Lemma: NBA $\mathfrak{A}$ akzeptiert $\alpha \Leftrightarrow$ im Berechnungsbaum existiert gemäß 2a-Reduktion ein unendlicher Pfad, der immer wieder nach unten verzweigt.

      Beweis:
      - $\Rightarrow$: Klar.
      - $\Leftarrow$: Wähle Pfad entsprechende Voraussetzung und betrachte den zugehörigen partiellen Laufbaum. Dieser Baum ist unendlich, endlich verzweigt. Das Lemma von König liefert die Behauptung (Existenz eine unendlichen Pfades).

   b) Reduktion: Streiche Zustand q, wenn er weiter unten auftritt.

      Lemma: $\mathfrak{A}$ akzeptiert $\alpha \Leftrightarrow$ Im Berechnungsbaum existiert gemäß 2a- und 2b-Reduktion ein unendlicher Pfad, der immer wieder nach unten verzweigt.

   c) Reduktion: Komprimiere Pfadsegmente zu Einzelknoten.
      Idee: Hierbei kennzeichne Pfadstück (Knoten) mit
      - grün: "hat Endzustand dazubekommen"
      - gelb: "hat Endzustand, aber nicht dazubekommen"
      - rot: "hat keinen Endzustand"

3. Ansatz für Rabin-Automaten:
   - Zustandsmenge = Menge dieser (reduzierten) Bäume (= MS-Bäume = Muller-Schupp-Bäume)
   - Transitionsfunktion gemäß Update-Vorschrift
   - $\Omega = ((E_1, F_1), \ldots, (E_r, F_r))$
   - F_i = Menge der Bäume mit Knotenname i, grün
   - E_i = Menge der Bäume mit Knotenname i

   Knotenspezifikation: Name, (evtl.) Zustände, Farbe
   Anfangszustand: $1, q_0, \begin{cases} \text{grün} & \text{falls } q_0 \in F \\ \text{rot} & \text{falls } q_0 \notin F \end{cases}$
   Zu einem MS-Baum $t, a \in \Sigma$ definiere neuen MS-Baum $\delta(t, a)$:

   a) Kopiere t mit Farbwechsel grün *to* gelb
   b) Ersetze Blatt mit Zustandsmenge P durch $\{q \in Q | \exists p \in P, (p, a, q) \in \Delta\}$. Streiche einen Zustand, falls er weiter unten auftritt. Spalte die Menge jeweils auf in Endzustände und Nichtendzustände, kreiere neue Söhne, Knotennamen neu, grün für Endzustände.
   c) Streiche alle Knoten, die in diesem Schritt keinen neuen $\neq \emptyset$ Nachkommen erhalten haben.
   d) Ziehe Pfadsegmente zusammen in den Knoten am weitesten zur Wurzel (gemeint ist wohl der der Wurzel naheste Knoten) mit Färbung grün, falls neuer Endzustand hinzukommt (Pfadsegment hinzu, grün oder gelb)

Beispiel (s.o.):

Größenabschätzung für DRA (Anzahl der MS-Bäume): Vorgabe Q, $|Q| = n$

- Bemerkung: MS-Baum über Q ist strikt binär mit höchstens $|Q|$ Blättern, folglich höchstens $|Q|-1$ inneren Knoten
- Knotennamen für neu eingeführte Knoten jeweils neu (im Vorgängerbaum unbenutzt). Folglich genügt ein Namenreservoir von $3n$ Namen $N := \{1, \ldots, 3n\}$
- MS-Baum beschreibbar durch folgende Funktionen
  - $p : N \to N \cup \{0, *\}$ mit

$$p(i) = \begin{cases} * & i \text{ nicht vorhanden} \\ 0 & i \text{ Wurzel} \\ \text{Vaterknoten von } i & \text{sonst} \end{cases}$$

    Konvention: linker Sohnname jeweils kleiner als rechter Sohnname
  - $c : N \to \{\text{rot, gelb, grün}\}$
  - $z : Q \to N \cup \{*\}$

$$z(q) = \begin{cases} * & q \text{ nicht vorhanden} \\ \text{Blattname mit } q & \text{sonst} \end{cases}$$

- Anzahl der MS-Bäume $\leq$ Anzahl der Funktionentripel (p, c, z)
  - Anzahl der p: $(3n+2)^{3n}$
  - Anzahl der c: 3^{3n}
  - Anzahl der z: $(3n+1)^n$

  Insgesamt $\leq (3n+2)^{7n} \leq 4n^{7n} = 2^{O(n \log n)}$
- Ergebnis: Der DRA zu NBA mit n Zuständen hat $2^{O(n \log n)}$ Zustände.

Ziel:

Satz: Es gibt keine Übersetzung, die nichtdeterministische Büchi-Automaten mit $O(n)$ Zuständen in deterministische Rabin-Automaten mit $2^{O(n)}$ übersetzt.

Beweisstrategie:

1. Definition einer ω-Sprache $L_n \subseteq \{1, \ldots, n, \#\}$, die durch NBA mit $O(n)$ Zuständen erkannt wird.
2. Jeder DRA, der L_n erkennt, hat $\geq n!$ Zustände.

Vorbereitung:

Vereinigungslemma: Gegeben zwei Läufe ρ_1, ρ_2 eines Rabin-Automaten mit $\Omega = ((E_1, F_1), \ldots, (E_k, F_k))$ und ein Lauf ρ mit $\text{Inf}(\rho) = \text{Inf}(\rho_1) \cup \text{Inf}(\rho_2)$. Sind ρ_1, ρ_2 nicht akzeptierend, so auch ρ.

Beweis: Anname: ρ_1, ρ_2 nicht akzeptierend, aber ρ doch. ρ erfüllt Bedingung zu Ω, also $\exists i \in \{1, \ldots, k\}$: $\text{Inf}(\rho) \cap E_i = \emptyset, \text{Inf}(\rho) \cap F_i \neq \emptyset$. Insbesondere $\text{Inf}(\rho_1) \cap E_i = \emptyset$, $\text{Inf}(\rho_2) \cap E_i = \emptyset$. Ebenso $\text{Inf}(\rho_1) \cap F_i \neq \emptyset$ oder $\text{Inf}(\rho_2) \cap F_i \neq \emptyset$. Also ρ_1 akzeptierend oder ρ_2 akzeptierend. Widerspruch.

q.e.d.

Zu 1:

- NBA $\mathfrak{A}_n$ für L_n:
- Behauptung: $\alpha \in L_n \Leftrightarrow$ es existiert eine Folge $i_1, \ldots, i_k$ mit: α beginnt mir i_1 und jedes Buchstabenpaar $i_1 i_2, i_2 i_3, \ldots, i_{k-1} i_k, i_k i_1$ kommt unendlich oft in α vor (Zykeleigenschaft).
  - $\Leftarrow$) α erfülle die Zykeleigenschaft mit $i_1, \ldots, i_k$. Akzeptierender Lauf:

– $\Rightarrow$) $\mathfrak{A}_n$ akzeptiere α, aber die Zykeleigenschaft ist verletzt. Wähle eine Position in α; von wo an alle auftretenden Segmente i_1, i_2 unendlich oft auftreten. Wenn $q_i \neq q_j$ danach besucht wird und dann verlassen, dann q_i später nicht wieder besucht. Wegen Wahl Position erhalte sonst aus Wiederbesuch von q_i einen Zykel, der immer wieder durchlaufen wird.

- Aus Zykeleigenschaft folgt: Ist $(i_1, \ldots, i_n)$ eine Permutation von $(1, \ldots, n)$, dann $(i_1 \ldots i_n \#)^\omega \notin L_n$.

Zu 2:

- Betrachte DRA $\mathcal{R}_n$, der L_n erkennt. Betrachte zwei Permutationen $(i_1 \ldots i_n)$, $(j_1 \ldots j_n)$ von $(1 \ldots n)$. $\mathcal{R}_n$ akzeptiert weder $\alpha_1 = (i_1 \ldots i_n \#)^\omega$ noch $\alpha_2 = (j_1 \ldots j_n \#)^\omega$. Seien ρ_1, ρ_2 die enstprechenden Läufe und $R = \mathrm{Inf}(\rho_1)$, $S = \mathrm{Inf}(\rho_2)$. Weder ρ_1 noch ρ_2 akzeptierend.
- Zeige: $R \cap S = \emptyset$. (Dann fertig, da für die $n!$ vielen möglichen $(i_1 \ldots i_n \#)^\omega$ jeweils neue Zustände benötigt werden.)
- Annahme: $q \in R \cap S$. Neues Input durch Alternation zwischen oberem, unterem Loop. Inputfolge hat $i_1 \ldots i_n, j_1 \ldots j_n$ als Infix unendlich oft. Lauf ρ : $\mathrm{Inf}(\rho) = R \cup S$. Nach Vereinigungslemma: ρ ist nicht akzeptierend. Aber: Neues Inputwort erfüllt die Zykeleigenschaft (müßte also akzeptiert werden, Widerspruch!). Zykeleigenschaft: Sei k die erste Stelle mit $i_k \neq j_k$

$$\begin{array}{ccccccccc} i_1 & \ldots & i_{k-1} & i_k & \ldots & i_l & \ldots & i_n \\ \| & & \| & & \times & & & \\ j_1 & \ldots & j_{k-1} & j_k & \ldots & j_{l'} & & j_n \end{array}$$

Erhalte Zykel $i_k i_{k+1} \ldots i_l j_{k+1} \ldots j_{l'} i_{k+1}$.

Offenes Problem: Analoger Satz für Transformation NBA $\rightarrow$ DMullerA

Schlußbemerkung zu Akzeptierbedingungen:

- Muller-Bedingung für ρ: $\mathrm{Inf}(\rho) \in \mathcal{F} = \{F_1, \ldots, F_k\}$, $\mathrm{Inf}(\rho) = F_1 \vee \ldots \vee \mathrm{Inf}(\rho) = F_k$
- Rabin-Bedingung $(\mathrm{Inf}(\rho) \cap E_n = \emptyset \wedge \mathrm{Inf}(\rho) \cap F_1 \neq \emptyset) \vee \ldots \vee (\mathrm{Inf}(\rho) \cap E_k = \emptyset \wedge \mathrm{Inf}(\rho) \cap F_k \neq \emptyset)$
- Street-Bedingung – Negation von Rabin-Bedingung: Konjunktion von Fairness-Bedingung

3 Logische Theorien und ω-Automaten

3.1 Hintergrund

Model-Checking-Problem (klassisch, mathematisch)

- Gegeben: Mathematische Struktur S (z.B. $(\mathbb{R}, +, \cdot, 0, 1)$), Satz φ einer Logik-Sprache $\mathcal{L}$
- Frage: Gilt φ in S ? ($S \vDash \varphi$?)

Zusammenfassung der in S gültigen $\mathcal{L}$-Sätze liefert die "$\mathcal{L}$-Theorie von S". Für festes S ist die Frage nach algorithmischer Lösbarkeit des Model-Checking-Problems die Frage nach Entscheidbarkeit der $\mathcal{L}$-Theorie von S.

Model-Checking-Problem (informatisch)

- Gegeben: System (Programm, Protokoll, ...) S, Spezifikation φ formuliert in Logik-Sprachen
- Frage: Genügt S der Spezifikation ? ($S \vDash \varphi$?)

Zu klassischen Entscheidungsproblemen: Zentral die Struktur "Struktur der Arithmetik" $\mathfrak{N} = (\mathbb{N}, +, \cdot, 0, 1, y)$

- Standardsprache $\mathcal{L}_1$ der Logik erster Stufe:
  - Variablen $x, y, z, \ldots$ für Zahlen
  - Terme: $x, y, \ldots, 0, 1, (1+1) \cdot x, \ldots$
  - Formeln: $x < y + 1, x = y \cdot z, \neg, \vee, \wedge, \rightarrow, \leftrightarrow, \exists, \forall$
- $\mathcal{L}_1$-Theorie von $\mathfrak{N}$ ist die Menge der $\mathcal{L}_1$-Sätze, die in $\mathfrak{N}$ gelten (kurz "Arithmetik 1.Stufe").
- Beispielsätze: $\forall x \exists y (x < y \wedge \exists z z \cdot z = y)$
- Gödel, Taski (1931/1936): Die Arithmetik 1.Stufe ist unentscheidbar.
- Fragmente: $\mathcal{L}_1$-Theorie von $(\mathbb{N}, +, 0, <)$ = Presburger-Arithmetik: entscheidbar
- In der Logik $\mathcal{L}_2$ zweiter Stufe stehen auch Variablen und Quantoren für Relationen über $\mathbb{N}$ zur Verfügung.

Bemerkung (Gödel): Die $\mathcal{L}_2$-Theorie von $(\mathbb{N}, 0, +1)$ ist unentscheidbar.

- "$x+y=z$" ist in $\mathcal{L}_2$ definierbar, ebenso $x \cdot y = z$. $x+y=z :\Leftrightarrow$ jede Relation (zweistellig), die $(x, 0)$ enthält und mit (x_1, x_2) jeweils auch $(x_1 + 1, x_2 + 1)$ enthält, enthält (z, y). [Dann notwendigerweise $z = x + y$.]
- $\forall R (R(x, 0) \wedge \forall x_1 x_2 (R(x_1, x_2) \rightarrow R(x_1 + 1, x_2 + 1) \rightarrow R(z, y))$

Frage von Tarski: Was ergibt sich für $(\mathbb{N}, 0, +1)$ mit der "monadischen Logik zweiter Stufe" $M\mathcal{L}_2$, wobei neben $\mathcal{L}_1$ noch Variablen, Quantoren für **Zahlenmengen** erlaubt sind, z.B. Induktionsprinzip:

$$\forall X (X(0) \wedge \forall y (X(y) \rightarrow X(y+1)) \rightarrow \forall z X(z))$$

gilt in $(\mathbb{N}, 0, +1)$ mit $0 \in X$ und gehört also zur $M\mathcal{L}_2$-Theorie von $(\mathbb{N}, 0, +1)$. Entscheidbarkeit durch Büchi 1961 (mit Automaten).

3.2 S1S und Büchi-Automaten

Syntax von S1S (second order theory of 1 successor, $M\mathcal{L}_2$-Logik für $(\mathbb{N}, 0, +1)$)

- Variablen $x, y, \ldots$ für Zahlen, $X, Y, \ldots$ für Mengen natürlicher Zahlen
- Terme: $0, x, \ldots, 0+1, 0+1+1, \ldots, x+1, x+1+1, \ldots$, $[\text{succ}(0), \text{succ}(\text{succ}(0)), \ldots]$
- Formeln: $\tau_1 = \tau_2$ (τ_1, τ_2 Terme), $X(\tau)$, τ Term, $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \exists, \forall$

Zur Semantik: Eine Menge $P \subseteq \mathbb{N}$ ist identifizierbar mit der Bitfolge $\alpha_p = \alpha_p(0)\alpha_p(1)\alpha_p(2)\ldots$ mit

$$\alpha_p(i) = \begin{cases} 0 & \text{falls } i \notin P \\ 1 & \text{falls } i \in P \end{cases}$$

Beispiel P = Menge der geraden Zahlen: $\alpha_P = 10101010\ldots$, $\mathbb{P}$ = Menge der Primzahlen: $\alpha_{\mathbb{P}} = 00110101\ldots$
Menge $P \subseteq \mathbb{N} \sim \omega$-Wort über $\{0,1\}$.

Analog: Tupel $(P_1, \ldots, P_n)$ von Mengen $\subseteq \mathbb{N}$ identifizierbar mit ω-Wort über $\{0,1\}^n$, $\alpha_{P_1 \ldots P_n} = \alpha_{\bar{P}}(0)\alpha_{\bar{P}}(1)\ldots$:
$\alpha_{\bar{P}}(i)$ ist Bit-n-Tupel $(b_1, \ldots, b_n)$ mit $b_j = \begin{cases} 0 & i \notin P_j \\ 1 & i \in P_j \end{cases}$

Beispiel $\bar{P} = (P_1, P_2)$ wie in Beispiel 3.2, $\alpha_{\bar{P}} = \binom{1}{0}\binom{0}{0}\binom{1}{1}\binom{0}{1}\ldots$

Folglich definiert S1S-Formel $\varphi(X_1, X_2)$ die Menge derjenigen P_1, P_2, die φ erfüllen, äquivalent zu der Menge der zugehörigen $\alpha_{\bar{P}}$, d.h. ω-Sprache $L \subseteq (\{0,1\}^2)\omega$.
Auf dieser Basis haben wir die Semantik wie üblich über der Struktur $(\mathbb{N}, 0, +1)$.

Bemerkung: "$x < y$" ist durch S1S-Formel $\varphi_<(x, y)$ ausdrückbar. Idee:

- $\exists X(\neg X(x) \wedge X(y) \wedge \forall z(X(z) \rightarrow X(z+1)))$
- $L \subseteq (\{0,1\}n)^\omega$ ist **S1S-definierbar**, falls eine S1S-Formel $\varphi(X_1, \ldots, X_n)$ existiert, sodaß für $\bar{P} = (P_1, \ldots, P_n)$ jeweils gilt: $(P_1, \ldots, P_n)$ erfüllt $\varphi(X_1, \ldots, X_n) \Leftrightarrow \alpha_{\bar{P}} \in L$

Beispiel

- $L_1 = \{\alpha \in \{0,1\}^\omega | \alpha \text{ hat unendlich viele } 1\}$
- $\varphi_1(X_1) : \forall x \exists y(x < y \wedge X_1(y))$

Beispiel

- $L_2 = (00)^* 1^\omega$
- $\varphi_2(X_1) : \exists y[\forall x(x < y \rightarrow \neg X_1(x)) \wedge \forall z(z = y \vee y < z \rightarrow X_1(z)) \wedge \exists X(X(0) \wedge \forall t(X(t) \leftrightarrow \neg X(t+1)) \wedge X(y))]$

Beispiel

- L_3 definiert durch Büchi-Automat
- $\varphi_3(X_1)$ gesucht.
- Idee:

α	$=$	1	1	1	0	1	1	1	
$Lauf$		q_1	q_2	q_1	q_2	q_3	q_3	q_3	$\ldots$
φ_1		*		*					
φ_2			*		*				
φ_3						*	*	*	

- $\varphi_3(X_1) : \exists Y_1 \exists Y_2 \exists Y_3(Y_1, Y_2, Y_3$ bilden Partition $\wedge\, Y_1(0) \wedge \forall t(Y_1(t) \wedge X_1(t) \wedge \varphi_2(t+1)) \vee \ldots \vee \ldots \vee (Y_3(t) \wedge (X_1(t) \wedge Y_3(t+1)) \wedge \forall x \exists y(x < y \wedge Y_3(y)))$

Beispiel

- S1S-Formel: $\varphi(x) := \forall x \exists y(x < y \wedge X(y))$
- Menge $X \subseteq \mathbb{N} \sim \omega$-Wort: $\alpha_x \in \{0,1\}^\omega$
- Beispiel: Primzahlmenge $P \sim 00110101\ldots$

Satz (Büchi): $L \subseteq (\{0,1\}^n)^\omega$ ist Büchi-erkennbar $\Leftrightarrow$ L S1S-definierbar (durch Formel $\varphi(X_1, \ldots, X_n)$).

Beweis:

- $\Rightarrow$:
  - Gegeben: NBüchi-Automat $\mathfrak{A} = (\{q_1, \ldots, q_m\}, \{0,1\}^n, q_1, \Delta, F)$
  - Gesuchte Formel $\varphi_{\mathfrak{A}}(X_1, \ldots, X_n)$ muß ausdrücken: Die Fole zu $X_1, \ldots, X_n$ über $\{0,1\}^n$ wird von $\mathfrak{A}$ akzeptiert (d.h. es existiert ein erfolgreicher Lauf von $\mathfrak{A}$ auf dieser Folge).
  - Ansatz: Formuliere in S1S: Es existiert eine Menge $Y_1, \ldots, Y_m$, welche erfolgreichen Lauf kodiert (Y_k = Menge der Positionen, wo q_k besucht wird)
  - Bemerkung 1: $Y_1, \ldots Y_m$ sollen Partitionen der Menge $\mathbb{N}$ bilden:
    $$(Y_1, \ldots, Y_m) := \forall y \bigvee_{i=1}^{m} Y_i(y) \wedge \forall y \bigwedge_{i+j} \neg(Y_i(y) \wedge Y_j(y))$$
  - Konvention: Für Vektor $a = (b_1, \ldots, b_n) \in \{0,1\}^n$ schreibe $X_a(y)$ für $[b_1]X_1(y) \wedge \ldots \wedge [b_n]X_n(y)$ mit $[1] = \varepsilon, \quad [0] = \neg$.
  - Beispiel: $a = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$, $X_a(y)$ statt $X_1(y) \wedge \neg X_2(y) \wedge \neg X_3(y)$.
    $$\begin{aligned} \varphi_a(X_1, \ldots, X_n): \quad & \exists Y_1 \ldots \exists Y_m(\text{Partition } (Y_1, \ldots, Y_m) \wedge Y_1(0) \wedge \\ & \forall z \bigvee_{q_0, a, y_j) \in \Delta} (Y_i(z) \wedge X_a(z) \wedge Y_j(z+1)) \wedge \\ & \forall x \exists y(x < y \wedge \bigvee q_i \in F Y_i(y)) \end{aligned}$$
  - Bemerkung 2: In der Klammer nach $\exists Y_1, \ldots, Y_m$ steht eine Formel erster Stufe (nur Quantifizierungen über Zahlen). $q(X_1, \ldots, X_n)$ ist "existentielle S1S-Formel", kurz ES1S-Formel:
    $$\exists Y_1, \ldots, \exists Y_m(\underbrace{\varphi \vee \psi(\bar{X}, \bar{Y})}_{\text{1.Stufe}})$$
- $\Leftarrow$:
  - Ansatz: Iduktion über den Aufbau von S1S-Formeln
  - Vorbereitung: Umformung der S1S-Formeln in $S1S_0$-Formeln, in denen nur **Mengenvariablen** auftreten.
  - Arbeite mit $\{y\}$ statt y, $\{y\} \subseteq X$ statt $X(y)$.
  - $S1S_0$-Formeln:
    $$\begin{array}{ll} \text{Sing}(X) & "X \text{ ist Einermenge"} \\ X \subseteq Y & \\ \text{Succ}(X,Y) & X = \{x\}, Y = \{y\}, x+1 = y \end{array}$$
  - Von S1S-Formel zu $S1S_0$-Formel
    1. Eliminiere 0 durch Übergang von z.B. $X(0)$ zu $\exists y(X(y) \wedge \neg \exists x x < y)$.
    2. Eliminiere $<$ wie zuvor.
    3. Eliminiere die Mehrfachanwendung von "+1", statt $x+1+1 = y$ schreibe $\exists y_1(x+1 = y_1 \wedge y_1+1 = y)$. Erhalte S1S-Formel mit atomaren Formeln $X(y), \quad x+1 = y, \quad x = y$.
    4. Übergang zu $S1S_0$ durch Ersetzen von
       - $X(y)$ durch $\text{Sing}(Y) \wedge Y \leq X$,
       - $x+1 = y$ durch $\text{Sing}(X) \wedge \text{Sing}(Y) \wedge \text{Succ}(X,Y)$,

    - $x = y$ durch $X = Y \wedge \mathrm{Sing}(X) \wedge \mathrm{Sing}(Y)$,
    - $\forall x(\varphi(x, Y)$ durch $\forall X(\mathrm{Sing}(X) \to \tilde{\varphi}(X, Y))$,
    - $\exists \varphi(x, Y)$ durch $\exists X(\mathrm{Sing}(X) \wedge \tilde{\varphi}(X, Y))$.
  - Beispiel: $\forall x \exists y(x + 1 = y \wedge z(y))$ wird zu $\forall X(\mathrm{Sing}(X) \to \exists Y(\mathrm{Sing}(Y) \wedge \mathrm{Succ}(X, Y) \wedge X \subseteq Z))$
  - Fehlt / Fazit: Zu jeder $S1S_0$-Formel $q(X_1, \ldots, X_n)$ kann man einen äquivalenten Büchi-Automaten (über $\{0,1\}^n$) konstruieren.
  - Induktion: $\mathrm{Sing}(X), \quad X \subset Y, \quad \mathrm{Succ}(X, Y))$
    - Induktionsschritt: $\neg, \vee, \exists X$
    - Büchi-Automat (über $\{0,1\}$ für $\mathrm{Sing}(X_1)$, über $\{0,1\}^2$ für $X_1 \subseteq X_2$)
    - Zum Induktionsschritt: Zu $\varphi_1(X_1, \ldots, X_n)$, $\varphi_2(X_1, \ldots, X_n)$ seien gemäß Induktionsvoraussetzung äquivalente Büchi-Automaten $\mathfrak{A}_1, \mathfrak{A}_2$ gegeben. Finde Büchi-Automaten zu
      - $\neg\varphi_1(X_1, \ldots, X_n)$ mit Übergang zu deterministischen Muller-Automaten, Komplement-Muller-Automaten, Konstruktion von nichtdeterministischen Büchi-Automaten.
      - $\varphi_1 \vee \varphi_2$ einfach (Vereinigungsautomat)
      - $\exists X$: Betrachte $\psi(X_1, \ldots, X_n) = \exists X_1 \varphi_1(X_1, \ldots, X_n)$. In $\mathfrak{A}_1$ streiche in den Transitionen jeweils die erste Komponente und erhalte so $\mathfrak{A}_1'$.

        Behauptung: $\mathfrak{A}_1'$ und ψ sind äquivalent.

        Beweis: $\mathfrak{A}_1'$ akzeptiert $\alpha \in (\{0,1\}^{n-1})^\omega \Leftrightarrow$ es existiert eine Bitfolge $c_0 c_1 \ldots$, sodaß $\mathfrak{A}_1$ akzeptiert: $\underbrace{(c_0, \alpha(0)), (c_1, \alpha(1)), (c_1, \alpha(2)), \ldots}_{\text{erfüllt nach IV } \varphi(X_1, \ldots, X_n)} \Leftrightarrow \alpha(0), \alpha(1), \ldots$ erfüllt $\exists X_1 \varphi_1(X_1, \ldots, X_n)$

Folgerungen und Bemerkungen zum Äquivalenzsatz

Satz: Die Menge der in $(\mathbb{N}, 0, +1)$ wahren S1S-Sätze (ohne freie Variablen) ist entscheidbar.

Beweis: Wende auf Satz φ die Transformation in Büchi-Autmaten $\mathfrak{A}_\varphi$ an. $\mathfrak{A}_\varphi$ hat unbeschriftete Transitionen, erlaubt erfolgreichen Lauf genau dann, wenn φ wahr ist. (Äquivalenz zu $n = 0$)

Anwendung: $(n = 0)$ S1S-Satz φ ist wahr $\Leftrightarrow$ inputfreier Büchi-Automat $\mathfrak{A}_\varphi$ mit unbeschrifteten Transitionen hat einen erfolgreichen Lauf.

Zur doppelten Anwendung des Äquivalenzsatzes:

Bemerkung: Jede S1S-Formel $\varphi(X_1, \ldots, X_n)$ ist zu einer ES1S-Formel $\psi(X_1, \ldots, X_n)$ äquivalent: $(\psi(X_1, \ldots, X_n) : \exists Y_1, \ldots, Y_n \underbrace{\psi_0(X_1, \ldots, X_n, Y_1, \ldots, Y_m)}_{\text{1.Stufe}})$

$\varphi \curvearrowright$ Baumautomat $\mathfrak{A} \curvearrowright \psi_{\mathfrak{A}_\varphi} ES1S$

Zur Komplexität der Transformationen $S1S \leftrightarrow$ Büchi-Automat:

- Transformation Büchi-Automat (mit $|\Sigma| = k, k = 2^n, |Q| = m$) führt auf S1S-Formel mit m Mengenoperatoren, Größe von ψ_0 in $O(k \cdot m^2)$.
- Transformation S1S $\curvearrowright$ Büchi-Automaten erfordert
  - $\geq$ exponentieller Aufwand für jeden Komplementschritt (via deterministischem Muller-Automat)
  - Einführung von Nichtdeterminismus bei $\exists$-Schritt

  Wegen Iteration ergibt sich hyperexponentieller Aufwand.

Satz: Sei A ein Transformationsalgorithmus von S1S nach Büchi-Automaten und sei $k > 0$ gegeben. Dann existieren Formeln φ_n, sodaß die jeweils durch A gelieferten Büchi-Automaten eine Zustandszahl haben, die nicht abschätzbar ist durch

$$2^{2^{\cdots^{2^n}}} \Big\} k$$

Komplexität ist "nicht-elementar".

Einheitsresultate für andere Strukturen

1. $\mathcal{L}_1 - Th(\mathbb{N}, +, 0)$ "Presburger-Arithmetik" – entscheidbar
2. Monadische $\mathcal{L}_2$-Theorie des binären Baumes $(\{0,1\}^*, \text{succ}_0, \text{succ}_1)$ "S2S" – entscheidbar (siehe Vorlesung Baumautomaten)
3. Monadische $\mathcal{L}_2$-Theorie des $(\mathbb{N} \times \mathbb{N})$-Gitters

Zu 1 – Presburger-Arithmetik

- Variablen $x, y, \ldots$ für Zahlen
- atomare Formeln $x + y = z, \ldots, x + 0 = z, \ldots, x = y$
- $\neg, \vee, \wedge, \exists, \forall$

Satz: Man kann eine Presburger-Formel $\varphi_(x_1, \ldots, x_n)$ in eine S1S-Formel $\hat{\varphi}(X_1, \ldots, X_n)$ transformieren mit

$$(\mathbb{N}, +, =) \vDash \varphi[k_1, \ldots, k_n] \Leftrightarrow (\mathbb{N}, 0, +1) \vDash \hat{\varphi}[M_{k_1}, \ldots, M_{[k_n}]]$$

Zur Abbildung $k \mapsto M_k$: M_k ist die Menge, die durch Invertieren der Darstellung von k dargestellt wird.

$k = 13$ Dualdarstellung:0001101 Inversion:1011000 ... $M_k = \{0, 2, 3\}$

Bemerkung: Jede Menge M_k ist endlich und erfüllt somit die Formel

$$\text{Fin}(X) := \exists y \forall x (X(x) \rightarrow x < y$$

Zum Beweis: (Formelübersetzung $\varphi \rightarrow \hat{\varphi}$) betrachte $\varphi : x + y = z$. Finde $\hat{\varphi}(X, Y, Z)$:
Beispiel:

X:	1	1	0	1	0	0	0	...
Y:	1	0	0	1	1	0	0	...
C:	0	1	1	0	1	1	0	...
Z:	0	0	1	0	0	1	0	...

$$\begin{aligned}\hat{\varphi_+}(X, Y, Z): \quad & \exists C(\text{Fin}(C) \wedge \neg C(0) \wedge Z(0) \leftrightarrow (((X(0) \wedge \neg Y(0)) \vee (\neg X(0) \wedge Y(0))) \wedge \\ & \forall y(C(y+1) \leftrightarrow \text{"zwei oder drei von } X(y), Y(y), C(y) \text{ treffen zu"}) \\ & \forall y(Z(y+1) \leftrightarrow \text{"eine oder drei von } X(y+1), Y(y+1), C(y+1) \text{ treffen zu"}))\end{aligned}$$

Induktionsschritte einfach: $Eins(X) : \forall x(X(x) \leftarrow x = 0 + 1$

Beispiel:

- $\forall x \exists y(y + y = y \vee \overbrace{y + y}^{Y'} + 1 = x)$
- $$\begin{aligned}\forall X(\text{Fin}(X) \rightarrow \quad & \exists Y(\text{Fin}(Y) \wedge \exists Y', Y''(\text{Fin}(Y') \wedge \\ & [\hat{\varphi_+}(Y, Y, X) \vee Eins(Y'') \wedge \hat{\varphi_+}(Y, Y, Y') \wedge \hat{\varphi_+}(Y', Y'', X)])\end{aligned}$$

Satz (zu): Folgendes Problem ist unentscheidbar:

- Gegeben: Monadische $\mathcal{L}_2$-Formel mit Konstanten $(0,0)$, Funktionen, $\text{succ}_\rightarrow$, $\text{succ}_\downarrow$, ohne freie Variablen, $\text{succ}_\rightarrow(i,j) = (i, j+1)$, $\text{succ}_\downarrow(i,j) = (i+1, j)$
- Frage: Gilt φ im $(\mathbb{N} \times \mathbb{N})$-Gitter ?

Beweisansatz: Zu Turingmaschine M finden einen Satz φ_M mit M stoppt angesetzt auf dem leeren Band $\Leftrightarrow$ $(\mathbb{N} \times \mathbb{N}, \text{succ}_{\rightarrow}, \text{succ}_{\downarrow}) \vDash \varphi_M$ (im Gitter gilt φ_M).

- Annahmen: Die Turingmaschine M hat k Zustände mit Anfang q_1, Stop q_k, m Buchstaben im Arbeitsalphabet, Blank a_1
- φ_M muß ausdrücken: Es gibt eine Folge von Konfigurationen

$$\begin{array}{cccccc} K_0 & q_1 & a_1 & a_1 & a_1 & \dots \\ K_1 & a_2 & q_2 & a_1 & a_1 & \dots \\ K_2 & a_2 & q_3 & q_5 & a_5 & \dots \\ & a_2 & q_3 & q_5 & a_1 & \dots \\ & & \vdots & & & \ddots \end{array}$$

- Hilfsmenge $X_i (1 \leq i \leq k)$ für Positionen mit Zustand q_i, $Y_j (1 \leq j \leq m)$ für Positionen mit Arbeitsbuchstabe a_j
- φ_M : $\exists Y_1 \dots \exists Y_m (\text{Partition}(X_1 \dots X_k Y_1 \dots Y_m) \wedge X_1(0,0) \wedge \forall y (\underbrace{y \text{ ist rechts von } (0,0)}_{\text{ausdrückbar wie } < \text{ mit } +1} \rightarrow Y_1(y)) \wedge$
  $\forall x_1 \dots x_6 (x_1 \dots x_6 \text{ bilden } 2 \times 3\text{-Fenster} \rightarrow \text{Mitgliedschaft in } X_i, Y_j \text{ – kompatibel mit TM } M \wedge \exists z X_i(z))$

4 LTL Model Checking

- Model-Checking-Problem: Zu Struktur S und Logik-Formel φ prüfe, ob $S \vDash \varphi$
- Klassisches Problem: $S = (\mathbb{N}, 0, +1)$, φ in einer Logik wie 1. Stufe, monadische 2. Stufe
- Informatik: S ist Transitionsgraph (Repräsentation des Zustandsraumes eines Programmes), φ ist Formel für Bedingung an die Programmläufe. Hier Logik LTL (linear time temporal logic).
- Vorbereitung: Festlegung der Transitionsstrukturen S
- Idee: Knoten des Transitionsgraphen = Zustände eines Programmes; hier zumeist Kontrollprogramme wie z.B. Mutual-Exclusion-Protokolle. Der Zustandsraum ist endlich, falls endlich viele Boolesche Variablen beteiligt sind, ausgezeichnet ist normalerweise ein Anfangszustand.
  Betrachte alle möglichen Läufe (Pfade durch den Transitionsgraphen). Hinzu kommen die Zustandseigenschaften $p_1, \ldots, p_n$. Zu $s \in S$ (Zustandsmenge) ist jeweils festgelegt, welche p_i dort gelten.
- Formalisierung: Kripke-Struktur $\underline{S} = (S, R, \lambda)$ mit Anfangszustand $s_0 \in S$:
  - S Zustandsmenge
  - $R \subseteq S \times S$ Transitionsrelation
  - $\lambda_i S \to \mathrm{Pot}(\{p_1, \ldots, p_n\})$, $p_i \in \lambda(s) \Leftrightarrow$ in S gilt p_i. Alternativ ist $\lambda(s)$ ein Bitvektor $(b_1, \ldots, b_n)$ mit $b_i = 1 \Leftrightarrow p_i \in \lambda(s)$.

  Beispiel: Kripke liefert die Menge der zulässigen Zustandsfolgen $q_0 q_1 q_2 \ldots \in S^\omega$ mit $q_0 = s_0$, $(q_i, q_{i+1}) \in R$.
  - Zugehörige Folge der Beschriftungen: $\lambda(q_0)\lambda(q_1)\ldots \in (\{0,1\}^n)^\omega$
  - Sprache $L(\underline{S})$ = Menge dieser Folgen

Model-Checking-Problem:

- Gegeben: $\underline{S}$, Logik-Formel φ als Bedingung an Beschriftungsfolgen
- Frage: Erfüllt jede Beschriftungsfolge $\alpha \in L(\underline{S})$ die Formel φ

Ziel:

- Einführung von LTL
- Lösung des MC-Problemes $L(\underline{S}) \subseteq L(\mathfrak{A}_\varphi)$

4.1 Logik LTL

Syntax Benutze $p_1, \ldots, p_n$, Boolesche Junktoren und temporale Operatoren:

- Xp_1: Zum nächsten Zeitpunkt gilt p_1 ("next p_1")
- Gp_1: Von jetzt an gilt p_1 ("global p_1")
- Fp_1: Irgendwann von jetzt an gilt p_1 ("finally p_1")
- $p_1 U p_2$: Irgendwann von jetzt an gilt p_2, und bis dann (ausschließlich) gilt immer p_1 ("until")

Syntax – formal: LTL-Formeln über $p_1, \ldots, p_n$ sind induktiv wie folgt gegeben:

- $p_1, \ldots, p_n$ sind LTL-Formeln
- Mit φ_1, φ_2 sind jeweils auch $\neg\varphi_1, \varphi_1 \vee \varphi_2, \varphi_1 \wedge \varphi_2, \varphi_1 \to \varphi_2$ LTL-Formeln
- Mit φ_1, φ_2 sind jeweils auch $X\varphi_1, G\varphi_1, F\varphi_1, \varphi_1 U \varphi_2$ LTL-Formeln

Zur Semantik: Definiere für $\alpha \in (\{0,1\}^n)^\omega$ und LTL-Formel φ, ob $\alpha \vDash \varphi$ für $\alpha = \alpha(0)\alpha(1)\alpha(2)\ldots$ sei $\alpha^i = \alpha(i)\alpha(i+1)\ldots$; $(\alpha(i))_j = j$-te Komponente von $\alpha(i)$.

Semantik-Definition – induktiv Für $\alpha \in (\{0,1\}^n)^\omega, \varphi$ LTL-Formel über $p_1, \ldots, p_n$.

- $\alpha \vDash p_i \Leftrightarrow ((\alpha(0))_i = 1$
- $\alpha \vDash \neg\varphi(\varphi_1 \vee \varphi_2, \varphi_1 \wedge \varphi_2, \ldots) \Leftrightarrow$ nicht $\alpha \vDash \varphi$ ($\alpha \vDash \varphi_1$ oder $\alpha \vDash \varphi_2 \ldots$ und $\ldots$)
- $\alpha \vDash X\varphi \Leftrightarrow \alpha^1 \vDash \varphi$
- $\alpha \vDash G\varphi \Leftrightarrow \forall i \geq 0(\alpha^i \vDash \varphi)$
- $\alpha \vDash F\varphi \Leftrightarrow \exists i \geq 0(\alpha^i \vDash \varphi)$
- $\alpha \vDash \varphi_1 U \varphi_2 \Leftrightarrow \exists i \geq 0(\alpha^i \vDash \varphi_2$ und $\forall j$ mit $0 \leq j < i : \alpha^j \vDash \varphi_1)$

Zu φ über $p_1, \ldots, p_n$ sei

$$L(\varphi) = \{\alpha \in (\{0,1\}^n)^\omega | \alpha \vDash \varphi\}$$

L heißt LTL-definierbar, falls eine LTL-Formel φ existiert mit $L = L(\varphi)$.

Beispiel:

- Sicherheitseigenschaft "immer p_1": Gp_1
- Garantieeigenschaft "irgendwann p_1": Fp_1
- Periodizitätseigenschaft "anfangs gilt p_1 und dann zu genau jedem dritten Zeitpunkt": $p_1 \wedge X\neg p_1 \wedge XX\neg p_1 \wedge G(p_1 \leftrightarrow XXXp_1)$
- Büchi-Eigenschaft "immer wieder p_1": GFp_1
- Fairness-Eigenschaft $GFp_1 \rightarrow GFP_2$
- Request-Response "immer wenn p_1, dann später einmal p_2": $G(p_1 \rightarrow XFp_2)$
- Eine Until-Bedingung: "irgendwann p_1, später noch einmal p_1, und dazwischen immer p_2": $F(p_1 \wedge X(p_2 U p_1))$

Bemerkung (zur Formulierung in S1S): Festes Modell α, Angabe der Positionen (Zeitpunkte) durch Elementvariablen $x, y, \ldots$. Gelten von p_i durch Behauptung über X_i *[Satz nicht fortgesetzt.]*

Beispiel:

- p_1, p_2 LTL: $G(p_1 \rightarrow XFp_2$
- X_1, X_2 S1S: $\forall x(X_1(x) \rightarrow \exists y(x < y \wedge X_2(y))$
- Periodizität (s.o.): $X_1(0) \wedge \neg X_1(0+1) \wedge \neg X_1(0+1+1) \wedge \forall y(X_1(y) \leftrightarrow X_1(y+1+1+1))$

Ziel: Lösung des Model-Checking-Problems für LTL

Ansatz:

- Gegeben: Kripke-Struktur $(\mathcal{M}, s)$, LTL-Formel φ
- Frage: Gilt $(\mathcal{M}, s) \vDash \varphi$?
  1. Schritt: Transformation von $(\mathcal{M}, s)$ in Büchi-Automaten $\mathfrak{A}_{(\mathcal{M},s)}$
     - Konstruktion für $\mathcal{M} = (S, R, \lambda)$, $\lambda : S \rightarrow \{0,1\}^n(\text{Pot}\{p_1, \ldots, p_n\})$
     - Büchi-Automat $\mathfrak{A}_{\{}(\mathcal{M}, s)\} = (S, \{0,1\}^n, s, \Delta, S)$ mit $(s', \bar{b}, s'') \in \Delta :\Leftrightarrow \lambda(s') = \bar{b}$
  2. Schritt: Übergang von $\varphi = \varphi(p_1, \ldots, p_n)$ zu Büchi-Automat $\mathfrak{A}_{\neg\varphi}$ über $\{0,1\}^n$. $L(\mathfrak{A}_{\neg\varphi}) =$ Menge der $\alpha \in (\{0,1\})^n)^\omega$, die φ verletzen
     Test, ob $L(\mathfrak{A}_{(\mathcal{M},s)}) \cap L(\mathfrak{A}_{\neg\varphi}) \neq \emptyset$ klärt, ob in $(\mathcal{M}, s)$ Pfade existieren, die φ verletzen.
  3. Schritt: Konstruktion des Durchschnittsautomaten zu $\mathfrak{A}_{(\mathcal{M},s)}, \mathfrak{A}_{\neg\varphi}$ und Test auf Leerheit.

zu 2: Führe für $\alpha = (\{0,1\}^n)^\omega$ und LTL-Formel $\varphi(p_1, \ldots, p_n)$ ein:

- Folge der Subformeln nach wachsender Komplexität: $\varphi_1, \ldots, \varphi_n, \varphi_{n+1}, \ldots, \varphi_{n+m}$
  Beispiel: $\varphi = Xp_1UX\neg p_2 \rightsquigarrow p_1, p_2, Xp_1, \neg p_2, X\neg p_2, Xp_1UX\neg p_2$
- Diagramm der "φ-Expansion von α", formal ω-Wort $\beta \in (\{0,1\}^{n+m})^\omega$ definiert durch $(\beta(i))_j = 1 \Leftrightarrow \alpha^i \vDash \varphi_j$
  Beispiel: $\varphi = p_1 \vee X(\neg p_2 U p_2) \rightsquigarrow$

β

p_1	0	0	1	0	1	0	0	1	0	0
p_2	0	0	0	1	0	0	0	1	0	0
$\neg p_2$	1	1	1	0	1	1	1	0	1	1
$\neg p_2 U p_1$	1	1	1	0	1	1	1	1	0	0
$X(\neg p_2 U p_1)$	1	1	0	1	1	1	1	0	0	0
φ	1	1	1	1	1	1	1	1	0	0

Vorbereitende Konvention: Nehme als Operatoren zum Formelaufbau nur: $\neg, \vee, X, U$ (Ansonsten bringe Formel in diese Form, mit üblichen Booleschen Umformungen und den Äquivalenzen $F\psi \equiv trueU\psi$, $G\psi \equiv \neg F\neg\psi$) $\psi_1 U \psi_2 \equiv \psi_2 \vee (\psi_1 \wedge X(\psi_1 U \psi_2))$

Verträglichkeitsbedingungen für die φ-Expansion (Formelnotation wie oben)

$$
\begin{array}{ll}
\varphi_j = \neg\varphi_{j_1} & (\beta(i))_j = 1 \Leftrightarrow (\beta(i))_{j_1} = 0 \\
\varphi_j = \varphi_{j_1} \overset{\vee}{\wedge} \varphi_{j_2} & (\beta(i))_j = 1 \Leftrightarrow (\beta(i))_{j_1} \overset{\vee}{\wedge} (\beta(i))_{j_2} = 1 \\
\varphi_j = X\varphi_{j_1} & (\beta(i))_j = 1 \Leftrightarrow (\beta(i+1))_{j_1} = 1 \\
\varphi_j = \varphi_{j_1} U \varphi_{j_2} & (\beta(i))_j = 1 \Leftrightarrow (\beta(i))_{j_2} = 1 \text{ oder } ((\beta(i))_{j_1} = 1 \text{ und } (\beta(i+1))_j = 1)
\end{array}
$$

Zusatz $(*)$**:** Für $\varphi_j = \varphi_{j_1} U \varphi_{j_2}$: Es existiert kein k, sodaß für alle $l \geq k$ und $(\beta(l))_{j_2} = 0$.

Lemma: Gelten zu gegebenem α für β die Verträglichkeitsbedingungen, so ist β genau die φ-Expansion zu α.

Beweis: Induktion über den Formalaufbau, d.h. zeilenweise beim Aufbau der φ-Expansion. Trivial für Subformeln mit $\neg, \vee, \wedge, X$. Fall $\varphi_j = \varphi_{j_1} U \varphi_{j_2}$: Falls j_2-Komponente von $\beta(k) = 1$, dann für $l \leq k$ $(\beta(l))_j$ korrekt (nutze U-Verträglichkeitsbedingung aus).
Daraus folgt die Behauptung, falls die j_2-Komponente immer wieder wahr wird. Sonst ist aber der Stelle k die j_2-Komponente immer 0. Zeige, daß dann auch j-te Komponente gleich null ist. Sonst in j-ter Komponente 1 und j_2-ter Komponente j_2-ter Komponente immer Widerspruch zu $(*)$.

Definition von $\mathfrak{A}_\varphi$: (gemäß Notation wie oben, mit $p_1, \ldots, p_n$, Subformeln $\varphi_1, \ldots, \varphi_n, \varphi_{n+1}, \ldots, \varphi_{n+m}$)

- Zustandsmenge: $q_0 \cup \{0,1\}^{n+m}$ (restringiert auf die zulässigen Vektoren gemäß $\neg, \wedge, \vee$-Verträglichkeitsbedingung)
- Transitionen: $q_0 \xrightarrow{\frac{n}{b}} (\frac{n}{b}, \frac{m}{c})$ gemäß Verträglichkeitsbedingungen, und mit letzter c-Komponente $= 1$, $(\bar{b}, \bar{c}) \xrightarrow{\bar{b'}} (\bar{b'}, \bar{c'})$, falls Verträglichkeitsbedingung für $(\bar{b}, \bar{c})$ und $(\bar{b'}, \bar{c'})$ erfüllt.
- Endzustandsmenge F_j (für $\varphi_j = \varphi_{j_1} U \varphi_{j_2}$) = Menge der Vektoren mit j-Komponente gleich null und j_2-Komponente gleich eins.

[2004/01/17]

- LTL-Formel $\rightarrow \mathfrak{A}_\varphi$
- $\varphi(p_1, \ldots, p_n)$, Subformeln neben $p_1, \ldots, p_n : \varphi_{n+1}, \ldots, \varphi_{n+m} (= \varphi)$
- $\mathfrak{A}_\varphi$: Zustände: $q_0 \cup \{0,1\}^{n+m}$
  Transitionen formuliert gemäß Verträglichkeitsbedingungen Endzustandsmengen $F_1, \ldots, F_k$ für jede Until-Subformel $\varphi_j = \varphi_{j_1} U \varphi_{j_2}$ eine entsprechende Endzustandsmenge
- F_j enthält diejenigen Vektoren $\bar{b} \in \{0,1\}^{n+m}$ mit j-ter Komponente 1, j_2-ter Komponente 2. Dann gilt für jedes ω-Wort $\alpha \in (\{0,1\}^{n+m})^\omega$: α erfüllt $\varphi \Leftrightarrow \mathfrak{A}_\varphi$ hat auf α einen Lauf, der jede Menge F_j unendlich oft besucht.

Reduktion von verallgemeinerten Büchi-Automaten zu Büchi-Automaten Idee für den Fall zweier Mengen F_1, F_2:

1. Warte auf F_1-Zustand
2. falls F_1 besucht, warte auf F_2 Zustand
3. falls F_2 besucht, Übergang in Zustandsindex 3, dann zurück zu 1.

```
Lauf: F1   v v       v
      F2 v v   v v v v   v   v
```

Allgemeine Konstruktion zu $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F_1, F_2)$:

- $\mathfrak{A}' = (Q', \Sigma, q_0', \Delta', F'$
- $Q' := Q \times \{1,2,3\}$, $q_0' = (q_0, 1)$
- $((p,i), a, (q,i)) \in \Delta' :\Leftrightarrow i \leq 2, (p,a,q) \in \Delta, q \notin F_i$
- $((p,i), a, (q,i+1)) \in \Delta' :\Leftrightarrow i \leq 2, (p,a,q) \in \Delta, q \in F_i$
- $((p,3), a, (q,1)) \in \Delta' :\Leftrightarrow (p,a,q) \in \Delta$
- $F' := Q \times \{3\}$

Anwendung im MC-Problem $(\mathcal{M}, s_0) \vDash \varphi$ (LTL-Formel)

Ansatz:

1. $(\mathcal{M}, s_0) \mapsto$ Büchi-Automat $\mathfrak{A}_{(\mathcal{M},s_0)}$
2. $\varphi \mapsto$ Büchi-Automat $\mathfrak{A}_{\neg\varphi}$
   Zu testen: $L(\mathfrak{A}_{(\mathcal{M},s_0)})\mathcal{L}(\mathfrak{A}_{\neg\varphi} \neq \varnothing$?
3. Übergang von $\mathfrak{A}_{(\mathcal{M},s_0)}, \mathfrak{A}_{\neg\varphi}$ zum Durchschnittsautomaten $\mathfrak{A}_\cap$
4. $L(\mathfrak{A}_\cap) \neq \varnothing$? (einfach)

zu 4:

Satz: Zu zwei Büchi-Automaten $\mathfrak{A}_1 = (Q_1, \Sigma, q_{01}, \Delta_1, F_1)$ und $\mathfrak{A}_2 = (Q_2, \Sigma, q_{02}, \Delta_2, F_2)$ kann man $\mathfrak{A}_\cap$ konstruieren mit $L(\mathfrak{A}_\cap) = L(\mathfrak{A}_1) \cap L(\mathfrak{A}_2)$.

$$\mathfrak{A}_\cap = (Q_1 \times Q_2 \times \{1,2,3\}, \Sigma, (q_{01}, q_{02}, 1), \Delta', F')$$

Δ' gemäß Produktion mit Behandlung des Index wie oben, $F' = Q_1 \times Q_2 \times \{3\}$.

Satz (zur Komplexität des MC-Problems = LTL-MC): Das Problem 'Gegeben $(\mathcal{M}, s), \varphi$, gilt $(\mathcal{M}, s) \vDash \varphi$?' ist NP-schwer.

Beweis: $SAZ(3) \leq_p LTL - MC$: Transformiere KNF-Ausdruck ψ mit drei Literalen pro Klausel in $(\mathcal{M}, s)_\psi$ mit *psi* erfüllbar $\Leftrightarrow$ nicht $(\mathcal{M}, s) \vDash \varphi_\psi$.

Konstruktion per Beispiel $\psi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$ Beschrifte $(\neg)x_i$ mit (b_1, b_2), wobei $b_1 = 1$, falls $(\neg)x_i$ in der ersten Klausel vorkommt, $b_2 = 1$, falls $(\neg)x_2$ in der zweiten Klausel vorkommt. Dann ψ erfüllbar $\Leftrightarrow$ es existiert eine Belegung der Variablen x_1, x_2, x_3, die jede Klausel wahr macht $\Leftrightarrow$ es existiert durch $(\mathcal{M}, s)_\psi$ eine Pfad, der in jeder Komponente eine 1 trifft $\Leftrightarrow$ nicht: für alle Pfade existiert eine Komponente konstant 0 $\Leftrightarrow$ nicht $(\mathcal{M}, s) \vDash G\neg p_1 \vee G\neg p_2$ mit ψ erfüllbar $\Leftrightarrow$ nicht $(\mathcal{M}, s) \vDash \varphi_\psi$.

5 Schwache Automaten und Constraints über $(\mathbb{R}, +, <)$

Ein Automat $(Q, \Sigma, q_0, \delta, F)$ heißt E-/A-Automat, wenn er über ω-Wörtern mit folgender Akzeptierbedingung benutzt wird: $\mathfrak{A}$ akzeptiert $\alpha :\Leftrightarrow$ für den eindeutigen Lauf ρ auf α gilt: $\exists i \rho(i) \in F / \forall i \rho(i) \in F$.

- $\exists i \rho(i) \in F$: Garantieeigenschaft
- $\forall i \rho(i) \in F$: Sicherheitseigenschaft

[2005/01/24]

Beispiel: $\Sigma = \{0, 1\}, \quad L = \Sigma^* \cdot 1 \Sigma^\omega$

- Behauptung: L nicht A-erkennbar. $(A = \forall)$
- Sei $\mathfrak{A}$ Automat, der L A-erkennt, etwa mit n Zuständen. Betrachte $\mathfrak{A}$ auf $0^n 1 0^\omega$, wird A-akzeptiert. In der Zustandsfolge $p_0 \xrightarrow{0} p_1 \to \ldots \xrightarrow{0'} p_n \xrightarrow{1} \ldots$ erhalte Wiederholung $i < j \leq n$. Außerdem sind alle $p_i (i \leq n)$ Endzustände. Lauf von $\mathfrak{A}$ auf 0^ω hat Form $p_0 \xrightarrow{0^*} p_i \xrightarrow{0^*} p_j \xrightarrow{0^*} p_j \ldots$ nur Endzustände.
- Also: $\mathfrak{A}$ akzeptiert 0^ω, aber $0^\omega \notin L$.
- Bemerkung: $\Sigma^\omega \setminus L = \{0^\omega\}$ wird A-erkannt.
- Allgemein: $L \subseteq \Sigma^\omega$ E-erkennbar $\Leftrightarrow$ $\Sigma^\omega \setminus L$ A-erkennbar.

Beweis:

- $\Rightarrow$) Zu E-Automat $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ gehe über zu A-Automat $\mathfrak{A}' = (Q, \Sigma, q_0, \delta, Q \setminus F)$.
- $\Leftarrow$) Analog.

Beispiel: $L_2 := \{\alpha \in \{0, 1\}^\omega | 00 \text{ kommt in } \alpha \text{ vor, } 11 \text{ niemals}\}$

- Behauptung: L_2 weder E- noch A-erkennbar (Übung)
- Bemerkung: L_2 ist (deterministisch) Büchi-erkennbar.
- Bemerkung: Wenn L E-erkennbar oder L A-erkennbar $\Rightarrow$ L Büchi-erkennbar.

Beachte: In diesem Fall gilt für einen Lauf ρ von $\mathfrak{A}'$: ρ besucht q_f immer wieder $\Leftrightarrow$ ρ besucht q_f von gewisser Stelle an fortwährend.

- Rekurrenzeigenschaft: Immer wieder wird q_f besucht.
- Persistenzeigenschaft: Ab einem gewissen Punkt in ρ wird nur noch q_f besucht.

$\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ Büchi- bzw. co-Büchi-akzeptiert $\alpha \Leftrightarrow$ für den eindeutigen Lauf von $\mathfrak{A}$ auf α gilt:

$\forall i \exists j \geq i \rho(j) \in F \quad$ bzw. $\quad \exists i \forall j \geq i \rho(j) \in F$

Beispiel: $L_3 = (0 + 1)^* 0^\omega$ "in α nur endlich oft 1" nicht Büchi-erkennbar (bereits bewiesen), aber co-Büchi-erkennbar:

Bemerkung: $L \subseteq \Sigma^\omega$ Büchi-erkennbar $\Leftrightarrow$ $\Sigma^\omega \setminus L$ co-Büchi-erkennbar.

Erinnerung: Ein Muller-Automat hat das Format $\mathfrak{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ mit $\mathcal{F} = \{F_1, \ldots, F_k\}$, $F_i \subseteq Q$.
$\mathfrak{A}$ akzeptiert $\alpha \Leftrightarrow$ für eindeutigen Lauf ρ von $\mathfrak{A}$ auf α gilt $\mathrm{Inf}(\rho) \in \mathcal{F}$.

Definition: $\mathrm{Occ}(\rho) = \{q \in Q | \exists i \rho(i) = q\}$

Definition: $\mathfrak{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ Staiger-Wagner-akzeptiert $\alpha \Leftrightarrow$ für den eindeutigen Lauf ρ von $\mathfrak{A}$ auf α gilt $\mathrm{Occ}(\rho) \in \mathcal{F}$.

Beispiel: Automat für L_2 "Staiger-Wagner-Automat"
Mengen der besuchten Zustände $\mathcal{F} = \{A, D, E\}, \{A, B, D, F\}, \{A, D, E, F\}, \{A, B, D, E, F\}$.

Satz von Landweber: Es gibt einen Algorithmus, der zu gegebenem Muller-Automaten $\mathfrak{A}$ entscheidet, ob $L(\mathfrak{A})$ Büchi-erkennbar bzw. co-Büchi-erkennbar bzw. E-erkennbar bzw. A-erkennbar ist.

Definition: Zu $\mathfrak{A} = (Q, \Sigma, q_0, \delta)$ ist $S \subseteq Q$ Schleife, falls für $s, s' \in S$ existiert jeweils nicht-leerer Pfad von s nach s'.

Definition: Eine starke Zusammenhangskomponente (engl. strongly connected component, SCC) ist eine bzgl. Mengeninklusion maximale Schleife.

Bemerkung: Die SCCs bilden vermöge der Erreichbarkeitsrelation eine partielle Ordnung (SCC-Zerlegung).

Abschlußbedingungen: In einem Muller-Automaten $\mathfrak{A} = (Q, \ldots, \mathcal{F})$ heißt $\mathcal{F}$

1. bzgl. Oberschleifen abgeschlossen, falls Schleife $S \in \mathcal{F}, S \subseteq S'$ Schleife $\Rightarrow S' \in \mathcal{F}$.
2. bzgl. Unterschleifen abgeschlossen, falls $S \in \mathcal{F}; S' \subseteq S \Rightarrow S' \in \mathcal{F}$
3. bzgl. erreichbarer Schleifen abgeschlossen, falls $S \in \mathcal{F}$, von $q \in S$ ist Schleife S' erreichbar $\Rightarrow S' \in \mathcal{F}$.

Beweisansatz von Landweber: Sei $\mathfrak{A}$ Muller Automat. $L(\mathfrak{A})$

- Büchi-erkennbar $\Leftrightarrow$ 1
- co-Büchi-erkennbar $\Leftrightarrow$ 2
- E-erkennbar $\Leftrightarrow$ 3

Satz: L Staiger-Wagner-erkennbar $\Leftrightarrow$ L Büchi-erkennbar und L ist co-Büchi-erkennbar.

Vorbereitung: L Staiger-Wagner-erkennbar $\Leftrightarrow$ L ist Boolesche Kombination von E-erkennbaren ω-Sprachen.

Schwache Sprachen: L schwach (durch SW-Automat) erkennbar $\Leftrightarrow$ für beliebigen Muller-Automaten $\mathfrak{A}$, der L erkennt, gilt: Die Schleifen einer Zusammenhangskomponente (SCC) von $\mathfrak{A}$ sind entweder alle akzeptierend oder alle nicht-akzeptierend.

SCC-Zerlegung: Sei $\mathfrak{A}$ ein Muller-Automat gemäß Charakterisierung (also mir "positiven" SCCs und "negativen" SCCs). Sei $\mathfrak{A}'$ daraus gewonnen mit Akzeptierkomponente $F =$ Vereinigung allerdings positiven SCCs. Dann akzeptierten $\mathfrak{A}'$ als Büchi-Automat und als co-Büchi-Automat ebenfalls die Sprache $L(\mathfrak{A})$.

Satz: Zu einem nichtdeterministischen co-Büchi-Automaten kann man einen äquivalenten deterministischen co-Büchi-Automaten konstruieren.

Beweis: $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$, Anfangszustand: $(\{q_0, \varnothing\})$ [$\mathfrak{A}$ akzeptiert durch Existenz eines Laufes, der schließlich nur Endzustände hat.]

- Idee: Zustände haben Form $(P, R) \in \mathcal{P}(Q)^2$
  - P festgelegt gemäß Potenzmengenkonstruktion
  - R enthält jeweils die Zustände, die seit letztem $\varnothing$-Auftreten nur innerhal F erreicht werden können.
  - $\delta((P, R), a) = (P', R')$ mit
    - $P' = \{q | \exists p \in P, (p, a, q) \in \Delta\}$,
    - $\begin{cases} R' = \{q \in F | \exists r \in R (r, a, q) \in \Delta\} & \text{falls } R \neq \varnothing \\ R' = P' \cap F & \text{falls } R = \varnothing \end{cases}$
- Dann führt ein Lauf von $\mathfrak{A}$ mit Endzuständen von gewisser Stelle an auf einen $\mathfrak{A}'$-Lauf, in dem von gewisser Stelle an $\varnothing$ als zweite Komponente nicht auftritt.
- $F' = \{(P, R) | R \neq \varnothing\}$

Bemerkung (später mehr): Deterministische schwache Automaten (Büchi- oder co-Büchi-Automaten) lassen sich effizient minimieren.

Anwendung: Theorie erster Stufe der reellen Zahlen mit Addition, genau $(\mathbb{R}, \mathbb{Z}, +, \leq, 0)$.

- Spache:
  - Variablen $x, y, \ldots$ für reelle Zahlen
  - Relationen $\mathbb{Z}(x)$ "x ist ganz", $\leq$, $=$
  - Funktion $+$
- Anwendungsbereich: algorithmische Geometrie, zeitbehaftete und hybride Systeme
- Erwünscht z.B.: Test, ob Formel $\varphi(x_1, \ldots, x_n)$ durch Tupel $(r_1, \ldots, r_n) \in \mathbb{R}$ erfüllbar
- Beispiel-Formeln:
  - $\underbrace{y \leq \frac{1}{2} x}_{y+y \leq x} \ \wedge\ y \geq 0 \ \wedge\ x \leq 3$
  - $\exists r_1, r_2 \exists z_1 \in \mathbb{Z} \exists z_2 \in \mathbb{Z} : 0 \leq r_1 \leq r_2 \leq 1 \ \wedge\ x = 2z_1 + r_1 \ \wedge\ y = 2z_2 + r_2$

Ansatz für Erfüllbarkeitstest: Transformation → Automat. Benutze Kodierung von reellen Zahlen durch Dualdarstellung / -brüche ⇝ Zahlmengen ⇝ ω-Sprachen

Kodierung im Detail: Schreibe Dualdarstellung als Wort $\underbrace{u}_{\text{Vorkommateil}} * \underbrace{\alpha}_{\text{Nachkommateil}}$.

- Beachte: Keine Eindeutigkeit – z.B. Zahlen für $010 * 0101 \ldots = 10 * 0101 \ldots$ oder $0.111 \ldots = 1.000 \ldots$
- Konvention: Zahltupel kodiert durch $*$ an gleicher Stelle, d.h. über Alphabet $\{0, 1\}^n \cup *$, z.B. $\langle \alpha_1, \alpha_2 \rangle = \binom{0}{0}\binom{1}{0}\binom{0}{0} * \binom{0}{1}\binom{1}{1}\binom{0}{1}\binom{1}{1} \ldots$

Satz: Zu jeder Formel erster Stufe (FO) $\varphi(x_1, \ldots, x_n)$ mit $\mathbb{Z}, +, \leq, 0$ kann man einen schwachen (co-Büchi-)Automaten $\mathfrak{A}_\varphi$ konstruieren mit: Ein Tupel $(r_1, \ldots, r_n) \in \mathbb{R}^n$ erfüllt $\varphi(x_1, \ldots, x_n)$ gdw. $\mathfrak{A}_\varphi$ akzeptiert jede Kodierung $\langle \alpha_1, \ldots, \alpha_n \rangle$ von $(r_1, \ldots, r_n)$.

Beweisansatz: (analog S1S → Büchi-Automaten) induktiv über Formelaufbau:

- atomare Formeln, $\neg$, $\wedge$, $\exists$

Ende der Vorlesung