3 Automaten und Logiken auf unendlichen Wörtern

Wir wollen jetzt die Beschränkung, dass Mengenvariablen nur über endliche Mengen rangieren, fallenlassen. Wertebereich für die Mengenvariablen ist nun also $\mathcal{P}(\mathbb{N})$, die Potenzmenge von $\mathbb{N}$. Mit MSO bezeichnen wir die so entstehende Logik. Syntax der Formeln ist dieselbe wie bei der wMSO, aber die Semantik unterscheidet sich eben in der Interpretation der Mengenvariablen. So ist zum Beispiel die folgende Formel in der wMSO falsch, in der MSO aber wahr. Sie bezeichnet die Existenz der Menge der geraden Zahlen:

$$\exists C.0 \in C \wedge \forall x.x \in C \iff x+1 \notin C$$

Gerade wenn man Zeit modellieren möchte, ist die MSO besser geeignet als die wMSO. Wir werden im folgenden zeigen, dass auch die MSO entscheidbar ist, sogar mit derselben Komplexität wie die wMSO. In der Praxis allerdings verhalten sich die Entscheidungsverfahren für die MSO weniger gut und die hauptsächliche Bedeutung der MSO ist theoretischer Natur: hat man irgendeine Logik, die sich in der MSO kodieren lässt, so weiß man dann mit dem noch zu beweisenden Resultat, dass diese Logik entscheidbar ist.

Hat man eine MSO-Formel mit freien Mengenvariablen $P_1, \ldots, P_n$ so lässt sich eine Belegung dieser Mengenvariablen i.a. nicht mehr durch ein endliches Wort repräsentieren—unendliche Wörter sind erforderlich. Das Entscheidungsverfahren für die MSO ist analog zu dem für die wMSO mit dem Unterschied, dass endliche Automaten auf *unendlichen Wörtern* zum Einsatz kommen. Diesen wenden wir uns nunmehr zu.

3.1 Unendliche Wörter

Sei Σ ein endliches Alphabet. Ein *unendliches Wort* über Σ ist eine unendliche Folge $w = a_0a_1a_2\ldots$ von Buchstaben $a_i \in \Sigma$. Man bezeichnet die Menge der unendlichen Wörter mit Σ^ω. Man kann die Konkatenation w_1w_2 von endlichen und unendlichen Wörtern definieren, wobei $w_1w_2 = w_1$ falls w_1 unendlich ist. Unendliche Wörter heißen auch ω-Wörter. Eine Teilmenge von Σ^ω heißt ω-Sprache. Ist $A \subseteq \Sigma^*$ und $B \subseteq \Sigma^\omega$, so setzt man $AB = \{ab \mid a \in A, b \in B\}$ und $A^\omega = \{w_0w_1w_2\cdots \mid w_i \in A\}$. Zum Beispiel ist $((aa)^*b)^\omega$ die ω-Sprache aller Wörter, in der jeweils nach einer geraden Anzahl von a's ein b kommt. Man kann mit $\Sigma, \emptyset, \epsilon, \cup, (-)^*, (-)^\omega$ in der naheliegenden Weise ω-reguläre Ausdrücke definieren und bezeichnet die von ihnen beschriebenen ω-Sprachen als ω-regulär.

3.2 Büchi Automaten

Ein *Büchiautomat* ist genau dasselbe wie ein NFA, also ein Tupel $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ wobei Q, Σ endliche Mengen sind, $q_0 \in Q$ (Anfangszustand), $\delta : Q \times \Sigma \to 2^Q$ und $F \subseteq Q$.

Ist solch ein Büchiautomat gegeben und außerdem ein unendliches Wort $w = a_1 a_2 a_3 \ldots$, so ist ein *Lauf* von A auf w eine unendliche Folge von Zuständen $q_0 q_1 q_2 \ldots$, beginnend mit dem Anfangszustand q_0, sodass $q_i \in \delta(q_{i-1}, a_i)$ für alle $i \geq 1$. Ein solcher Lauf ist *akzeptierend*, wenn ein $q \in F$ existiert, derart dass $q_i = q$ für unendliche viele i. Mit anderen Worten ist der Lauf akzeptierend, wenn er immer wieder die Endzustandsmenge besucht. Beachte: wird die endliche Endzustandsmenge unendlich oft besucht, so gibt es auch einen festen Endzustand, der unendlich oft besucht wird. Die Sprache $L(\mathcal{A})$ des Büchiautomaten ist definiert als die Menge aller ω-Wörter, für die ein akzeptierender Lauf existiert. Wohlgemerkt ist nicht verlangt, dass jeder Lauf akzeptierend sein muss; nur, dass es eben einen gibt.

Beispiele: Sei $\Sigma = \{a, b\}$ und $L_1 = (a^*b)^\omega$ die Sprache aller Wörter, die unendlich viele b's, also immer wieder b's, zeigen. Ein Büchiautomat $\mathcal{A}_1$ mit $L(\mathcal{A}_1) = L_1$ ist gegeben durch $Q = \{q_0, q_1\}, \delta(q_0, a) = \{q_0\}, \delta(q_0, b) = \{q_1\}, \delta(q_1, a) = \{q_0\}, \delta(q_1, b) = \{q_1\}, F = \{q_1\}$. Der Zustand q_1 zeigt an, das der aktuelle Buchstabe b lautet. Interessiert man sich für unendlich viele b's, so muss man q_1 unendlich oft besuchen. Dieser Automat ist sogar deterministisch.

Sei jetzt $L_2 = (a \cup b)^* a^\omega$ die Sprache der Wörter, die nur endlich viele b's enthalten. Beachte, dass L_2 das Komplement von L_1 ist. Ein Büchiautomat $\mathcal{A}_2$ mit $L(\mathcal{A}_2) = L_2$ ist gegeben durch $Q = \{q_0, q_1\}, \delta(q_0, a) = \{q_0, q_1\}, \delta(q_0, b) = \{q_0\}, \delta(q_1, a) = \{q_1\}, \delta(q_1, b) = \emptyset, F = \{q_1\}$. Will man ein Wort akzeptieren, muss man irgendwann nach q_1 wechseln, aber man sollte sicher sein, dass dann nicht noch ein b kommt, denn sonst bleibt man stecken. Der Automat $\mathcal{A}_2$ ist nichtdeterministisch und es ist klar, dass kein deterministischer Büchiautomat die Sprache L_2 erkennen kann. Solch einer müsste ja quasi hellsehen können, also ob z.B. nach 100000 a's jetzt wirklich Schluss ist mit den b's.

Jetzt sei $\Sigma = \{a, b, c\}$ und L_3 die Sprache aller Wörter, in denen auf jedes b irgendwann ein c folgt. Denke b = Tür auf, c = Tür zu. Es ist

$$L_3 = (a \cup c)^\omega \cup (a \cup b \cup c)^* c (a \cup c)^\omega \cup (a \cup b \cup c)^* (b(a \cup b \cup c)^* c)^\omega$$

Ein passender Büchiautomat $\mathcal{A}_3$ ist gegeben durch $Q = \{q_0, q_1\}$. $\delta(q_0, b) = \{q_1\}, \delta(q_0, a) = \delta(q_0, c) = \{q_0\}, \delta(q_1, c) = \{q_0\}, \delta(q_1, a) = \delta(q_1, b) = \{q_1\}, F = \{q_0\}$.

Satz 12
Eine Sprache L ist ω-regulär genau dann, wenn ein Büchi Automat $\mathcal{A}$ existiert mit $L(\mathcal{A}) = L$.

Beweis Sei ein ω-regulärer Ausdruck gegeben. Unter Verwendung von Rechenregeln wie

$$U^\omega V = \textit{if } \epsilon \in U \textit{ then } V \cup U^* V \cup U^\omega \textit{ else } U^\omega$$

können wir annehmen, dass $L = \bigcup_{i=1}^n A_i B_i^\omega$ für reguläre Sprachen A_i, B_i ist.

Seien jetzt NFAs $\mathcal{A}_i, \mathcal{B}_i$ mit $L(\mathcal{A}_i) = A_i$ und $L(\mathcal{B}_i) = B_i$ gegeben. Man kombiniert diese Automaten so, als wollte man nach der bekannten Konstruktionsvorschrift einen NFA für $\bigcup_i A_i B_i^*$ bauen. Der so erhaltene Automat aufgefasst als Büchiautomat erkennt L.

Ist umgekehrt ein Büchiautomat $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ gegeben, so bezeichne $L_{q,q'}$ die reguläre Sprache aller Wörter, für die ein endlicher Lauf von q nach q' existiert. Es ist

$$L(\mathcal{A}) = \bigcup_{q \in F} L_{q_0,q} L_{q,q}^{\omega}$$

Die ω-regulären Sprachen sind per Definition unter $\cup, (-)^*, (-)^\omega$ abgeschlossen, also sind auch die Büchi erkennbaren Sprachen unter diesen Operationen abgeschlossen.

Satz 13
Die ω-regulären Sprachen sind unter Durchschnitt abgeschlossen.

Beweis Seien Büchiautomaten $\mathcal{A}$ und $\mathcal{B}$ gegeben. Einen Büchiautomaten für $L(\mathcal{A}) \cap L(\mathcal{B})$ konstruiert man wie folgt. Zunächst schaltet man $\mathcal{A}$ und $\mathcal{B}$ parallel gemäß der bekannten Produktkonstruktion. Zusätzlich führt man zwei Lampen ein, eine für $\mathcal{A}$, eine für $\mathcal{B}$. Formal ist also der Zustandsraum des konstruierten Automaten $Q_\mathcal{A} \times Q_\mathcal{B} \times \{0,1\} \times \{0,1\}$. Ein Wort wird nun auf $\mathcal{A}$ und auf $\mathcal{B}$ parallel verarbeitet. Kommt $\mathcal{A}$ oder $\mathcal{B}$ in einen Endzustand, so wird die entsprechende Lampe eingeschaltet. Sind beide Lampen an, so werden sie im nächsten Schritt beide wieder ausgemacht. Endzustände sind alle Zustände, in denen beide Lampen an sind. ∎

Seien Σ, Δ Alphabete und $h : \Sigma \to \Delta$ eine Funktion. Man erweitert h auf endliche und unendliche Wörter in der üblichen Weise zu einem Homomorphismus. Ist $L \subseteq \Sigma^\omega$ so bezeichnet $h(L) \subseteq \Delta^\omega$ die ω-Sprache $\{h(w) \mid w \in L\}$.

Satz 14
Seien Σ, Δ Alphabete und $h : \Sigma \to \Delta$. Ist $L \subseteq \Sigma^\omega$ ω-regulär, so auch $h(L)$.

Beweis Ist $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein Automat für L so erhält man einen für $h(L)$ als $(Q, \Delta, q_0, \delta', F)$,

$$\delta'(q, a) = \{q' \mid \exists b \in \Sigma. h(b) = a \wedge q' \in \delta(q, b)\}$$

Die ω-regulären Sprachen sind wie die regulären auch unter Bildern und inversen Bildern beliebiger Homomorphismen abgeschlossen. Wir verzichten auf explizite Angabe der entsprechenden Konstruktionen.

Wie wir schon gesehen haben, lassen sich Büchiautomaten nicht determinisieren, denn $(a \cup b)^* a^\omega$ wird von keinem deterministischen Büchiautomaten erkannt. Dennoch sind die ω-regulären Sprachen unter Komplement abgeschlossen, wie wir im folgenden zeigen werden.

3.2.1 Zwei Sätze der unendlichen Kombinatorik

Die Komplementierung von Büchiautomaten nach Büchi verwendet zwei kombinatorische Sätze: Königs Lemma und den Satz von Ramsey, die wir hier der Vollständigkeit halber mit Beweis angeben.

Lemma 6 (Königs Lemma)
In einem Baum mit unendlich vielen Knoten, aber endlichem Verzweigungsgrad (kein Knoten hat unendlich viele Kinder) gibt es einen unendlichen Pfad.

Beweis Wir konstruieren induktiv eine Folge von Knoten $a_0, a_1, \ldots$, so dass a_{i+1} Kind von a_i ist und unterhalb a_i jeweils unendlich viele Knoten liegen, also a_i unendlich viele Nachkommen hat. Man beginnt mit der Wurzel a_0, diese hat nach Voraussetzung unendlich viele Nachkommen. Sind $a_0, \ldots, a_n$ schon gefunden, so wählt man a_{n+1} als eines derjenigen Kinder von a_n, das unendlich viele Nachkommen hat. Ein solches muss vorhanden sein, denn sonst hätte a_n selbst nur endlich viele Nachkommen. In dieser Weise entsteht wie verlangt ein unendlicher Pfad. ∎

Hier ist eine Anwendung von Königs Lemma. Gegeben sei ein Satz von quadratischen Kacheln mit gefärbten Kanten. Zwei Kanten passen zusammen, wenn sie dieselbe Farbe haben. Es ist unentscheidbar, ob man mit einem gegebenen Satz die Ebene pflastern kann. Immerhin gilt folgendes: kann man mit einem Satz von Kacheln einen Quadranten pflastern, so kann man damit auch die ganze Ebene pflastern. Achtung, man kann nicht einfach die vier Quadranten pflastern; an den Nähten könnten ja dann die Farben nicht stimmen.

Man arrangiert Pflasterungen von Quadraten der Kantenlänge $2i$ als Baum, indem die Kinder einer Pflasterung der Größe $2i \times 2i$ diejenigen der Größe $(2i+2) \times (2i+2)$ sind, die die gegebene um einen äußeren Ring erweitern. Die leere Pflasterung des 0×0 Quadrates bildet die Wurzel. Der Baum hat unendlich viele Knoten, da nach Voraussetzung Quadrate beliebiger Größe pflasterbar sind. Ein unendlicher Pfad, der nach Königs Lemma existiert, definiert die gesuchte Pflasterung der Ebene.

Für eine Menge A bezeichne $\binom{A}{2}$ die Menge der zweielementigen Teilmengen von A.

Satz 15 (Ramsey)
Sei A eine unendliche Menge, C eine endliche Menge von "Farben" und $f : \binom{A}{2} \to C$ eine "Färbung" der zweielementigen Teilmengen von A mit k "Farben". Es existiert eine unendliche Teilmenge B von A, sodass f eingeschränkt auf $\binom{B}{2}$ konstant ist, d.h. es gibt eine feste Farbe $c \in C$, sodass $f(\{x,y\}) = f(\{u,v\})$ für alle $x, y, u, v \in B$.

Beweis Zunächst müssen wir A anordnen (wir können o.B.d.A A mit $\mathbb{N}$ identifizieren und die normale Ordnung verwenden). Sollte A überabzählbar sein, dann schränken wir zunächst willkürlich auf eine abzählbare Teilmenge ein. Außerdem können wir o.B.d.A. annehmen, dass $C = \{1, \ldots, k\}$.

Wir konstruieren jetzt eine Folge $(a_0, A_0, c_0), (a_1, A_1, c_1), (a_2, A_2, c_2), \ldots$ sodass $a_0 < a_1 < a_2 < \ldots$ und $A_0 \supseteq A_1 \supseteq A_2 \supseteq \ldots$ und so dass die A_i alle unendlich sind und $f(\{a_i, x\}) = c_i$ für alle $x \in A_i$ und $a_{i+1} \in A_i$ für alle i.

Wir wählen a_0 beliebig und c_0 als eine Farbe, die unendlich viele Paare der Form $\{a_0, x\}$ färbt und dann $A_0 = \{x \mid f(\{a_0, x\}) = c_0$ Das geht, weil es nur endlich viele Farben gibt.

Ist die Folge schon bis n gebildet, dann nehmen wir aus A_n wiederum ein beliebiges Element a_{n+1} und verschaffen uns A_{n+1}, c_{n+1} wie bei (a_0, A_0, c_0).

Eine Farbe taucht nun unendlich oft auf in der Folge der c_i.

Die Menge derjenigen a_n's, die diese Farbe haben, leistet das Verlangte. ■

3.2.2 Komplementierung von Büchiautomaten

Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein Büchiautomat. Wir wollen einen Büchiautomaten $\mathcal{A}'$ konstruieren, sodass $L(\mathcal{A}) = \overline{L(\mathcal{A}')}$. Für Wort $u \in \Sigma^*$ führen wir folgende Notationen ein:

$u : q \to q'$ bedeutet: von Zustand q kommt man unter Abarbeitung von u im Automaten nach q'.

$u : q \to_F q'$ bedeutet: von Zustand q kommt man unter Abarbeitung von u im Automaten nach q' und durchläuft dabei einen Endzustand.

Wenn $u : q \to_F q'$ dann natürlich auch $u : q \to q'$. Wenn $u : q \to q_1$ und $v : q_1 \to q'$ und $q_1 \in F$, dann auch $uv : q \to_F q'$.

Wir definieren eine Äquivalenzrelation $\sim \subseteq \Sigma^* \times \Sigma^*$ wie folgt:

$$u \sim v \iff \forall q\ q'.(u : q \to q' \Leftrightarrow v : q \to q') \wedge (u : q \to_F q' \Leftrightarrow v : q \to_F q')$$

Lemma 7
Die Äquivalenzrelation $\sim$ hat endlichen Index, das heißt, sie hat nur endlich viele Klassen.

Beweis Jede Klasse ist eindeutig bestimmt durch zwei Mengen von Zustandspaaren, also gibt es höchstens $2^{2|Q|^2}$ viele Klassen. ■

Lemma 8
Jede Äquivalenzklasse der Relation $\sim$ ist regulär.

Beweis Für Zustände q, q' sei $L_{q,q'} = \{u \mid u : q \to q'\}$ (vgl. Beweis von Satz 12) und $L^F_{q,q'} = \{u \mid q \to_F q'\}$. Man sieht leicht (Übung), dass jede Klasse regulärer Ausdruck in den (offensichtlich regulären) $L_{q,q'}$ und $L^F_{q,q'}$ ist. ■

Lemma 9
Seien U, V zwei Klassen von $\sim$ und sei $w \in UV^\omega$. Wenn $w \in L(\mathcal{A})$, so ist sogar UV^ω eine Teilmenge von $L(\mathcal{A})$.

Beweis Wir zerlegen das Wort w als $w = uv_1v_2v_3v_4\ldots$ mit $u \in U$ und $v_i \in V$ und betrachten einen akzeptierenden Lauf. Aus der Annahme, dass U, V Äquivalenzklassen sind können wir dann einen akzeptierenden Lauf für jedes andere Wort aus UV^ω konstruieren. ■

Lemma 10
Seien U, V zwei Klassen von $\sim$ und $w \in UV^\omega$. Wenn w in $\overline{L(\mathcal{A})}$, so ist sogar UV^ω eine Teilmenge von $\overline{L(\mathcal{A})}$.

Beweis Wäre $w' \in UV^\omega$ in $L(\mathcal{A})$, so auch w selbst wegen Lemma 9. ∎

Lemma 11
Für jedes beliebige Wort $w = a_0a_1a_2\cdots \in \Sigma^\omega$ existieren Äquivalenzklassen U, V von $\sim$, sodass $w \in UV^\omega$.

Beweis Jedes endliche Wort liegt in einer Klasse (nämlich "seiner" Klasse). Wir betrachten folgende Färbung von $\binom{\mathbb{N}}{2}$. Setze $f(\{i,j\})$ gleich der Äquivalenzklasse von $a_i \dots a_{j-1}$, wobei o.B.d.A. $i < j$. Der Satz von Ramsey liefert eine Äquivalenzklasse V und eine unendliche Teilmenge $S \subseteq \mathbb{N}$, sodass $i, j \in S, i < j$ impliziert $a_i \dots a_{j-1} \in V$. Wenn jetzt i_0 das kleinste Element von S ist und U die Äquivalenzklasse von $a_0 \dots a_{i_0-1}$ ist, dann folgt $w \in UV^\omega$. ∎

Satz 16 (Büchi)
Sei $\mathcal{A}$ ein Büchiautomat. Die Sprache $\overline{L(\mathcal{A})}$ ist ω-regulär.

Beweis Aus dem vorher Gesagten folgt

$$\overline{L(\mathcal{A})} = \bigcup \{UV^w \mid U, V \in \Sigma^*/\sim \wedge\ UV^\omega \cap L(\mathcal{A}) = \emptyset\}$$

Damit aber liegt ein regulärer Ausdruck vor. ∎

Es folgt, dass ein Automat $\mathcal{A}'$ existiert, der gerade das Komplement von $L(\mathcal{A})$ erkennt.

Will man diesen effektiv berechnen, so kann man wie folgt vorgehen: Die endlich vielen Äquivalenzklassen werden durch Repräsentanten dargestellt und aus dem Automaten abgelesen. Durch Tiefensuche im Automaten kann man dann diejenigen Klassen U, V identifizieren, für die UV^ω disjunkt von $L(\mathcal{A})$ ist. Wegen Lemma 9 und 10 ist das ja gleichbedeutend damit, dass uv^ω nicht in $L(\mathcal{A})$ ist, wobei $u \in U$ und $v \in V$ beliebige Repräsentanten sind.

Dies liefert einen Algorithmus zur Konstruktion von $\mathcal{A}'$. Allerdings hat der so gewonnene Automat $2^{2^{O(|\mathcal{A}|)}}$ Zustände.

3.3 Entscheidbarkeit der MSO

Wir erinnern uns, dass die monadische Logik zweiter Stufe (MSO) dieselbe Syntax wie wMSO hat, aber Mengenvariablen und damit auch Formeln beliebige also möglicherweise unendliche Teilmengen von $\mathbb{N}$ bezeichnen.

Satz 17 (Büchi)
Es existiert ein effektives Verfahren, das zu gegebener MSO Formel ϕ entscheidet, ob sie erfüllbar ist.

Beweis Der Beweis des Satzes von Büchi erfolgt ganz analog zum Beweis des Satzes von Büchi-Elgot: zu gegebener MSO-Formel ϕ mit freien Mengenvariablen $X_0, \ldots, X_{n-1}$ (möglicherweise $n = 0$, also ϕ geschlossen) konstruiert man einen Büchiautomaten $\mathcal{A}_\phi$ über dem Alphabet $2^{\{0,\ldots,n-1\}}$ (also einelemeniges Alphabet wenn $n = 0$), derart, dass $L(\mathcal{A}_\phi) = \{w \mid w \models \phi\}$, wobei $w \models \phi$ bedeutet, dass ϕ wahr ist, wenn man die Variable X_i als die Menge $\{p \mid i \in a_p\}$ interpretiert, wobei $w = a_0 a_1 a_2 \ldots$. Man liest also das Wort w als unendliche Wertetabelle für die X_i.

Die Konstruktion des Automaten erfolgt durch Induktion über den Aufbau von ϕ. Die Boole'schen Operationen gehen wie üblich (Produktautomat, Komplementierung). Existenzquantoren behandelt man mit Satz 14 unter Einführung von Nichtdeterminismus.■

Wie schon gesagt wird bei der Komplementierung die Zustandszahl (doppelt) exponentiell erhöht, also auch bei der Behandlung von Allquantoren, die man ja als $\neg\exists\neg$ auffasst. Dies führt wie bei Büchi-Elgot zu einem Verfahren der Komplexität $DTIME(2_{O(n)})$.

In der Praxis ist aber dieses Verfahren wesentlich schwieriger zu implementieren als das Verfahren für die wMSO und meines Wissens existiert überhaupt keine allgemein verfügbare Implementierung. Unter anderem wird dies daran liegen, dass es für Büchiautomaten kein Minimierungsverfahren gibt. Allerdings gibt es in letzter Zeit einige erfolgversprechende Ansätze. So hat Thomas Wilke Arbeiten zur Minimierung von Büchiautomaten verfasst und Vardi-Kupferman haben einen alternativen Ansatz mit alternierenden Paritätsautomaten ins Spiel gebracht.

3.4 Determinisierung von Büchiautomaten

Der Titel dieses Abschnitts klingt paradox; haben wir nicht bereits gesehen, dass Büchiautomaten nicht ohne Nichtdeterminismus auskommen?

Die Antwort liegt in der Akzeptanzbedingung; der Automat $Q = \{q_0, q_1\}$, $\delta(q_0, a) = \delta(q_1, a) = \{q_0\}, \delta(q_0, b) = \delta(q_1, b) = \{q_1\}$ "akzeptiert" $(a \cup b)^* a^\omega$ wenn wir festlegen, dass ein Wort w akzeptiert ist, wenn der Lauf des Automaten auf w den Zustand q_0 unendlich oft besucht, aber den Zustand q_1 nur endlich oft.

Automaten mit solchen Akzeptanzbedingungen heißen Rabinautomaten und bilden einen Spezialfall der Mullerautomaten.

Definition 12
Ein Mullerautomat ist ein Tupel $(Q, \Sigma, q_0, \delta, \mathcal{F})$, wobei Q, Σ, q_0, δ wie bei NFA sind und $\mathcal{F}$ eine Menge von Zustandsmengen ist. Diese Menge $\mathcal{F}$ heißt Akzeptanzbedingung des Mullerautomaten.

Ein Wort w wird von solch einem Mullerautomaten $\mathcal{A}$ akzeptiert, wenn ein Lauf ρ von $\mathcal{A}$ auf w existiert, sodass die Menge der in ρ unendlich oft vorkommenden Zustände in $\mathcal{F}$ aufgeführt ist. Mit $L(\mathcal{A})$ bezeichnet man die Menge der in diesem Sinne akzeptierten Wörter.

Der o.a. Automat ist ein Mullerautomat mit $\mathcal{F} = \{\{q_0\}\}$.

Jeder Büchiautomat mit Endzustandmenge F kann auch als Mullerautomat mit Akzeptanzbedingung $\mathcal{F} = \{M \mid M \cap F \neq \emptyset\}$ aufgefasst werden.

Definition 13
Ein Rabinautomat ist ein Mullerautomat, derart dass Zustandsmengen $U_1, \ldots, U_n, V_1, \ldots, V_n$ existieren, derart dass $F \in \mathcal{F} \iff \exists i \leq n. U_i \cap F \neq \emptyset \wedge V_i \cap F = \emptyset$.

Ein Paar (U_i, V_i) heißt Rabinbedingung des Automaten. Der Rabinautomat akzeptiert ein Wort, wenn ein Lauf und eine Rabinbedingung (U, V) des Automaten existiert, sodass ein Zustand aus U unendlich oft besucht wird, aber kein Zustand aus V unendlich oft besucht wird, also alle Zustände in V nach endlicher Zeit überwunden sind.

Satz 18
Jede von einem Mullerautomaten erkannte Sprache ist ω-regulär.

Beweis Hat man einen Mullerautomaten mit Akzeptanzbedingung $\mathcal{F}$ gegeben, so kann man wie folgt einen äquivalenten Büchiautomaten konstruieren. Der Büchiautomat rät zunächst nichtdeterministisch eine Menge $M \in \mathcal{F}$. Nun arbeitet er wie der gegebene Muller Automat, führt aber zusätzlich noch eine dynamische Menge S von Zuständen, initialisiert mit $\emptyset$, mit. Zunächst bleibt diese Menge unverändert. Nach einer Weile aber schaltet der Büchiautomat nichtdeterministisch um und möchte ab da nur noch Zustände in M sehen und jeden dieser Zustände unendlich oft. Hat also der Büchiautomat diese Transition gemacht, so wird ein durchlaufener Zustand aus M jeweils in S hineingegeben und immer wenn $S = M$ eine Lampe geblitzt (also per Definition ein Endzustand durchlaufen) und S auf $\emptyset$ zurückgesetzt. Kommt in dieser zweiten Phase doch noch wider Erwarten ein Zustand aus $Q \setminus M$, so wird man die Verarbeitung sofort abbrechen. ∎

Interessant an den Mullerautomaten ist, dass sie auch in ihrer deterministischen Version alle ω-regulären Sprachen zu erkennen in der Lage sind und dann natürlich leicht komplementiert werden können.

Für den Rest dieses Abschnitts sei $\mathcal{A}$ ein Büchiautomat. Wir wollen einen deterministischen Mullerautomaten $\mathcal{M}$ konstruieren, sodass $L(\mathcal{A}) = L(\mathcal{M})$. Diese Konstruktion wurde 1989 von S. Safra in seiner Dissertation vorgestellt und ist deshalb als *Safrakonstruktion* bekannt.

Es ist nützlich, sich den Mullerautomaten $\mathcal{M}$ imperativ vorzustellen: In diesem Sinne ist der momentane Zustand von $\mathcal{M}$ ein Baum von Prozessen, wobei ein Prozess besteht aus

- einer eindeutigen Nummer PID
- einer Menge von $\mathcal{A}$-Zuständen
- einem Zeiger auf den Vaterprozess (außer beim Wurzelprozess) und Zeigern auf die Kinderprozesse, falls vorhanden
- einer Lampe

Zusätzlich wird eine lineare Ordnung aller momentan aktiven Prozesse mitgeführt, sodass man feststellen kann, welcher eher da war.

Am Anfang gibt es nur einen Prozess beschriftet mit $\{q_0\}$.

Das Fortschalten beim Lesen des Symbols a erfolgt durch sukzessives Ausführen der folgenden Schritte:

1. Sollte ein Prozess Endzustände enthalten, so wird ein neuer Kindprozess erzeugt beschriftet mit diesen Endzuständen. Die Endzustände verbleiben aber auch beim Elternprozess.

2. Jeder Prozess (einschließlich der neu erzeugten) wird gemäß der Potenzmengenkonstruktion weitergeschaltet. Man ersetzt also seine Beschriftung B durch $\bigcup_{q \in B} \delta(q, a)$.

3. Die Beschriftungen der Kinder sollen disjunkt sein: kommt ein Zustand in beiden vor, so muss der jüngere Prozess verzichten. Sollte dabei die Beschriftung leer werden, so wird der Prozess samt seinen (ebenfalls leeren) Kindern und Kindeskindern entfernt.

4. Sollten die Beschriftungen aller Kinder die Beschriftung des Elternknotens ergeben, so werden alle Kinder und Kindeskinder entfernt und die Lampe des Elternknotens kurz aufgeblitzt.

Ein Wort ist akzeptiert, wenn im entsprechenden Lauf ein Knotenname ab einem gewissen Zeitpunkt immer vorhanden ist (nicht entfernt wird) und immer wieder blitzt.

Die folgenden Invarianten lassen sich beobachten:

1. Die Beschriftungen der Kinder bilden zusammen immer eine echte Teilmenge der Beschriftung des Elternknotens.

2. Die Wurzelbeschriftung entspricht der naiven Potenzmengenkonstruktion (die ja für Büchiautomaten nicht funktioniert.)

Formal ist die Zustandsmenge von $\mathcal{M}$ als die Menge der *Safrabäume* erklärt:

Definition 14
Sei eine Zustandsmenge Q gegeben. Ein Safrabaum über Q ist ein Tupel $(K, <, p, l, !)$, wobei folgendes gilt.

- K (Knotenmenge) ist eine Teilmenge von $\{1 \dots 2|Q|\}$.
- "$<$" ist eine lineare Ordnung auf K. ($k_1 < k_2$ bedeutet, dass k_1 älter als k_2 ist.)
- $p : K \to K$ (Elternfunktion) ist so, dass $p(r) = r$ für genau ein $r \in K$. Dieses r muss der älteste Knoten sein (kleinster Knoten bzgl $<$).
- $l : K \to 2^Q$ (Beschriftung)
- $p(k_1) = p(k_2) \Rightarrow l(p_1) \cap l(p_2) = \emptyset$.
- $l(k) \subseteq l(p(k))$
- $\bigcup_{k:p(k)=x} l(k)$ ist echte Teilmenge von $l(x)$.
- $! : K \to \{\mathtt{tt}, \mathtt{ff}\}$. Wenn $!k$, dann hat k keine Kinder (es gibt kein x mit $p(x) = k$).

Bemerkung: Ein Safrabaum hat höchstens $|Q|$ Knoten, somit sind stets mindestens $|Q|$ Knotennamen unbenutzt.

Mit $S(Q)$ bezeichnen wir die Menge der Safrabäume über Q. Es gilt $|S(Q)| = O(2^{cn \log n})$ für festes c.

Definition 15
Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein Büchiautomat. Die Übergangsfunktion $\delta^S(\mathcal{A}) : S(Q) \times \Sigma \to S(Q)$ ist wie folgt definiert. Sei t ein Safrabaum, a ein Eingabesymbol. Der Safrabaum $t' = (K', p', l', !') = \delta^S(\mathcal{A})(t, a)$ wird aus $t = (K, p, l, !)$ gemäß folgender Schritte berechnet:

- Ist $l(k) \cap F \neq \emptyset$ für einen Knoten k, so führe neues Kind mit frischem Namen ein und beschrifte es mit $l(k) \cap F$. Adjustiere die Altersrelation, sodass das neue Kind der jüngste Knoten wird.
- Wende die Potenzmengenkonstruktion auf alle Knotenbeschriftungen an, d.h., ersetze $l(k)$ durch $\bigcup_{q \in l(k)} \delta(q, a)$.
- Erscheint ein Zustand in mehreren Kindern desselben Knotens, so belasse ihn nur im ältesten derselben. Entferne Knoten, die dabei leer werden.
- Entsprechen die Beschriftungen aller Kinder eines Knotens k der Beschriftung von k, so entferne die Kinder und setze $!'(k) = \texttt{tt}$.
- Für alle anderen Knoten setze $!'(k) = \texttt{ff}$, selbst wenn $!(k) = \texttt{tt}$.

Der deterministische Mullerautomat $\mathcal{M}$ wird nun wie folgt gebildet:

Definition 16
Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein Büchiautomat. Wir definieren den zugehörigen deterministischen Mullerautomaten $\mathcal{M}$ als $(S(Q), \Sigma, t_0, \delta^S(\mathcal{A}), \mathcal{F})$, wobei:

- $t_0 = (\{1\}, l, p, !)$ mit $l(1) = \{q_0\}$, $p(1) = 1$, $!(1) = \texttt{ff}$
- Eine Menge M von Safrabäumen ist in $\mathcal{F}$, wenn es einen Knotennamen k gibt, der in allen Bäumen aus M vorhanden ist und ein Baum in M ist, in dem $!(k) = \texttt{tt}$.

Beobachtung: Gilt $w : t_0 \to t$ im Mullerautomaten, so auch $w : q_0 \to q$ für jeden Zustand q der in t erwähnt wird.

Beobachtung: Gilt $w : t \to t'$ in $\mathcal{M}$ und hat k in t keine Kinder und bleibt k im gesamten Lauf bis t' am Leben und hat k in t' dann ein Kind n, so existiert für jeden Zustand $q' \in l(n)$ ein Zustand $q \in l(k)$ mit $w : q \to_F q'$.

Folgerung: Gilt $w : t \to t'$ in $\mathcal{M}$ und hat k in t keine Kinder und bleibt k im gesamten Lauf bis t' am Leben und ist $!(k) = \texttt{tt}$ in t', so existiert für jeden Zustand $q' \in l(k)$ ein Zustand $q \in l(k)$ mit $w : q \to_F q'$.

Lemma 12 (Lauflemma)
Seien $U_0 = q_0, U_1, U_2, \ldots$ Teilmengen von Q, seien $u, v_1, v_2, v_3, \ldots$ endliche Wörter, sodass gilt:

- *für alle* $q \in U_1$ *gilt* $u : q_0 \to q$
- *für alle* $q' \in U_{i+1}$ *existiert* $q \in U_i$*, sodass* $v_i : q \to_F q'$

Dann ist $uv_1v_2v_3 \cdots \in L(\mathcal{A})$.

BEWEIS Konstruiere einen Baum, dessen Knoten mit Zuständen beschriftet sind. An der Wurzel steht q_0, die Knoten der Tiefe i sind mit den Zuständen aus U_i beschriftet. Zustand q' auf Niveau $i+1$ ist Kind von Zustand q auf Niveau i, wenn gilt $v_i : q \to_F q'$. Der Baum ist unendlich, da auf jedem Niveau Knoten vorhanden sind. Somit gibt es nach Königs Lemma einen unendlichen Pfad, der einen akzeptierenden Lauf in $\mathcal{A}$ definiert. ■

Satz 19
$L(\mathcal{M}) \subseteq L(\mathcal{A})$.

BEWEIS Sei $w \in \Sigma^\omega$ und $w \in L(\mathcal{M})$. Sei k ein Knotenname k, der im Lauf von $\mathcal{M}$ auf w ab einem gewissen Zeitpunkt immer präsent ist und unendlich oft markiert ist ("blitzt").

Sei t_i der Safrabaum beim i-ten Blitzen von k nach k's letztmaligem Verschwinden. Sei U_i die Beschriftung von k in t_i und v_i das Segment von w, das von t_i bis t_{i+1} abgearbeitet wird. Sei weiter u das Präfix von t, das bis zu t_1 abgearbeitet wird. Das Lauflemma zusammen mit der Folgerung liefert $w \in L(\mathcal{A})$. ■

Satz 20
$L(\mathcal{A}) \subseteq L(\mathcal{M})$.

BEWEIS Sei $w \in \Sigma^\omega$ und $r = q_0q_1q_2 \ldots$ ein akzeptierender Lauf von $\mathcal{A}$ auf w. Der Lauf wird auf jeden Fall in der Wurzel nachgebildet. Falls die Wurzel unendlich oft blitzt, dann sind wir fertig. Ansonsten sei f ein Endzustand, der unendlich oft in r vorkommt. Betrachte das erste Vorkommen von f in r nach dem letztmaligen Blitzen der Wurzel. Zu diesem Zeitpunkt wird ein neues Kind der Wurzel eingerichtet, dessen Beschriftung f enthält. Wir verfolgen den Lauf nun in diesem Kind. Es könnte sein, dass ein älteres Kind irgendwann (möglicherweise gleich am Anfang) den Lauf von diesem Kind übernimmt oder das Kind gar verschwindet. Dann konzentrieren wir uns entsprechend auf dieses ältere Kind usw. und finden auf diese Weise ein Kind der Wurzel, in dem der restliche Lauf nachgebildet wird. Sollte auch dieses nicht immer wieder blitzen, so finden wir mit demselben Argument untterhalb von ihm ein Kind, das nicht mehr verschwindet und den gesamten Restlauf nachbildet etc. Nachdem aber die Safrabäume eine beschränkte Tiefe haben, kann das nicht ewig so weitergehen und wir finden einen Knoten, der am Leben bleibt und immer wieder blitzt.

Etwas formaler argumentieren wir wie folgt: mit $t_0, t_1, t_2, \ldots$ bezeichnen wir die Folge der Safrabäume im (eindeutig bestimmten) Lauf von $\mathcal{M}$ auf w. Wir schreiben $t_i = (K_i, <_i, p_i, l_i, !_i)$. Ein Knoten k heiße *persistent*, wenn er ab einem gewissen Zeitpunkt i nicht mehr verschwindet, also $k \in K_j$ für alle $j \geq i$. Der *Beginn* $B(k)$ eines persistenten Knotens k ist das kleinste i, sodass $k \in K_j$ für alle $j \geq i$. Sind k, k' persistent und gilt $k <_j k'$ für ein $j \geq \max(B(k), B(k'))$, so folgt $k <_j k'$ für alle $j \geq \max(B(k), B(k'))$ und

wir schreiben dann $k < k'$. Hierdurch werden die persistenten Knoten total geordnet. Es gilt im übrigen $k < k' \iff B(k) < B(k')$.

Sind k, k' persistent und gilt $k = p_j(k')$ für ein $j \geq \max(B(k), B(k'))$, so folgt $k = p_j(k')$ für alle $j \geq \max(B(k), B(k'))$ und wir schreiben dann $k = p(k')$. Hierdurch bilden die persistenten Knoten einen (endlichen!) Baum. Die Tiefe $H(k)$ eines persistenten Knotens ist die Tiefe von k in diesem Baum, also sein Abstand von der Wurzel.

Ein persistenter Knoten k *trifft den Lauf*, wenn $j \geq B(k)$ existiert mit $q_j \in l_j(k)$. Wir bemerken, dass die Wurzel 1 den Lauf trifft.

Unter allen persistenten Knoten, die den Lauf treffen, suchen wir nun einen mit maximaler Tiefe und unter allen, die diese maximale Tiefe erreichen, bestimmen wir den ältesten und bezeichnen ihn mit k_0. Wir behaupten nun: k_0 blitzt unendlich oft, also $!_i(k_0) = \mathtt{tt}$ für unendlich viele i.

Sei $q_i \in l_i(k_0)$. Nachdem k_0 der älteste Knoten auf diesem Niveau ist, gilt auch $q_j \in l_j(k_0)$ für alle $j \geq i$. (Man muss sich hier kurz überlegen, dass es nicht möglich ist, dass k_0 von einem nicht-persistenten Knoten etwas weggenommen wird.)

Zu jedem $j_1 \geq i$ findet man $j_2 \geq j_1$ mit $q_{j_2} = f \in l_{j_2}(k_0)$, da ja f immer wieder kommt. Zu diesem Zeitpunkt wird ein Kind von k_0 erzeugt, wessen Beschriftung f enthält. Würde k_0 ab diesem Zeitpunkt nicht mehr blitzen, so bliebe dieses oder ein älteres Kind am Leben und träfe den Lauf im Widerspruch zur Extremalität von k_0. ∎

Zusammengefasst haben wir nun bewiesen:

Satz 21 (Safra)
Zu jedem Büchiautomaten $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ mit $|Q| = n$ existiert ein deterministischer Mullerautomat $\mathcal{M}$ mit $2^{O(n \log n)}$ Zuständen mit $L(\mathcal{A}) = L(\mathcal{M})$.

Beweis Die Konstruktion von $\mathcal{M}$ und ihre Korrektheit wurde bereits gezeigt. Die Größenbeschränkung ergibt sich leicht aus folgender Überlegung. Sei $t = (K, p, l, !) \in S(Q)$. Für jedes $q \in Q$ existiert ein eindeutiger Knoten $k \in K$, so dass $q \in l(k)$ aber $q \notin l(k')$ für alle k' mit $k' \neq k$ und $p(k') = k$. D.h. ist ein Zustand in einem Knoten vorhanden, dann auch in dem Elternknoten. Daher gilt $|K| \leq n$. Da in jedem Schritt höchstens so viele neue Kinder erzeugt werden können, wie es gerade Knoten gibt, reicht es eben aus, sich auf die Knotennamen $\{1, \ldots, 2n\}$ zu beschränken.

Beachte jetzt, dass ein Safrabaum eindeutig bestimmt ist durch

- eine Funktion $f : Q \to \{\bot, 1, \ldots, 2n\}$, die jedem Zustand den eindeutigen "tiefsten" Knoten zuordent, in dem er vorkommt,
- die Elternfunktion $p : \{1, \ldots, 2n\} \to \{\bot, 1, \ldots, 2n\}$,
- die Bruderfunktion $s : \{1, \ldots, 2n\} \to \{\bot, 1, \ldots, 2n\}$,
- die Menge $K \subseteq \{1, \ldots, 2n\}$ mit $|K| \leq n$. ∎

Die Anzahl der möglichen Safrabäume über Q ist somit beschränkt durch

$$(2n+1)^n \cdot ((2n+1)^{2n})^2 \cdot \Big(\sum_{i=0}^{n} \prod_{j=1}^{i} 2n - j + 1\Big) \;=\; 2^{O(n \log n)}$$

Dieser Satz liefert einen alternativen Zugang zum Satz von Büchi (Entscheidbarkeit der MSO), da man einen deterministischen Mullerautomaten sehr leicht komplementieren kann: man ersetzt einfach die Akzeptanzbedingung $\mathcal{F}$ durch ihr Komplement. Man kann somit einer MSO-Formel ϕ induktiv einen deterministischen Mullerautomaten zuordnen. Beim Behandeln des Existenzquantors muss man dann aber den Umweg über einen nichtdeterministischen Büchiautomaten gehen, der dann wieder mithilfe der Safrakonstruktion zu determinisieren ist.

Für einen Rabinautomaten $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ mit $\mathcal{F} = \{(G_1, F_1), \ldots, (G_k, F_k)\}$ sei k der *Index* von $\mathcal{A}$.

Korollar 22
Zu jedem Büchiautomaten $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ mit $|Q| = n$ existiert ein deterministischer Rabinautomat $\mathcal{M}$ mit $2^{O(n \log n)}$ Zuständen und Index höchstens $2n$, so dass $L(\mathcal{A}) = L(\mathcal{M})$ gilt.

Beweis Der deterministische Mullerautomat $\mathcal{M}$ aus der Safrakonstruktion läßt sich leicht als Rabinautomat auffassen. Dazu sei lediglich

$$\begin{aligned} G_i &:= \{t \in S(Q) \mid \text{ der Knoten } i \text{ blitzt in } t\ \} \\ F_i &:= \{t \in S(Q) \mid\ t \text{ enthält den Knoten } i \text{ nicht } \} \end{aligned}$$

für $i = 1, \ldots, 2n$. ∎

Satz 23 (Safra)
Es gibt Büchi-Automaten $\mathcal{A}_n$ mit $O(n)$ vielen Zuständen, für die es keine deterministischen Rabinautomaten mit nur $2^{O(n)}$ vielen Zuständen und Index $O(n)$ gibt.

3.5 Weitere Akzeptanzbedingungen

Neben den bereits eingeführten Büchi-, Rabin- und Mullerautomaten betrachten wir noch zwei weitere Arten, Akzeptanz in einem unendlichen Lauf zu definieren.

- Beachte, dass die Safrakonstruktion eingeführt wurde, um nichtdeterministische Büchiautomaten zu komplementieren. Man erhält so zwar einen deterministischen Automaten. Der ist aber entweder ein Muller-Automat, der sich zwar leicht komplementieren lässt, jedoch eine exponentiell große Endzustandskomponente $\mathcal{F}$ haben kann. Oder man fasst diesen als Rabinautomaten mit linear großer Endzustandskomponente $\{(G_1, F_1), \ldots, (G_n, F_n)\}$ auf, was sich nicht mehr leicht komplementieren lässt. Beachte, dass dazu die Rabin-Begingung $\exists i \ldots$ zu $\forall i \ldots$ komplementiert wird, was sich nicht einfach wieder als Rabin-Bedingung ausdrücken lässt. Aus diesem Grund definieren wir noch Streett-Automaten, welche eine Akzeptanzbedingung dual zu der Rabin-Bedingung haben.

  Darüberhinaus bieten sich Streett-Automaten auch an, um starke Fairnesseigenschaften zu modellieren.

- *Paritätsautomaten* ordnen jedem Zustand eine Priorität zu. Das Akzeptanzkriterium betrachtet dann lediglich diejenigen Prioritäten, die in einem Lauf unendlich oft vorkommen. Ihre deterministische Variante ist wiederum einfach zu komplementieren.

Definition 17
Ein *Streett-Automat* ist ein Tupel $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$, wobei Q, Σ, q_0, δ wie bei einen nicht-deterministischen Büchiautomaten definiert sind, und $\mathcal{F} = \{(G_1, F_1), \ldots, (G_k, F_k)\}$ mit $G_i, F_i \subseteq Q$.

Ein Lauf $\pi = q_0, q_1, \ldots$ eines Streett-Automaten heißt akzeptierend, falls für alle $i = 1, \ldots, k$ gilt: $Inf(\pi) \cap G_i \neq \emptyset \Rightarrow Inf(\pi) \cap F_i \neq \emptyset$.

Der *Index* eines Streett-Automaten ist die Größe seiner Endzustandskomponente, also hier k.

Die Endzustandskomponente beschreibt also Abhängigkeiten der Form, dass ein bestimmter Zustand nur unendlich oft durchlaufen werden darf, wenn auch ein anderer unendlich oft durchlaufen wird.

Der Index misst in vernünftiger Weise die Größe einer Repräsentation eines Automaten. Beachte, dass sich ein Rabinautomat genauso durch explizite Auflistung der Mengen U_i, V_i repräsentieren läßt. Bei einem Mullerautomaten ist diese Definition weniger sinnvoll.

Definition 18
Ein *Paritätsautomat* ist ein Tupel $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$, wobei Q, Σ, q_0, δ wie üblich definiert sind, und $\Omega : Q \to \mathbb{N}$.

Ein Lauf $\pi = q_0, q_1, \ldots$ eines Paritätsautomaten heißt akzeptierend, falls $\max\{\Omega(q) \mid q \in Inf(\pi)\}$ gerade ist.

Der *Index* eines Paritätsautomaten ist die maximale Priorität, die einem seiner Zustände zugeteilt ist.

Die Begriffe *Rabin-*, *Streett-erkennbar*, etc. für ω-Sprachen werden in der üblichen Weise erklärt.

Satz 24
Für eine Sprache $L \subseteq \Sigma^\omega$ sind die folgenden Aussagen äquivalent.

a) L ist (nicht-deterministisch) Büchi-erkennbar,

b) L ist Rabin-erkennbar,

c) L ist Streett-erkennbar,

d) L ist paritätserkennbar,

e) L ist Muller-erkennbar.

Beweis Die Teile $(a) \Rightarrow (b)$, $(a) \Rightarrow (c)$, $(a) \Rightarrow (d)$ und $(a) \Rightarrow (e)$ sind einfach, da sich die Büchi-Bedingung z.B. durch eine Rabin- oder Streett-Bedingung vom Index 1 oder auch

eine Paritätsbedingung mit Prioritäten 1 und 2 modellieren lässt. Ausserdem wurden diese Richtungen teilweise bereits in stärkerer Form im letzten Absatz gezeigt.

Die Richtungen $(b) \Rightarrow (e)$, $(c) \Rightarrow (e)$ und $(d) \Rightarrow (e)$ sind ebenfalls einfach, da sich die entsprechenden Akzeptanzkriterien einfach durch Auflisten aller Teilmengen von Zuständen, die die jeweiligen Bedingungen erfüllen, als Mullerbedingung modellieren lassen.

Schließlich wurde die Richtung $(e) \Rightarrow (a)$ bereits in Proposition 18 skizziert. ∎

Beispiel 6
Sei $\Sigma = \{a, b, c\}$. Betrachte die Sprache $L := \{w \in \Sigma^\omega \mid |w|_a = \infty \Rightarrow |w|_b = \infty\}$. Betrachte außerdem den Automaten $\mathcal{A}$ mit Komponenten $Q = \{q_a, q_b, q_c\}$, $q_0 = q_a$ und $\delta(q_x, y) = q_y$ für alle $x, y \in \Sigma$.

- Mit $\mathcal{F} = \{(\{q_a\}, \{q_b\})\}$ wird $\mathcal{A}$ zu einem Streett-Automaten, der L mit Index 1 erkennt.
- Mit $\mathcal{F} = \{(\{q_b\}, \emptyset), (\{q_c\}, \{q_a, q_b\})\}$ wird $\mathcal{A}$ zu einem Rabin-Automaten, der L mit Index 2 erkennt.
- Mit $\mathcal{F} = \{F \subseteq Q \mid q_a \in F \Rightarrow q_b \in F\}$ wird $\mathcal{A}$ zu einem Muller-Automaten, der L erkennt.
- Mit $\Omega(q_a) = 1$, $\Omega(q_b) = 2$, $\Omega(q_c) = 0$ wird $\mathcal{A}$ zu einem Paritätsautomaten, der L mit Index 2 erkennt.

3.6 Entscheidungsverfahren für ω-Automaten

Wir betrachten im wesentlichen das Leerheitsproblem (ist $L(\mathcal{A}) = \emptyset$?) und das Universalitätsproblem (ist $L(\mathcal{A}) = \Sigma^\omega$?). Andere Probleme wie das Wortproblem für endlich repräsentierte ω-Wörter etc. lassen sich normalerweise leicht auf diese reduzieren.

Satz 25
Das Leerheitsproblem für einen nicht-deterministischen Rabin-Automaten mit n Zuständen und Index k ist in Zeit $O(n^2 k)$ lösbar.

Beweis Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, \{(G_1, F_1), \ldots, (G_k, F_k)\})$ ein NRA. Dann gilt: $L(\mathcal{A}) \neq \emptyset$ gdw. es ein $i \in \{1, \ldots, k\}$ gibt, so dass es einen Pfad von q_0 zu einem $q \in G_i$ gibt und einen nicht-leeren Pfad von q nach q, auf dem kein Zustand in F_i vorkommt.

Für ein festes i ist es möglich, mit z.B. zwei verschachtelten Breitensuchen in Zeit $O(n^2)$ dies zu entscheiden. ∎

Korollar 26
Das Leerheitsproblem für nicht-deterministische Büchi-Automaten mit n Zuständen ist in Zeit $O(n^2)$ lösbar.

Beweis Ein nicht-deterministischer Büchiautomat mit Endzustandsmenge F ist ein Spezialfall eines nicht-deterministischen Rabinautomaten mit Endzustandskomponente $\mathcal{F} = \{(F, \emptyset)\}$ und somit vom Index 1. ∎

Korollar 27
Das Leerheitsproblem für nicht-deterministische Paritätsautomaten mit n Zuständen und Index k ist in Zeit $O(n^2k)$ lösbar.

BEWEIS Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ ein nicht-deterministischer Paritätsautomat vom Index k. Sei $k' := \lfloor \frac{k}{2} \rfloor$. Es ist leicht zu sehen, dass der Rabinautomat $\mathcal{A}' = (Q, \Sigma, q_0, \delta, \{(G_0, F_0), \ldots, (G_{k'}, F_{k'})\})$ mit

$$\begin{aligned} G_i &:= \{q \in Q \mid \Omega(q) = 2i\} \\ F_i &:= \{q \in Q \mid \Omega(q) > 2i\} \end{aligned}$$

dieselbe Sprache erkennt wie $\mathcal{A}$. Außerdem ist sein Index $O(k)$. ∎

Korollar 28
Das Universalitätsproblem für deterministische Streett-Automaten mit n Zuständen und Index k ist in Zeit $O(n^2k)$ lösbar.

BEWEIS Dies folgt aus der Tatsache, dass sich ein deterministischer Rabin-Automat leicht zu einem deterministischen Streett-Automaten komplementieren lässt. ∎

Satz 29
Das Leerheitsproblem für nicht-deterministische Streett-Automaten mit n Zuständen und Index k ist in Zeit $O(n^2k^2)$ lösbar.

BEWEIS Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, \{(G_1, F_1), \ldots, (G_k, F_k)\})$ ein Streett-Automat. Jetzt gilt: $L(\mathcal{A}) \neq \emptyset$ gdw. es einen Lauf $\pi = q_0 q_1 \ldots$ gibt, auf dem die Streett-Bedingung erfüllt ist. Zuerst behaupten wir, dass es ausreicht, sich auf ultimativ periodische Läufe der Form uv^ω mit $u \in Q^*$, $v \in Q^+$ zu beschränken. Dass dies hinreichend ist, ist offensichtlich.

Für die Notwendigkeit betrachte einen nicht-ultimativ-periodischen Lauf $\pi = q_0 q_1 \ldots$. Dieser enthält aufgrund des Schubfachprinzips ein kleinstes Teilstück der Form $p_1 \ldots p_m p_1$, so dass für alle $i = 1, \ldots, k$ gilt: $G_i \cap \{p_1, \ldots, p_m\} = \emptyset$ oder $F_i \cap \{p_1, \ldots, p_m\} \neq \emptyset$. Dann erfüllt auch der Lauf $(p_1 \ldots p_m)^\omega$ die Streett-Bedingung. Schliesslich sieht man leicht, dass das Hinzufügen endlicher Präfixe nichts daran ändert.

Aufgrund dieser Überlegung gilt $L(\mathcal{A}) \neq \emptyset$ gdw. es eine starke Zusammenhangskomponente $C \subseteq Q$ gibt, die von q_0 aus erreichbar ist, so dass C als Streett-Automat eine nicht-leere Sprache akzeptiert.

Wir beschreiben zunächst einen Algorithmus, der von einer starken Zusammenhangskomponente C entscheidet, ob die von ihr erkannte Sprache nicht-leer ist. Offensichtlich hängt diese Frage nicht davon ab, welchen Zustand in dieser Komponente man als Anfangszustand wählt. Sei $C = \{p_1, \ldots, p_m\}$.

- Falls $C \cap F_i \neq \emptyset$ für alle $i = 1, \ldots, k$ gilt, dann ist $L(C) \neq \emptyset$. Beachte, dass in C jeder Zustand von jedem aus erreichbar ist. Daher lässt sich immer ein Lauf konstruieren, auf dem ein Zustand aus jedem F_i unendlich oft vorkommt.

- Falls nicht, dann gibt es ein $i \in \{1, \ldots, k\}$, so dass $C \cap F_i = \emptyset$. Damit C ein Wort akzeptieren kann, muss es einen Lauf geben, der keinen Zustand aus G_i unendlich oft durchläuft und außerdem noch die Streett-Bedingung vom Index $k-1$ auf den restlichen Paaren erfüllt. Beachte auch hier wieder, dass sich "unendlich oft" in der starken Zusammenhangskomponente auf "einmal" reduzieren läßt.

Somit funktioniert der Leerheitstest folgendermaßen.

Algorithmus *Streett-Nicht-Leerheit*
Eingabe: Streett-Automat $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$
Ausgabe: $L(\mathcal{A}) \neq \emptyset$?

`let` $\{(G_1, F_1), \ldots, (G_k, F_k)\} = \mathcal{F}$
`if` $k = 0$ `then return True`
`else` zerlege $\mathcal{A}$ in von q_0 erreichbare SCCs $C_1, \ldots, C_l$
  `for` $j = 1, \ldots, l$ `do`
    `if` $C_j \cap F_i \neq \emptyset$ `for all` $i = 1, \ldots, k$ `then return True`
    `for all` $i = 1, \ldots, k$ `with` $C_j \cap F_i = \emptyset$ `do`
      lösche alle G_i-Zustände in C_j
      $\mathcal{F} := \mathcal{F} \setminus \{(G_i, F_i)\}$
    `done`
    `let` $q =$ irgendein Zustand aus C_j
    `if` *Streett-Nicht-Leerheit*$(C_j, \Sigma, q, \delta, \mathcal{F})$ `then return True`
  `done`
  `return False`

Die Korrektheit zeigt man leicht durch Induktion über den Index k von $\mathcal{A}$. Für $k = 0$ beachte, dass eine leere Streett-Bedingung immer erfüllbar ist. Sei nun $k > 0$. Falls es ein C_j gibt, welches einen nicht-leeren Schnitt mit allen F_i bildet, dann lässt sich mit obiger Überlegung leicht ein akzeptierender Lauf konstruieren. Beachte, dass C_j vom Anfangszustand aus erreichbar ist.

Falls der Algorithmus `True` aufgrund des rekursiven Aufrufs liefert, dann gibt es ein $q \in C_j$ mit einem partiellen Lauf $q \ldots q$, auf dem die um einige (G_i, F_i) reduzierte Streett-Bedingung gilt. Da in diesem partiellen Lauf kein Zustand aus einem G_i vorkommen kann, lässt sich dieser wieder zu einem unendlichen Lauf erweitern, der ebenfalls die ursprüngliche Streett-Bedingung erfüllt.

Vollständigkeit ist einfacher einzusehen: Der Algorithmus liefert nur dann `False`, wenn es keine starke Zusammenhangskomponente gibt, in der sich die Streett-Bedingung auf einem ultimativ-periodischen Lauf realisieren läßt. Nach obiger Überlegung gibt es dann auch keinen akzeptierenden Lauf. Außerdem sieht man leicht, dass der Algorithmus auf jeder Eingabe terminiert.

Zur Analyse der Laufzeit beachte, dass sich starke Zusammenhangskomponenten z.B. mit Tarjan's Algorithmus in Zeit $O(n)$ finden lassen. Sei $T(n, k)$ die worst-case Laufzeit

des Algorithmus. Offensichtlich gilt $T(n,0) = O(1)$. Desweiteren gilt für $k > 0$:

$$T(n,k) \;=\; O(n) + \sum_{j=1}^{l} O(kn \cdot |C_j|) + T(|C_j|, k-1) \;=\; O(n) + O(kn^2) + T(n, k-1)$$

Man sieht leicht, dass $T(n,k) = O(n^2k^2)$ eine Lösung dieser Rekurrenz ist. ∎

Korollar 30
Das Universalitätsproblem für nicht-deterministische Büchiautomaten mit n Zuständen ist in Zeit $2^{O(n \log n)}$ lösbar.

Beweis Sei $\mathcal{A}$ ein nicht-deterministische Büchiautomat mit n Zuständen. Laut Korollar 22 existiert ein deterministischer Rabin-Automat $\mathcal{A}'$ mit höchstens $2^{O(n \log n)}$ vielen Zuständen und Index höchstens n, so dass $L(\mathcal{A}') = L(\mathcal{A})$ gilt. Sei $\mathcal{A}' = (Q, \Sigma, q_0, \delta, \mathcal{F})$. Aufgrund des Determinismus läßt sich $\mathcal{A}'$ leicht zu dem Streett-Automaten $\overline{A} := (Q, \Sigma, q_0, \delta, \mathcal{F})$ komplementieren, also $L(\overline{A}) = \Sigma^\omega \setminus L(\mathcal{A})$.

Die Behauptung folgt dann sofort aus Proposition 29, da $L(\mathcal{A}) = \Sigma^\omega$ gdw. $L(\overline{A}) = \emptyset$ gilt. ∎

3.7 Temporale Logiken auf unendlichen Wörtern

3.7.1 LTL

Definition 19
Sei Σ ein Alphabet. Formeln der *Linear Time Temporal Logic* (LTL) über Σ sind gegeben durch die folgende Grammatik.

$$\varphi \;::=\; a \mid \varphi \vee \varphi \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi \mathtt{U} \varphi$$

wobei $a \in \Sigma$. Wir benutzen neben den üblichen Abkürzungen der Aussagenlogik auch die folgenden: $\varphi \mathtt{R} \psi := \neg(\neg\varphi \mathtt{U} \neg\psi)$, $\mathtt{F}\varphi := \mathtt{tt}\mathtt{U}\varphi$ und $\mathtt{G}\varphi := \mathtt{ff}\mathtt{R}\varphi$.

Sei $w = a_0 a_1 \ldots \in \Sigma^\omega$. Die Semantik einer LTL-Formel ist induktiv definiert für alle $i \in \mathbb{N}$ wie folgt.

$$\begin{aligned}
w, i \models a \quad &\text{gdw.} \quad a_i = a \\
w, i \models \varphi \vee \psi \quad &\text{gdw.} \quad w, i \models \varphi \text{ oder } w, i \models \psi \\
w, i \models \neg\varphi \quad &\text{gdw.} \quad w, i \not\models \varphi \\
w, i \models \bigcirc\varphi \quad &\text{gdw.} \quad w, i+1 \models \varphi \\
w, i \models \varphi \mathtt{U} \psi \quad &\text{gdw.} \quad \text{es gibt } k \geq i, \text{ so dass } w, k \models \psi \\
&\qquad\qquad \text{und für alle } j : i \leq j < k \Rightarrow w, j \models \varphi
\end{aligned}$$

Wir schreiben $w \models \varphi$ gdw. $w, 0 \models \varphi$, und $L(\varphi) := \{w \mid w \models \varphi\}$.

Satz 31
LTL $\leq$ FO.

Beachte, dass die Semantik einer LTL-Formel bereits in FO aufgeschrieben ist. Genauer: Die Semantik einer LTL-Formel φ ist eine FO-Formel $\phi(x)$, so dass für alle $i \in \mathbb{N}$ gilt: $w, i \models \varphi$ gdw. $w, i \models \phi(x)$. Somit gilt $w \models \varphi$ gdw. $w \models \phi(min)$.

Eine LTL-Formel ist in *positiver Normalform*, wenn sie nur aus Buchstaben $a \in \Sigma$ mit den Operatoren $\vee, \wedge, \bigcirc, \mathtt{U}$ und $\mathtt{R}$ aufgebaut ist.

Lemma 13
Jede LTL-Formel φ ist äquivalent zu einer LTL-Formel φ' in positiver Normalform, so dass $|\varphi'| = O(|\varphi|)$ gilt.

Beweis Übung. ■

Lemma 14
Für alle $\varphi, \psi \in LTL$ gilt: $\varphi \mathtt{U} \psi \equiv \psi \vee (\varphi \wedge \bigcirc(\varphi \mathtt{U} \psi))$.

Beweis Übung. ■

Definition 20
Ein *verallgemeinerter Büchi-Automat* ist ein Tupel $\mathcal{A} = (Q, \Sigma, I, \delta, F_1, \ldots, F_k)$ mit $I \subseteq Q$ und $F_1, \ldots, F_k \subseteq Q$. Ein Lauf eines solchen Automaten auf einem Wort $w \in \Sigma^\omega$ ist definiert wie bei einem Büchi-Automaten, mit dem Unterschied, dass er in einem beliebigen Zustand $q \in I$ anfängt.

Solch ein Lauf $q_0, q_1, \ldots$ heißt *akzeptierend*, falls für alle $i = 1, \ldots, k$ ein $q \in F_i$ gibt, so dass $q = q_j$ für unendlich viele j.

Ein verallgemeinerter Büchi-Automat hat also erstens eine Anfangszustandsmenge statt einem einzigen Anfangszustand und zweitens mehrere Endzustandsmengen, die jeweils unendlich oft besucht werden müssen.

Lemma 15
Für jeden verallgemeinerten Büchi-Automaten $\mathcal{A} = (Q, \Sigma, I, \delta, F_1, \ldots, F_k)$ gibt es einen Büchi-Automaten $\mathcal{A}'$, so dass $L(\mathcal{A}') = L(\mathcal{A})$ und $|\mathcal{A}'| = 1 + |Q| \cdot (k + 1)$.

Beweis Übung. ■

Definition 21
Der *Fischer-Ladner-Abschluss* einer LTL-Formel φ_0 in positiver Normalform ist die kleinste Menge $FL(\varphi_0)$, für die gilt:

- $\varphi \vee \psi \in FL(\varphi_0) \Rightarrow \{\varphi, \psi\} \subseteq FL(\varphi_0)$,
- $\varphi \wedge \psi \in FL(\varphi_0) \Rightarrow \{\varphi, \psi\} \subseteq FL(\varphi_0)$,
- $\bigcirc\varphi \in FL(\varphi_0) \Rightarrow \varphi \in FL(\varphi_0)$,
- $\varphi \mathtt{U} \psi \in FL(\varphi_0) \Rightarrow \{\varphi, \psi, \bigcirc(\varphi \mathtt{U} \psi), \varphi \wedge \bigcirc(\varphi \mathtt{U} \psi), \psi \vee (\varphi \wedge \bigcirc(\varphi \mathtt{U} \psi))\} \subseteq FL(\varphi_0)$,
- $\varphi \mathtt{R} \psi \in FL(\varphi_0) \Rightarrow \{\varphi, \psi, \bigcirc(\varphi \mathtt{R} \psi), \varphi \vee \bigcirc(\varphi \mathtt{R} \psi), \psi \wedge (\varphi \vee \bigcirc(\varphi \mathtt{R} \psi))\} \subseteq FL(\varphi_0)$,

Eine *Hintikka*-Menge für eine LTL-Formel φ_0 in positiver Normalform ist eine Menge $M \subseteq FL(\varphi_0)$, für die gilt:

- $\varphi \vee \psi \in M \Rightarrow \varphi \in M$ oder $\psi \in M$,
- $\varphi \wedge \psi \in M \Rightarrow \varphi \in M$ und $\psi \in M$,
- $\varphi \mathtt{U} \psi \in M \Rightarrow \psi \in M$ oder $\big(\ \varphi \in M$ und $\bigcirc(\varphi \mathtt{U} \psi) \in M\ \big)$,
- $\varphi \mathtt{R} \psi \in M \Rightarrow \psi \in M$ und $\big(\ \varphi \in M$ oder $\bigcirc(\varphi \mathtt{R} \psi) \in M\ \big)$.

Eine Hintikka-Menge M heißt *konsistent*, falls es keine $a, b \in \Sigma$ gibt mit $a \neq b$ und $\{a, b\} \subseteq M$. Beachte, dass die konsistenten Hintikka-Mengen genau die maximal konsistenten Teilmengen des Fischer-Ladner-Abschlusses sind. Sei $\mathcal{H}(\varphi_0)$ die Menge aller konsistenten Hintikka-Mengen über φ_0.

Satz 32
Für jede LTL-Formel φ gibt es einen Büchi-Automaten $\mathcal{A}_\varphi$, so dass $L(\mathcal{A}_\varphi) = L(\varphi)$ und $|\mathcal{A}_\varphi| = O(|\varphi| \cdot 2^{2|\varphi|})$ gilt.

Beweis Sei $\varphi \in$ LTL. Wegen Lemma 13 nehmen wir an, dass φ in positiver Normalform vorliegt. Seien $\psi_1 \mathtt{U} \chi_1, \ldots, \psi_k \mathtt{U} \chi_k$ alle in φ vorkommenden $\mathtt{U}$-Formeln. Wir definieren einen verallgemeinerten Büchi-Automaten $\mathcal{A}_\varphi := (\mathcal{H}(\varphi), \Sigma, I, \delta, F_1, \ldots, F_k)$ mit

- $I := \{M \mid \varphi \in M\}$,
- $F_i := \{M \mid \psi_i \mathtt{U} \chi_i \in M \Rightarrow \chi_i \in M\}$ für alle $i = 1, \ldots, k$,
- die Übergangsrelation ist wie folgt definiert

$$\delta(M, a) \ := \ \begin{cases} \emptyset, & \text{falls } \exists b \in \Sigma \cap M \text{ mit } b \neq a \\ \{M' \mid \text{ für alle } \bigcirc \psi \in M \text{ gilt: } \psi \in M'\}, & \text{sonst} \end{cases}$$

Intuitiv rät $\mathcal{A}_\varphi$ die Menge aller Unterformeln von φ, die von dem noch zu lesenden Suffix erfüllt werden. Diese Menge muss natürlich konsistent sein. Außerdem soll sie maximal sein, damit $\mathcal{A}_\varphi$ nicht evtl. ein Wort akzeptiert, welches φ nicht erfüllt. Also muss jede solche Menge eine konsistente Hintikka-Menge bilden.

Beachte: $FL(\varphi) \leq 2|\varphi|$. Somit gilt $|\mathcal{H}(\varphi)| \leq 2^{2|\varphi|}$. Laut Lemma 15 gibt es dann einen zu $\mathcal{A}_\varphi$ äquivalenten Büchi-Automaten, der höchstens $O(|\varphi| \cdot 2^{2|\varphi|})$ viele Zustände hat.

Es bleibt noch zu zeigen, dass für alle $w \in \Sigma^\omega$ gilt: $w \in L(\mathcal{A}_\varphi)$ gdw. $w \in L(\varphi)$ gilt.

"$\Leftarrow$" Definiere für jedes $i \in \mathbb{N}$ eine Menge $M_i := \{\psi \in FL(\varphi) \mid w, i \models \psi\}$. Folgendes ist leicht zu sehen:

1. Jedes M_i ist eine konsistente Hintikka-Menge.
2. $\varphi \in M_0$.
3. Wenn $\bigcirc\psi \in M_i$, dann $\psi \in M_{i+1}$.

4. Falls der i-te Buchstabe von w ein a ist, dann ist $M_i \cap \Sigma \subseteq \{a\}$.

5. Für jedes $j \in \{1, \ldots, k\}$ und jedes $i \in \mathbb{N}$ gilt: wenn $\psi_j \mathtt{U} \chi_j \in M_i$ dann existiert ein $i' \geq i$, sodass $\{\psi_j \mathtt{U} \chi_j, \chi_j\} \subseteq M_{i'}$.

Wegen (1) bildet $M_0, M_1, \ldots$ eine Sequenz von Zuständen von $\mathcal{A}_\varphi$. Wegen (2) beginnt diese in einem Anfangszustand. Wegen (3) ist diese Sequenz ein gültiger Lauf, der die Transitionsrelation befolgt. Wegen (4) bleibt dieser Lauf niemals stehen. Und wegen (5) ist er akzeptierend.

"$\Rightarrow$" Sei $w \in L(\mathcal{A}_\varphi)$. D.h. es existiert ein akzeptierender Lauf $M_0, M_1, \ldots$ von $\mathcal{A}$ auf w. Man zeigt nun leicht durch Induktion über den Formelaufbau, dass für alle $i \in \mathbb{N}$ und alle $\psi \in FL(\varphi)$ folgendes gilt. Wenn $\psi \in M_i$ dann $w, i \models \psi$.

Der Induktionsanfang für $\psi = a$ folgt aus der Tatsache, dass jedes M_i konsistent ist. Also gilt entweder $a \notin M_i$, oder, falls $a \in M_i$ und $w, i \not\models a$, es gäbe den unendlichen Lauf nicht.

Für die booleschen Operatoren folgt die Aussage aus der Hypothese und der Tatsache, dass jedes M_i eine Hintikka-Menge bildet. Für $\bigcirc$-Formeln folgt sie aus der Definition der Transitionsrelation.

Sei nun $\psi_j \mathtt{U} \chi_j \in M_i$ für ein $i \in \mathbb{N}$. Nach Voraussetzung existiert solch ein $j \in \{1, \ldots, k\}$.

Da M_i eine konsistente Hintikka-Menge ist, gilt entweder $\chi_j \in M_i$, woraus die Behauptung sofort der Induktion folgt. Oder aber es gilt $\psi_j \in M_j$ und $\bigcirc(\psi_j \mathtt{U} \chi_j) \in M_i$. Nach der Definition der Transitionsrelation ist dann aber $\psi_j \mathtt{U} \chi_j \in M_{i+1}$. Dieses Argument lässt sich nun iterieren, was $w, i' \models \psi_j$ für $i' = i, i+1, i+2, \ldots$ zeigt. Beachte außerdem, dass $\psi_j \mathtt{U} \chi_j \in M_{i'}$ für $i' = i, i+1, i+2, \ldots$ gilt.

Da der zugrundeliegende Lauf aber akzeptierend ist, muss es ein $i' > i$ geben, so dass $\{\psi_j \mathtt{U} \chi_j, \chi_j\} \subseteq M_{i'}$. Die Induktionshypothese, angewandt auf χ_j liefert dann $w, i' \models \chi_j$ und $w, h \models \psi_j$ für alle $i \leq h < i'$. Somit gilt ebenfalls $w, i \models \psi_j \mathtt{U} \chi_j$.

Der Fall einer `R`-Formel ist ähnlich zum `U`-Fall. ∎

3.8 Size-change Termination

In diesem Abschnitt betrachten wir eine Anwendung (The Size-Change Principle for Program Termination, Lee-Jones-Ben Amram, ACM POPL 2001) von Büchiautomaten auf die Frage, ob ein gegebenenes rekursives Programm terminiert. Im Rahmen dieser Anwendung wurde von Jones et al ein neues Entscheidungsverfahren für die Inklusion von Büchiautomaten entwickelt, welches wir (ML, MH, Christian Dax) kürzlich in erweiterter Form zur Entscheidung der Allgemeingültigkeit von Formeln des μTL – einer Erweiterung von LTL – eingesetzt haben.

3.8.1 Rekursive Programme

Gegeben sei eine Menge F von Funktionssymbolen verschiedener Stelligkeit. Ein (rekursives) *Programm* enthält für jedes Funktionssymbol $f \in F$ der Stelligkeit n genau eine

Gleichung der Form

$$f(x_1, \ldots, x_n) = f_1(\vec{t_1}), \ldots, f_l(\vec{t_l})$$

Hierbei sind $f_1, \ldots f_l$ beliebige Funktionssymbole in F (einschließlich f selbst). Die Terme $\vec{t_i}$ sind aufgebaut aus den Variablen $x_1, \ldots, x_n$ und der Vorgängerfunktion $x - 1$. Hier sind ein paar konkrete Beispiele:

Multiplikation: Ein zwei- und ein dreistelliges Symbol, p und m:

$$p(x, y) = p(x, y - 1)$$
$$m(x, y, u) = p(x, u), m(x, y - 1, u)$$

Ackermann: Ein dreistelliges Symbol a:

$$a(x, y, u) = a(x - 1, u, u), a(x, y - 1, u)$$

Künstlich I:

$$f(x, y, z) = g(x, x, y)$$
$$g(x, y, z) = h(x, y - 1, z - 1)$$
$$h(x, y, z) = i(x, y - 1, z - 1)$$
$$i(x, y, z) = k(x, y - 1, z - 1)$$
$$k(x, y, z) = f(x, y - 1, z - 1)$$

Künstlich II:

$$t(x, y, z, w) = t(x, x, z, w - 1), t(x - 1, z, w - 1, y - 1), t(z, x - 1, y, w - 1)$$

Die Idee ist, dass durch solch ein Programm partielle Funktionen $\mathbb{N}^k \to \{0\}$ definiert werden. Per Definition ist $f(x_1, \ldots, x_n) = 0$, falls $x_i = 0$ für ein i (Striktheit). Für andere Werte ergibt sich der Funktionswert aus der definierenden Gleichung, wobei die Semantik des Kommas so ist, dass $u_1, \ldots, u_k$ gleich 0 und damit insbesondere definiert ist, wenn alle Terme $u_1, u_2, \ldots, u_k$ definiert sind. Ist auch nur einer der Terme $u_1 \ldots u_k$ undefiniert, so ist der Komma-Ausdruck bereits undefiniert.

Ist ein Programm gegeben, so fragt man sich, welche, insbesondere ob alle Instanzen $f(u_1, \ldots, u_n)$ definiert sind. Bei den Beispielsfunktionen p, m, a sind alle Instanzen definiert; bei f haben wir $f(10, 10, 10) = g(10, 9, 9) = h(10, 8, 8) = i(10, 7, 7) = k(10, 6, 6) = f(10, 5, 5) = g(10, 10, 5) = h(10, 9, 4) = i(10, 8, 3) = k(10, 7, 2) = f(10, 6, 1) = g(10, 10, 6) = \ldots$.

Man sieht, dass $f(10, 10, 10)$ und größere Instanzen nicht definiert sind; $f(9, 9, 9)$ und kleinere Instanzen dagegen schon.

Schreiben wir $f(\vec{t}) \rightsquigarrow g(\vec{u})$ zum Zeichen, dass der Aufruf $f(\vec{t})$ den Aufruf $g(\vec{u})$ unmittelbar nach sich zieht.

Es gilt: $t(3, 10, 11, 4) \rightsquigarrow t(2, 11, 3, 9) \rightsquigarrow t(3, 1, 11, 8) \rightsquigarrow t(3, 3, 11, 7) \rightsquigarrow t(11, 2, 3, 6) \rightsquigarrow t(11, 11, 3, 5) \rightsquigarrow t(3, 10, 11, 4)$ und damit ist $t(3, 10, 11, 4)$ nicht definiert. Hingegen ist

$t(10, 10, 10, 10)$ definiert, wie man mithilfe eines Rechners (dynamische Programmierung!) feststellt.

Es sollte klar sein, dass in vielen Fällen ein reales funktionales Programm P, dessen Terminationsverhalten nicht vom Wertverlauf abhängt, zu einem Programm P' im obigen Sinne abstrahiert werden kann in einer solchen Weise, dass P' genau dann für alle Eingaben terminiert, wenn P es tut. Natürlich ist das nicht immer möglich, denn für unsere Programme ist die Frage nach der Termination für alle Eingaben entscheidbar, wie wir gleich sehen werden.

Vorher bemerken wir noch, dass solch eine automatische Terminationsanalyse nicht nur für Programme, sondern gerade auch für Beweise sehr nützlich ist. Induktive Beweise stellt man sich gern rekursiv vor; man verwendet die zu zeigende Aussage einfach und geht dabei davon aus, dass solche Verwendungen bezüglich irgendeines Maßes kleiner sind, als die gerade aktuelle. Ein solcher "rekursiver" Beweis ist natürlich nur dann gültig, wenn jeder "Aufruf" tatsächlich terminiert und ein, somit ist ein rechnergestützter Beweisprüfer, der solche Beweisstrategien anbietet, auf eine Terminationsanalyse angewiesen.

3.8.2 Termination als Sprachinklusion

Für den Rest dieses Kapitels sei ein Programm P mit Funktionssymbolen F vorgegeben. Es sei n die maximale Zahl von Funktionsaufrufen in der rechten Seite einer Gleichung.

Wir betrachten das Alphabet $\Sigma = F \times \{1, \ldots, n\}$ und definieren die ω-Sprache $CALL \subseteq F^\omega$ als die Menge aller unendlichen Aufrufsequenzen: ein Wort $(f_0, d_0)(f_1, d_1)(f_2, d_2)\ldots$ ist in $CALL$ genau dann, wenn ein Aufruf der Form $f_{k+1}(\vec{t})$ in der rechten Seite der definierenden Gleichung von f_k an d_k-ter Stelle vorkommt.

Wir lassen Klammern und Kommas weg und schreiben so ein Wort als $f_0 d_0 f_1 d_1 \ldots$. Im Beispiel mit $F = \{f, g, h, i, k\}$ enthält $CALL$ fünf Wörter, nämlich $w_f = (f1g1h1i1k1)^\omega$, $w_k = k1w_f$, $w_i = i1w_k$, $w_h = h1w_i$, $w_g = g1w_h$.

Im Beispiel mit $F = \{t\}$ ist $CALL = (t1 + t2 + t3)^\omega$

Im Beispiel mit $F = \{m, p\}$ schließlich wäre $CALL = ((m1 + m2)p1)^\omega + (p1(m1 + m2))^\omega$.

Eine Aufrufsequenz terminiert, wenn es eine Variable gibt, die immer wieder dekrementiert wird. Denn dann kann ja ein noch so hoher Startwert irgendwann zu Null reduziert werden und damit die Termination herbeiführen. Natürlich muss man verlangen, dass jede Aufrufsequenz in diesem Sinne terminiert und die dies bezeugende Variable kann immer jeweils eine andere sein. Es sei $TERM$ die Sprache der in diesem Sinne terminierenden Aufrufsequenzen.

Es gilt im Beispiel "Multiplikation", dass $(m2)^\omega \in TERM$, da hier die Variable y immer wieder dekrementiert wird. Im Beispiel "Künstlich II" gilt $(t1)^\omega \in TERM$, ja sogar $(t1 + t3)^\omega \subseteq TERM$. Auf der anderen Seite ist $(t2t3t1t3t1t3)^\omega \notin TERM$.

Das gesamte Programm P wird für alle Aufrufe terminieren genau dann, wenn $CALL \subseteq TERM$ gilt (was eben bei "Künstlich II" nicht der Fall ist.)

3.8.3 Terminationsanalyse mit Büchiautomaten

Die Sprache *TERM* ist aber gerade durch den folgenden Büchiautomaten $\mathcal{A}$ über dem Alphabet Σ erkennbar. Es sei m die maximale Stelligkeit eines Funktionssymbols in F.

- Zustandsmenge: $Q := \{1, \ldots, m\} \times \{0, 1\}$
- Startzustand beliebig (d.h. $I := Q$)
- Übergangsfunktion: $\delta((i, \_), (f, d))$ ergibt sich wie folgt: Sei $g(u_1, \ldots, u_m)$ der d-te Aufruf in der definierenden Gleichung für $f(x_1, \ldots, x_m)$ (der Einfachheit halber nehmen wir an, dass alle Symbole Stelligkeit m haben, ansonsten müsste man noch einen Papierkorbzustand einführen.). Ist $u_j = x_i$, so nehmen wir $(j, 0)$ in $\delta((i, \_), (f, d))$ auf. Ist $u_j = x_i - 1$, so nehmen wir $(j, 1)$ in $\delta((i, \_), (f, d))$ auf.
- Endzustände sind alle Zustände der Form $(i, 1)$.

Im Beispiel "Multiplikation" haben wir konkret:

$$\begin{aligned}
\delta((1, _), (p, 1)) &= \{(1, 0)\} \\
\delta((2, _), (p, 1)) &= \{(2, 1)\} \\
\delta((1, _), (m, 1)) &= \{(1, 0)\} \\
\delta((2, _), (m, 1)) &= \{\} \\
\delta((3, _), (m, 1)) &= \{(2, 0)\} \\
\delta((1, _), (m, 2)) &= \{(1, 0)\} \\
\delta((2, _), (m, 2)) &= \{(2, 1)\} \\
\delta((3, _), (m, 2)) &= \{(3, 1)\}
\end{aligned}$$

Im Beispiel "Künstlich I" haben wir einen nichtdeterministischen Übergang:

$$\delta((1, \_), (f, 1)) = \{(1, 0), (2, 0)\}$$

Die vollständige Übergangsfunktion geben wir hier nicht an.

Es sollte nunmehr klar sein, dass $TERM = L(\mathcal{A})$ und somit P für alle Eingaben terminiert, genau dann, wenn $CALL \subseteq L(\mathcal{A})$. Dies aber lässt sich mit den im letzten Kapitel gegebenen Algorithmen durchführen: Konkret konstruiert man durch Komplementierung und Produktbildung einen Automaten für $CALL \cap \overline{TERM}$ und stellt fest, ob letzterer Automat ein Wort erkennt. Ein solches entspricht gerade einer nichtterminierenden Aufrufsequenz.

3.8.4 Büchi-Inklusion durch Abschlussbildung

Jones et al merken an, dass die für dieses Standardverfahren erforderliche Komplementierung eines Büchiautomaten sei es nach Büchis Verfahren, sei es mit der Safrakonstruktion, in der Praxis aufwändig sei. Sie schlagen daher ein alternatives Verfahren vor, was auf die spezielle Anwendung zugeschnitten ist und der Praxis recht gut funktioniert. Es sei angemerkt, dass auch Safras Dissertation ein ähnliches Verfahren im Zusammenhang mit Streettautomaten enthält.

Definition 22
Seien f, g Funktionssymbole der Stelligkeiten m_1 und m_2. Ein Morphismus von f nach g ist eine Funktion $\alpha : \{1, \ldots, m_1\} \times \{1, \ldots, m_2\} \to \{0, 1, 2\}$. Man schreibt dann $\alpha : f \to g$. Ist $\alpha : f \to g$ und $\beta : g \to h$ so ist die Komposition $\alpha; \beta : f \to h$ definiert durch

$$(\alpha;\beta)(i,j) = \begin{cases} 0, \text{ falls } \forall k.\, 0 \in \{\alpha(i,k), \beta(k,j)\} \\ 1, \text{ falls } \exists k.\, \{\alpha(i,k), \beta(k,j)\} \in \{\{1\}, \{1,2\}\} \\ 2, \text{ sonst} \end{cases}$$

Die Intuition ist, dass ein Morphismus die Argumente von f mit denen von g verbindet, wobei $\alpha(i, j) = 0/1/2$ bedeutet, dass das i-te Argument von f mit dem j-ten Argument nicht verbunden ist / verbunden ist und abnimmt / verbunden ist. Bei Jones et al heißen die Morphismen size-change graphs.

Definition 23
Sei f ein Funktionssymbol und $d \in \mathbb{N}$ so dass die definierende Gleichung für f mindestens d Aufrufe enthält. Sei $g(u_1, \ldots, u_m)$ das d-te Funktionssymbol auf der rechten Seite der definierenden Gleichung für f. Der Morphismus $\alpha(f, d) : f \to g$ ist dann definiert durch

$$\alpha(f,d)(i,j) = \begin{cases} 2, \text{ falls } u_j = x_i \\ 1, \text{ falls } u_j = x_i - 1 \\ 0, \text{ sonst} \end{cases}$$

Ein Tripel (f, g, α) mit $\alpha : f \to g$ heißt *Morphismus*; f ist die *Quelle* und g das *Ziel* des Morphismus. Man identifiziert einen Morphismus (f, g, α) gern mit α selbst und schreibt dann $f = \mathrm{dom}(\alpha)$ (*domain*) und $g = \mathrm{cod}(\alpha)$ (*codomain*).

Zwei Morphismen α, β sind *komponierbar*, wenn $\mathrm{cod}(\alpha) = \mathrm{dom}(\beta)$. Ihre Komposition $\alpha; \beta$ ist dann definiert als $(\mathrm{dom}(\alpha), \mathrm{cod}(\beta), \alpha; \beta)$. Analog definiert man komponierbare Folgen von Morphismen und, bei endlichen Folgen, deren Komposition.

Jedes Wort $w = f_0 d_0 f_1 d_1 f_2 d_2 \ldots$ aus *CALL* induziert eine unendliche komponierbare Folge von Morphismen $\hat{w} = \alpha_0 \alpha_1 \alpha_2 \ldots$, wobei $\alpha_i = \alpha(f_i, d_i)$ und somit $\alpha_i : f_i \to f_{i+1}$.

Sei M die (endliche) Menge aller Morphismen der Form $\alpha(f, d)$.

Lemma 16
Sei $W = \alpha_0 \alpha_1 \alpha_2 \ldots$ ein endliches oder unendliches Wort über M. Das folgende ist äquivalent:

- *W ist komponierbar*
- *Es existiert $w \in CALL$ sodass $W = \hat{w}$.*

Beweis Übung

Ein komponierbares Wort über M ist in *TERM*, genau dann wenn eine Indexfolge $n_0 < n_1 < n_2 < \ldots$ und Positionen $l_0, l_1, \ldots$ existieren, sodass $(\alpha_{n_i}; \ldots; \alpha_{n_{i+1}-1})(l_i, l_{i+1}) = 1$ für alle i, denn dann wird eine Variable immer wieder dekrementiert. Uns geht es nun darum, herauszufinden, ob dies für eines dieser Wörter nicht der Fall ist.

Lemma 17
Sei $W = \alpha_0\alpha_1\alpha_2\ldots$ komponierbar, also $W = \hat{w}$ für $w \in CALL$. Es existiert eine Indexfolge $0 < n_0 < n_1 < n_2 < \ldots$ und Morphismen α, β, sodass:

- $\alpha_0; \alpha_1; \ldots; \alpha_{n_0-1} = \alpha$
- $\alpha_{n_i}; \alpha_{n_i+1}; \ldots; \alpha_{n_{i+1}-1} = \beta$ *für alle* $i \geq 0$
- $\beta; \beta = \beta$

Beweis Folgt unmittelbar aus Ramseys Theorem, wenn man die Menge der Morphismen als Farben nimmt. ∎

Ein Morphismus β mit $\beta = \beta; \beta$ heißt idempotent.

Satz 33
Sei $w = f_0d_0f_1d_1f_2d_2\cdots \in CALL$, sei $W = \hat{w} = \alpha(f_0, d_0)\alpha(f_1, d_1)\ldots$ das zugehörige komponierbare Wort über M. Sei $0 < n_0 < n_1 < \ldots$ eine Indexfolge wie in Lemma 17 und α, β die entsprechenden Morphismen, β idempotent.

Es ist $w \in TERM$, genau dann, wenn Positionen i, j existieren mit $\alpha(i, j) \geq 1$ und $\beta(j, j) = 1$.

Beweis Die Bedingung ist offensichtlich hinreichend. Für die Notwendigkeit sei

$$(i_0, f_0)(i_1, f_1)(i_2, f_2)\ldots$$

akzeptierender Lauf des Büchiautomaten auf w. Betrachte die Folge $i_{n_0}, i_{n_1}, \ldots$. Aufgrund der Konstruktion der Morphismen muss gelten $\beta(i_{n_j}, i_{n_{j+1}}) \geq 1$, also folgt wegen der Idempotenz von β, dass $i_{n_j} = i_{n_{j+1}}$ für hinreichend großes j und ab dann dann auch $\beta(i_{n_j}, i_{n_{j+1}}) = 1$. Die Behauptung ist damit klar. ∎

Zusammenfassend haben wir gezeigt:

Satz 34
Es existiert ein Wort $w \in CALL \setminus TERM$ genau dann wenn Morphismen $\alpha_1, \alpha_2, \ldots, \alpha_n \in M$ existieren, sodass für $\beta := \alpha_1; \alpha_2; \ldots; \alpha_n$ gilt:

- $\beta; \beta = \beta$
- *Für alle i gilt $\beta(i, i) \neq 1$.*

Die Bedingung dieses Theorems lässt sich aber effektiv überprüfen, indem man alle Komponate aus M (es sind nur endlich viele) systematisch berechnet.

3.9 Sternfreie Sprachen

Wir verlassen nunmehr die ω-Sprachen und studieren eine echte Teilklasse der regulären Sprachen: die *sternfreien Sprachen*. Wie die regulären Sprachen erlauben diese mehrere äquivalente Charakterisierungen: durch sternfreie reguläre Ausdrücke, durch erststufige Logik definierbar, in LTL definierbar.

Bevor wir ins Detail gehen können, müssen wir uns ein Hilfsmittel aus der endlichen Modelltheorie erarbeiten, das *Ehrenfeucht-Fraïssé-Spiel*.

3.9.1 Das Ehrenfeucht-Fraïssé-Spiel

Zwei Personen, genannt Spoiler (S) und Duplicator (D) spielen folgendes Spiel: Gegeben sind zwei Wörter u und v über einem Alphabet Σ, die untereinander geschrieben werden. Außerdem wird eine Zahl von Runden vereinbart.

S beginnt das Spiel, indem er auf eine Position in entweder u oder v zeigt. D antwortet, indem er auf eine Position im jeweils anderen Wort zeigt und die beiden Positionen durch eine Linie verbindet. Die Buchstaben an den beiden Positionen müssen dabei aber übereinstimmen. Außerdem darf dabei keine schon vorhandene Verbindungslinie gekreuzt werden. Sodann ist wieder S am Zug usw. Der Spieler S darf eine bereits gespielte Position nochmal spielen; in diesem Fall muss D mit der korrespondierenden Position antworten. In keinem anderen Fall darf D eine schon gespielte Position wiederholen.

Kann D nicht spielen, so verliert er das Spiel. Ist die vereinbarte Rundenzahl vorbei, ohne dass D verloren hätte, so gewinnt D. Wir sagen dann D gewinnt k Runden.

Beispiel I: $\Sigma = \{a, b\}$, $u = aabaacaa$, $v = aacaabaa$. Duplicator gewinnt $G_1(u, v)$, verliert aber $G_2(u, v)$ Um $G_2(u, v)$ zu gewinnen, würde S im ersten Zug das b in u spielen; D muss mit dem B in v antworten. In der zweiten Runde spielt S das c in u und D würde gerne c spielen, kann aber nicht, weil sich die Linien dann überkreuzen würden, und verliert.

Beispiel II: $\Sigma = \{a\}$, $|u| >= 2^k - 1$, $|v| >= 2^k - 1$. Hier kann Duplicator k Runden überleben.

Beispiel II: $\Sigma = \{a\}$, $|u| >= 2^k - 1$, $|v| >= 2^k - 1$. Hier kann Duplicator k Runden überleben (Binäre Suche).

Um geeignete Invarianten formulieren zu können, benötigen wir noch die folgende Verallgemeinerung: Seien u, v Wörter, $\vec{s} = (s_0, s_1, \ldots, s_{n-1})$ und $\vec{t} = (t_0, t_1, \ldots, t_{n-1})$ Zahlenfolgen mit $s_i < |u|, t_i < |v|$. Das Spiel $G_k((u, \vec{s}), (v, \vec{t}))$ wird so gespielt wie $G_k(u, v)$ mit der Ausnahme, dass die Verbindungen s_i—t_i bereits als vorhanden gelten, somit dürfen die Positionen s_i in u und t_i in v von D nicht mehr gespielt werden, es sei denn, S hätte zuvor die korrespondierende Position gespielt. Außerdem dürfen die Verbindungen s_i—t_i nicht gekreuzt werden. Sollten sich solche vorgegebenen Verbindungen s_i—t_i und s_j—t_j kreuzen (also z.B. $s_i < s_j$ und $t_j < t_i$) oder verschiedene Buchstaben verbinden, so hat D sofort verloren, also schon nach 0 Runden.

Der Satz von Ehrenfeucht und Fraïssé besagt, dass ein Spiel $G_k(u, v)$ von D genau dann gewonnen wird, wenn u und v durch Formeln der erststufigen Logik mit Quantorentiefe k nicht zu unterscheiden sind. Wenn bis auf weiteres von Formeln die Rede ist, so sind immer Formeln der erststufigen Logik über der Signatur $P_a(x)$ für $a \in \Sigma$ und $x < y$ gemeint.

Formeln der Quantorentiefe Null sind Boole'sche Kombinationen von atomaren Formeln, also $P_a(x)$ für a in Σ und $x < y$.

Formeln der Quantorentiefe $k+1$ sind von der Form $\exists x.\phi(x)$, wobei ϕ von Quantorentiefe k ist und außerdem Boole'sche Kombinationen solcher Formeln.

Ist ϕ Formel mit n freien Variablen $x_0, \ldots, x_{n-1}$ und $\vec{s} = (s_0, \ldots, s_{n-1})$, so schreiben wir $u, \vec{s} \models \phi$ um zu sagen, dass ϕ wahr ist unter der durch u gegebenen Interpretation der Prädikatsymbole und der durch $\vec{s}$ gegebenen Belegung der freien Variablen. Alternativ schreiben wir auch $u \models \phi(\vec{s})$. Mit der Notation aus Definition 7 $u, \vec{s} \models \phi \iff u \models_I \phi$, wobei $I(x_i) = s_i$.

Definition 24
Seien u, v Wörter. Wir schreiben $u \equiv_k v$, wenn gilt $u| = \phi \iff v \models \phi$ fuer alle geschlossenen Formeln ϕ der Quantorentiefe k. Seien u, v Wörter und $\vec{s} = (s_0, \ldots, s_{n-1})$, $\vec{t} = (t_0, \ldots, t_{n-1})$. Wir schreiben $(u, \vec{s}) \simeq_{k,n} (v, \vec{t})$, wenn gilt

$$u, \vec{s} \models \phi \iff v, \vec{t} \models \phi$$

für alle Formeln ϕ mit n freien Variablen und Quantorentiefe k.

Satz 35 (Ehrenfeucht-Fraïssé)
Das Spiel $G_k((u, \vec{s}), (v, \vec{t}))$ wird von Duplicator genau dann gewonnen, wenn gilt $(u, \vec{s}) \simeq_{k,n} (v, \vec{t})$.

Beweis "⇒": Durch Induktion über k. Duplicator gewinnt 0 Runden, wenn $s_i < s_j \iff t_i < t_j$ und $u(s_i) = v(t_i)$ für alle i. Dann aber gilt für jede atomare Formel ϕ, dass $u \models \phi(\vec{s}) \iff v \models \phi(\vec{t})$ und durch Induktion über den Formelaufbau setzt sich dies auf beliebige quantorenfreie Formeln fort. Sei nun die Behauptung für k schon gezeigt und gewinne D $k+1$ Runden. Wir zeigen wiederum durch Induktion über den Formelaufbau, dass $u \models \phi(\vec{s}) \iff v \models \phi(\vec{t})$. Der interessante Fall ist $\phi = \exists x.\phi$. Wenn $u \models \phi(\vec{s})$, so existiert eine Position s, derart dass $u \models \psi(s, \vec{s})$. Wir betrachten den Fall, in dem S mit der Position s eröffnet. Da ja nach Voraussetzung D eine Gewinnstrategie besitzt, muss er diesen Zug mit einer Position t beantworten können. Das Restspiel $G_k((u, s.\vec{s}), (v, t.\vec{t}))$ wird nun von D gewonnen, nach Induktionsvoraussetzung haben wir also $u, s.\vec{s} \simeq_{k,n+1} v, t.\vec{t}$ und es folgt $v \models \psi(t, \vec{t})$.

"⇐" Die Idee ist, zu jedem $(u, \vec{s})$ und k eine Formel $\chi = \chi^{(n,k)}_{(u,\vec{s})}$ anzugeben, derart dass $u, \vec{s} \models \chi$ gilt und weiterhin aus $v, \vec{t} \models \chi$ folgt, dass D das Spiel $G_k((u, \vec{s}), (v, \vec{t}))$ gewinnt. Es sollte klar sein, dass die Existenz solch einer Formel die Behauptung liefert. Wir konstruieren die Formel durch Induktion über k. Falls $k = 0$, so wählen wir für χ die Konjunktion der folgenden Bedingungen:

$$\begin{array}{ll} a(x_i) & \text{, falls } u_{s_i} = a \\ x_i < x_j & \text{, falls } s_i < s_j \end{array}$$

Es ist klar, dass $u, \vec{s} \models \chi$. Wenn jetzt auch $v, \vec{t} \models \chi$, so folgt, dass D das Spiel $G_0((u, \vec{s}), (v, \vec{t}))$ gewinnt. Falls $k > 0$, so setzen wir

$$\chi(\vec{x}) = \bigwedge_{i=0}^{|u|-1} \exists x.\chi_i(x, \vec{x}) \wedge \forall x. \bigvee_{i=0}^{|u|-1} .\chi_i(x, \vec{x})$$

wobei χ_i die charakterisierende Formel fuer $(k-1, n+1)$ und $(u, i.\vec{s})$ ist. Klar, dass $(u, \vec{s})$ dieses χ wahrmacht: Gegeben i, wählen wir einfach i für x. Gegeben x, wählen wir einfach x für i. Wenn jetzt $(v, \vec{t}) \models \chi$, so beschreiben wir eine Gewinnstrategie für D. Beginnt S in u, so liefert das erste Konjunkt die entsprechende Antwort; beginnt S in v, so nehmen wir das zweite Konjunkt her. ■

Als einfache Anwendung zeigen wir folgenden Satz:

Satz 36
Die Sprache $L = (aa)^*$ lässt sich nicht durch FOL definieren, d.h. es gibt keine Formel ϕ der FOL, sodass $u \models \phi$, gdw $u \in (aa)^*$.

Beweis Gäbe es so ein ϕ, dann hätte es eine bestimmte Quantorentiefe k. Das Spiel $G_k(a^{2^k}, a^{2^k-1})$ wird von D gewonnen, somit sind die beiden Wörter durch FOL Formeln der Quantorentiefe k nicht zu unterscheiden, also insbesondere nicht durch ϕ. Das eine Wort ist aber in L, das andere nicht. Widerspruch. ■

Man beachte die Analogie mit dem Pumpinglemma fuer reguläre Sprachen.

3.9.2 Sternfreie Sprachen durch Abschlusseigenschaften

Definition 25
Sei Σ ein Alphabet. Die sternfreien Sprachen über Σ, geschrieben $\mathrm{SF}(\Sigma)$ bilden die kleinste Klasse von Sprachen über Σ, die unter folgendem abgeschlossen ist:

- $\emptyset$ ist sternfrei.
- $\{\epsilon\}$ ist sternfrei.
- $\{a\}$ ist sternfrei für $a \in \Sigma$.
- Sind A, B sternfrei, so auch AB, $\overline{A}$, $A \cup B$

Beispiele: Σ^* ist sternfrei, da $\Sigma^* = \overline{\emptyset}$. Sind A, B sternfrei, so auch $A \cap B = \overline{\overline{A} \cup \overline{B}}$ und $A \setminus B = A \cap \overline{B}$. Falls $D \subseteq \Sigma$, so ist D^* sternfrei, da $D^* = \Sigma^* \setminus (\Sigma^*(\Sigma \setminus D)\Sigma^*)$. Die Sprache $(ab)^*$ ist auch sternfrei, da $(ab)^* = \Sigma^* \setminus b\Sigma^* \setminus \Sigma^* aa\Sigma^* \setminus \Sigma^* bb\Sigma^*$.

Satz 37
Sei L eine sternfreie Sprache mit $\epsilon \notin L$. Es gibt eine FOL Formel ϕ, sodass $w \in L \iff w \models \phi$.

Beweis Ist L sternfrei, so ist $L \setminus \{\epsilon\}$ FOL-definierbar. Wir beweisen das durch Induktion über die Definition von "sternfrei": $\emptyset$, $\{a\}$ sind alle FOL definierbar. , z.B.: $\emptyset = L(\neg(\forall x.x = x)))), \{a\} = L(\forall x, y.x = y \wedge a(x))$. Sei nun $L = \overline{L}_1$ und $L_2 = L_1 \setminus \epsilon$. Nach Induktionsvoraussetzung ist $L_2 = L(\phi)$ für eine Formel ϕ. Es gilt $L = L(\neg\phi)$.

Sei nun $L = L_1 \cup L_2$ und $L_1 = L(\phi_1), L_2 = L(\phi_2)$. Es ist $L = L(\phi_1 \vee \phi_2)$.

Sei nun $L = L_1 L_2$. Wir nehmen o.B.d.A. an, dass ϵ nicht in L_1, L_2 enthalten ist. Sei $L_1 = L(\phi_1)$, $L_2 = L(\phi_2)$. Es ist $L = L(\exists t.\phi_1 \restriction \{0 \dots t\} \wedge \phi_2 \restriction \{t+1 \dots\})$. Hierbei

bezeichnet $\phi \upharpoonright \{u \dots v\}$ die Formel, die man aus ϕ durch Relativierung aller Quantoren auf den Bereich $\{u \dots v\}$ erhält. Man ersetzt also ein Vorkommen von $\exists y. \dots$ durch $\exists y : (u \leq y \wedge y \leq v) \wedge \dots$. ∎

Für die Umkehrung dieses Satzes (FOL-definierbare Sprachen sind sternfrei) benötigen wir etwas Vorarbeit.

Lemma 18
Die Relation $\equiv_k$ (Ununterscheidbarkeit durch Formeln der Quantorentiefe k) ist Äquivalenzrelation mit endlich vielen Äquivalenzklassen.

Beweis Direkte Folgerung aus der Tatsache, dass es bis auf logische Äquivalenz nur endliche viele Formeln einer festen QT k gibt. Jede Äquivalenzklasse ist eindeutig bestimmt durch die Menge der Formeln, die in ihr gelten. Gibt es also höchstens p Formeln, dann gibt es höchstens 2^p Äquivalenzklassen. ∎

Lemma 19
Zu jeder $\equiv_k$-Äquivalenzklasse W gibt es eine Formel ϕ_W, sodass $(u, \vec{s}) \in W \iff (u, \vec{s}) \models \phi$.

Beweis Man nimmt die Konjunktion aller (bis auf Äquivalenz) Formeln der QT k, die in W gelten. ∎

Lemma 20
Jede durch eine Formel der QT k definierte Sprache ist (endliche) Vereinigung von $\equiv_k$-Äquivalenzklassen.

Beweis Ist ein Wort u in L, so erfüllt es die definierende Formel. Jedes äquivalente Wort erfüllt dann aber auch diese Formel, ist somit auch in L. ∎

Satz 38
Sei ϕ eine Formel der FOL. Die Sprache $L(\phi)$ ist sternfrei.

Beweis Induktion über die Quantorentiefe k und den Formelaufbau.

Induktionsanfang: die einzigen (bis auf Äquivalenz) geschlossenen Formeln von QT 0 sind die Formeln `tt` und `ff`. Beide definieren sternfreie Sprachen, nämlich $\emptyset$ und Σ^+.

Induktionsschritt: Habe ϕ die QT $k+1$. Boole'sche Operationen sind auf der Ebene der sternfreien Ausdrücke vorhanden; somit können wir uns auf den Fall $\phi = \exists x : \psi$ beschränken, wobei ψ die QT k hat. Wir dürfen die Behauptung für ψ und alle anderen Formeln der QT k voraussetzen. Insbesondere ist jede $\equiv_k$-Äquivalenzklasse sternfrei.

Wir behaupten nun folgendes:

$$L(\phi) = L(\exists x.\psi) = \bigcup_{(uav,|u|)\models\psi} [u]_{\equiv_k} a [v]_{\equiv_k}$$

Man beachte, dass dies eine endliche Vereinigung ist, da es ja überhaupt nur endlich viele $\equiv_k$-Äquivalenzklassen gibt.

Zum Beweis der Behauptung nehmen wir an, dass $w \models \phi$. Somit ist $w = uav$ und $(uav, |u|) \models \psi(x)$. Dann aber ist w in $[u]_{\equiv_k} a [v]_{\equiv_k}$.

Wenn umgekehrt $w = uav, u \equiv_k u'$, $v \equiv_k v'$ und $u'av', |u'| \models \psi(x)$, dann gilt mithilfe des Satzes von Ehrenfeucht-Fraïssé auch $(uav, |u|) \equiv_k (u'av', |u'|)$, somit $(uav, |u|) \models \psi$, somit $w \models \phi$. Hierzu argumentieren wir wie folgt: Nach dem Satz von Ehrenfeucht-Fraïssé gewinnt D die Spiele $G_k(u, u')$ und $G_k(v, v')$. Duplicator gewinnt dann auch das Spiel $G_k((uav, |u|), (u'av', |u'|))$, denn Züge von S auf u oder u' werden entsprechend der Gewinnstrategie für $G_k(u, u')$ beantwortet, analog fuer (v, v'). Die Verbindung a—a ist ja schon gespielt. Eine abermalige Anwendung des Satzes von Ehrenfeucht-Fraïssé liefert dann $(u'av', |u'|) \models \psi$, also $u'av' \models \phi$.

3.10 Alternierende Automaten

In diesem Kapitel erweitern wir das Konzept des Nichtdeterminismus, welches einen Automaten "raten" lässt, welcher Nachfolgezustand in einer gegebenen Situation gut ist. Dazu führen wir ein duales Konzept ein – das der *universellen Wahl*. Dies lässt einen Automaten sozusagen raten, welcher Nachfolgezustand in einer gegebenen Situation schlecht ist. Aus einem anderen Blickwinkel betrachtet bedeutet dies, dass der Automat bei einer universellen Verzweigung sich in mehrere Kopien verzweigt, die *alle* akzeptieren müssen, damit der Automat insgesamt akzeptiert. Beachte, dass man Nichtdeterminismus auch als Verzweigung in mehrere Kopien ansehen kann, von denen aber nur eine akzeptieren muss.

Lässt man beide Arten der Verzweigung innerhalb eines Automaten zu, dann spricht man von alternierenden Automaten.

Wie bei nicht-deterministischen Automaten bergen alternierdene Automaten auf unendlichen Wörtern wieder zusätzliche Schwierigkeiten zu denen auf endlichen Wörtern. Deswegen betrachten wir zunächst letztere.

3.10.1 Alternierende Automaten auf endlichen Wörtern

Definition 26
Sei Q eine Menge. Die Menge $\mathbb{B}^+(Q)$ der *positiv booleschen Formeln* über Q ist die kleinste Menge für die gilt:

- $Q \subseteq \mathbb{B}^+(Q)$,
- wenn $f, g \in \mathbb{B}^+(Q)$, dann $f \vee g, f \wedge g \in \mathbb{B}^+(Q)$.

Sei $M \subseteq Q$ und $f \in \mathbb{B}^+(Q)$. Wir sagen, dass M ein *Modell* von f ist, geschrieben $M \models f$, falls f unter den üblichen Regeln für $\vee$ und $\wedge$ als propositionale Formel zu 1 auswertet, wenn alle $q \in M$ in f durch 1 und alle $q \in Q \setminus M$ durch 0 ersetzt werden. Ein Modell M heisst *minimal*, falls für alle $N \subsetneq M$ gilt: $N \not\models f$. In diesem Fall schreiben wir $M \models_{min} f$.

Lemma 21
Seien $f, g \in \mathbb{B}^+(Q)$ und $P_f, P_g \subseteq Q$ mit $P_f \models_{min} f$ und $P_g \models_{min} g$. Dann gilt $P_f \cup P_g \models_{min} f \wedge g$.

Beweis Übung. ∎

Definition 27
Ein Q-beschrifteter Baum ist ein Baum r mit einer Funktion, die jedem Knoten $n \in dom(r)$ – der Knotenmenge – ein Element aus Q zuordnet. Wir schreiben einfach $r(n)$ für die Beschriftung von n.

Die *Tiefe* $depth(n)$ eines Knotens n ist induktiv definiert wie üblich: $depth(n) = 0$ für die Wurzel n von r, und $depth(n) = depth(n') + 1$, falls n Sohn von n' ist. Die i-te Ebene von r besteht aus allen Beschriftungen von Knoten der Tiefe i: $level_i := \{n \mid depth(n) = i\}$.

Mit $r|_n$ bezeichnen wir den Unterbaum unter dem Knoten n inklusive.

Definition 28
Ein *alternierender Automat* (AFA) ist ein Tupel $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$, wie bei einem NFA, aber mit dem Unterschied $\delta : Q \times \Sigma \to \mathbb{B}^+(Q)$.

Ein *Lauf* dieses AFA $\mathcal{A}$ auf einem Wort $w = a_1 \dots a_n \in \Sigma^*$ ist ein Q-beschrifteter, endlicher Baum r, mit den folgenden Eigenschaften.

- Die Wurzel ist beschriftet mit q_0.
- Ist ein Knoten in Tiefe i mit $0 \leq i \leq n$ beschriftet mit einem $q \in Q$, und sind seine Kinder beschriftet mit $M := \{q_1, \dots, q_m\}$, dann ist $M \models_{min} \delta(q, a_{i+1})$.

Der Lauf r heisst akzeptierend, falls alle Blätter von r auf der Ebene $|w|$ liegen und mit einem $q \in F$ beschriftet sind.

Wie üblich definieren wir wieder $L(\mathcal{A}) = \{w \in \Sigma^* \mid$ es gibt einen akzeptierenden Lauf von $\mathcal{A}$ auf $w\}$.

Beispiel 7
Es sollte klar sein, dass die Sprache $L_{29393} = \{w \in \Sigma^* \mid |w| \equiv 0 \mod 29393\}$ nicht von einem NFA mit weniger als 29393 Zuständen erkannt werden kann. Sie kann jedoch von einem AFA mit 57 Zuständen erkannt werden. Beachte, dass gilt:

$$\begin{aligned} 29393 &= 7 \cdot 13 \cdot 17 \cdot 19 \\ 56 &= 7 + 13 + 17 + 19 \end{aligned}$$

Seien $\mathcal{A}_i = (Q_i, \Sigma, q_{0,i}, \delta_i, F_i)$ jeweils die Standard-NFAs für die Sprachen $L_i = \{w \in \Sigma^* \mid |w| \equiv 0 \mod i\}$. Deren Transitionsrelation bildet einen Zykel auf Q_i, und $F_i = \{q_{0,i}\}$.

Sei nun $\mathcal{A}_{29393} = (\{q_0\} \uplus Q_7 \uplus Q_{13} \uplus Q_{17} \uplus Q_{19}, \Sigma, q_0, \delta, \{q_0\} \uplus F_7 \uplus F_{13} \uplus F_{17} \uplus F_{19})$ mit

$$\begin{aligned} \delta(q_0, a) &:= \delta_7(q_{0,7}, a) \wedge \delta_{13}(q_{0,13}, a) \wedge \delta_{17}(q_{0,17}, a) \wedge \delta_{19}(q_{0,19}, a) \\ \delta(q, a) &:= \delta_j(q, a) \qquad \text{falls } q \in Q_j \end{aligned}$$

Man sieht leicht, dass gilt: $L(\mathcal{A}_{29393}) = L_{29393}$.

Beispiel 8
AFAs bieten jedoch auch die Möglichkeit, universelle Verzweigungen beliebig tief (in einem Lauf) zu verschachteln. Beachte, dass universelle Wahl im vorigen Beispiel nur benutzt wurde, um den Durchschnitt mehrerer, von NFAs erkannten Sprachen zu erkennen.

Sei $L = \{w \in \{a,b\}^* \mid \text{auf jedes } a \text{ folgt irgendwann noch ein } b\}$. Diese Sprache wird von dem AFA $\mathcal{A} = (\{q_0, q_1, q_2\}, \{a,b\}, q_a, \delta, \{q_0, q_2\})$ mit

$$\begin{aligned} \delta(q_0, a) &:= q_0 \wedge q_1 \\ \delta(q_0, b) &:= q_0 \\ \delta(q_1, a) &:= q_1 \\ \delta(q_1, b) &:= q_2 \\ \delta(q_2, \_) &:= q_2 \end{aligned}$$

erkannt.

Es ist nicht schwer zu sehen, dass alternierende Automaten mindestens so ausdrucksstark wie nicht-deterministische sind. Beachte, dass sich der Nichtdeterminismus leicht durch das boolesche $\vee$ modellieren lässt.

Satz 39
Für jeden NFA $\mathcal{A}$ mit n Zuständen existiert ein AFA $\mathcal{A}'$ mit höchstens n Zuständen, so dass gilt: $L(\mathcal{A}') = L(\mathcal{A})$.

Um die Umkehrung zu zeigen, brauchen wir noch ein wenig technisches Rüstzeug.

Definition 29
Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein AFA. Ein Lauf r auf einem Wort $w \in \Sigma^*$ heisst *gedächtnislos*, falls für alle $i = 0, \ldots, n$ und alle Knoten n, n' mit $\mathit{depth}(n) = \mathit{depth}(n')$ gilt: wenn $r(n) = r(n')$ dann ist $r|_n = r|_{n'}$.

Intuitiv ist ein Lauf gedächtnislos, wenn die gleichen Missionen in derselben Weise erfüllt werden. Wenn der Automat also in verschiedenen Kopien zum gleichen Zeitpunkt in denselben Zustand kommt, dann soll er sich danach gleich verhalten. Einen gedächtnislosen Lauf kann man offensichtlich als gerichteten, azyklischen Graphen (DAG) darstellen. Da auch ein Baum ein DAG ist, gehen wir einfach im folgenden davon aus, dass ein Lauf immer ein DAG ist. Außerdem gehen wir davon aus, dass diese DAGs minimal sind. D.h. keine zwei verschiedenen Knoten haben die gleichen Nachfolger und die gleichen Beschriftungen.

Wir gehen im folgenden davon aus, dass jeder ein gedächtnisloser Lauf eines Automaten mit n Zuständen als DAG der maximalen Breite n repräsentiert ist.

Lemma 22
Sei $\mathcal{A}$ ein AFA und $w \in \Sigma^*$. Wenn $w \in L(\mathcal{A})$, dann gibt es einen gedächtnislosen und akzeptierenden Lauf von $\mathcal{A}$ auf w.

BEWEIS Sei r ein akzeptierenden Lauf von $\mathcal{A}$ auf w. Da minimale Modelle einer endlichen propositionalen Formel immer endlich sind, ist r endlich verzweigend. Da die Tiefe von r ebenfalls durch $|w|$ beschränkt und damit endlich ist, ist r insgesamt endlich. Seien nun n und n' zwei Knoten auf derselben Tiefe mit unterschiedlichen Unterbäumen, aber derselben Beschriftung. Offensichtlich kann man $r|_{n'}$ durch $r|_n$ ersetzen oder umgekehrt, wodurch man wiederum einen akzeptierenden Lauf von $\mathcal{A}$ auf w erhält. Da es in einem endlichen Baum nur endlich viele solcher Paare geben kann, lässt sich der Lauf sukzessive in einen gedächtnislosen und weiterhin akzeptierenden Lauf umwandeln. ■

Satz 40
Sei $\mathcal{A}$ ein AFA mit n Zuständen. Dann gibt es einen NFA $\mathcal{A}'$ mit höchstens 2^n vielen Zuständen, so dass $L(\mathcal{A}') = L(\mathcal{A})$ gilt.

BEWEIS Analog zur Umwandlung eines NFA in einen DFA. Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$. Wir lassen $\mathcal{A}'$ in jedem Schritt eine Ebene eines gedächtnislosen Laufs von $\mathcal{A}$ auf einem Eingabewort w erraten. Sei $\mathcal{A}' := (2^Q, \Sigma, \{q_0\}, \delta', F')$ mit $F' := \{Q' \mid Q' \subseteq F\}$ und

$$\delta'(Q', a) \;:=\; \{Q'' \mid Q'' \models_{min} \bigwedge_{q \in Q'} \delta(q, a)\}$$

Die Größenbeschränkung von $\mathcal{A}'$ ergibt sich daraus sofort. Es bleibt $L(\mathcal{A}') = L(\mathcal{A})$ zu zeigen.

"$\supseteq$" Sei $w \in L(\mathcal{A})$. Laut Lemma 22 existiert dann ein gedächtnisloser Lauf r von $\mathcal{A}$ auf w. Man sieht mit Lemma 21 leicht, dass $level_0^r, level_1^r, \ldots, level_n^r$ für $n := |w|$ ein akzeptierender Lauf von $\mathcal{A}'$ auf w ist. Insbesondere gilt $level_0 = \{q_0\}$ und $level_n \subseteq F$.

"$\subseteq$" Sei $w \in L(\mathcal{A}')$ und $Q_0, Q_1, \ldots, Q_n$ ein akzeptierender Lauf von $\mathcal{A}'$ auf w. Daraus lässt sich ebenso leicht ein gedächtnisloser Lauf von $\mathcal{A}$ auf w konstruieren. Dieser ist ebenfalls akzeptierend, da $Q_n \subseteq F$ sein muss. ■

3.10.2 Alternierende Büchi-Automaten

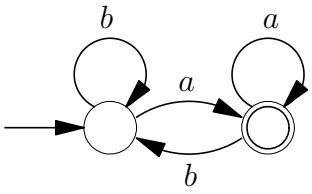
Definition 30
Ein *alternierender Büchi-Automat* (ABA) ist ein Tupel $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ wie bei einem AFA. Ein Lauf eines ABA auf einem Wort $w \in \Sigma^\omega$ is ein unendlicher, Q-beschrifteter Baum – definiert wie bei einem Lauf eines AFA auf einem endlichen Wort. Solch ein Lauf heißt *akzeptierend*, falls es auf jedem Ast des Baumes unendlich viele Knoten $n_0, n_1, \ldots$, so dass $r(n_i) \in F$ für alle $i \in \mathbb{N}$ gilt.

Wieder sieht man leicht, dass alternierende Automaten mindestens so ausdruckstark wie nicht-deterministische sind.

Satz 41
Für jeden NBA $\mathcal{A}$ mit n Zuständen existiert ein ABA $\mathcal{A}'$ mit höchstens n Zuständen, so dass $L(\mathcal{A}') = L(\mathcal{A})$ gilt.

BEWEIS Analog zum Beweis von Satz 39. ■

Beispiel 9
Betrachte die ω-reguläre Sprache $L := \{w \mid |w|_a = \infty\}$. Diese wird z.B. von dem NBA



erkannt. Somit existiert nach Satz 41 auch ein ABA, der L erkennt. Man kann jedoch L auch unter echter Verwendung von Alternierung mit einem ABA erkennen. Das Prinzip dabei ist das folgende. Der ABA liest das nächste Symbol des Eingabeworts. Ist dies ein a, so kann er im selben Zustand verbleiben, denn er muss lediglich prüfen, ob danach wieder ein a vorkommt, usw. Ist dieses jedoch nicht ein a, dann verzweigt er universell. Ein Teil verbleibt in dem Zustand, in dem auch im nächsten Schritt geprüft wird, ob später wieder ein a vorkommt. Der andere testet, ob wirklich irgendwann noch ein a vorkommt.

Sei $\mathcal{A} = (\{q_0, q_1, q_2\}, \Sigma, q_0, \delta, \{q_0, q_2\})$ mit

$$\begin{aligned}
\delta(q_0, a) &:= q_0 \\
\delta(q_0, _) &:= q_0 \wedge q_1 \\
\delta(q_1, a) &:= q_2 \\
\delta(q_1, _) &:= q_1 \\
\delta(q_2, _) &:= q_2
\end{aligned}$$

Es stellt sich die Frage, ob dieser ABA einen Vorteil gegenüber dem obigen NBA hat. Er ist zwar größer, aber strukturell simpler. Seine Zustandsmenge lässt sich in starke Zusammenhangskomponenten zerlegen, die jeweils entweder nur aus End- oder nur aus Nicht-Endzuständen bestehen. Wir werden später noch darauf zurückkommen.

Ein *gedächtnisloser Lauf* ist wie bei einem AFA definiert: auf einer Ebene des Laufes gibt es keine zwei verschiedenen Knoten mit derselben Beschriftung und unterschiedlichen Unterbäumen. Wiederum gilt, dass sich gedächtnislose Läufe als DAGs darstellen lassen, so dass jeder Level höchstens $|Q|$ viele Knoten enthält.

Der Beweis, dass es zu jedem akzeptierenden Lauf auch immer einen gedächtnislosen gibt, lässt sich jedoch nicht wie bei AFAs einfach durch sukzessives Umbauen führen. Bedenke, dass es in einem nicht-gedächtnislosen Lauf eines ABA unendlich viele Paare von Knoten geben kann, die die Eigenschaft der Gedächtnislosigkeit verletzen.

Definition 31
Sei t ein Lauf eines ABA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ auf einem Wort $w \in \Sigma^\omega$. Der *Rang* eines Knoten n, $rang(n)$ ist das maximale k in einer Sequenz $n_0, \ldots, n_k$, so dass

- $n = n_0$,
- für alle $i = 0, \ldots, k-1$ gilt: n_{i+1} ist ein Nachfolger von n_i,

- $n_k \in F$ und $n_i \notin F$ für alle $i < k$.

Insbesondere gilt $rang(n) = 0$, falls $t(n) \in F$ und $rang(n) = \infty$, falls es einen Pfad von n aus gibt, der niemals einen Endzustand besucht.

Der Rang eines Levels ist der maximale Rang eines seiner Knoten: Falls $N = level_i$ für ein i, dann gilt $rang(N) := \max\{rang(n) \mid n \in N\}$.

Lemma 23
Sei $\mathcal{A}$ ein ABA über Σ und t ein Lauf auf einem $w \in \Sigma^\omega$. Die folgenden Aussagen sind äquivalent.

a) Der Lauf t ist akzeptierend.

b) Jeder Knoten in t hat endlichen Rang.

c) Jeder Level in t hat endlichen Rang.

d) Unendlich viele Level haben endlichen Rang.

Beweis "(a) $\Rightarrow$ (b)" Durch Widerspruch. Angenommen, es gäbe einen Knoten mit unendlichem Rang. Dann gibt es auch einen Pfad in t, auf dem nur endlich viele Endzustände vorkommen, womit t nicht akzeptierend ist.

"(b) $\Rightarrow$ (a)" Ebenfalls durch Widerspruch. Angenommen, der Lauf ist nicht akzeptierend. Dann muss es also einen Pfad geben, auf dem nicht unendlich oft Endzustände vorkommen. Man beachte, dass wegen der Definition der positiv booleschen Formeln jeder Zustand einen Nachfolger haben muss. Daher ist jeder Pfad in solch einem Lauf unendlich, und somit gibt es auf solch einem angenommenen Pfad einen letzten Endzustand, der einen Nachfolger hat. Dieser kann aber dann offensichtlich nicht endlichen Rang haben.

"(b) $\Rightarrow$ (c)" Folgt sofort daraus, dass Läufe nur endlich verzweigend sind und deswegen jeder Level nur endlich viele Knoten besitzt.

"(c) $\Rightarrow$ (b)" und "(c) $\Rightarrow$ (d)" sind trivial.

"(d) $\Rightarrow$ (c)" Seien N_1 und N_2 zwei aufeinanderfolgende Level in t. Man sieht leicht, dass der Rang von N_1 endlich ist, wenn der seines Nachfolgers N_2 endlich ist und N_1 selbst nur endlich viele Elemente hat. Genauer: Es gilt $rang(N_1) \leq rang(N_2) + 1$, da im schlimmsten Fall der Knoten mit maximalem Rang in N_2 einen Vorgänger in N_1 hat, der nicht mit einem Endzustand beschriftet ist.

Die Behauptung folgt nun aus der Tatsache, dass in einem Lauf, in dem unendlich viele Level endlichen Rang haben, auf jeden Level später einer mit endlichem Rang folgt. ■

Lemma 24
Sei $\mathcal{A}$ ein ABA über dem Alphabet Σ und $w \in \Sigma^\omega$. Falls $w \in L(\mathcal{A})$ dann gibt es einen gedächtnislosen und akzeptierenden Lauf von $\mathcal{A}$ auf w.

BEWEIS Angenommen $w \in L(\mathcal{A})$, d.h. es gibt einen akzeptierenden Lauf t von $\mathcal{A}$ auf w. Seien $N_0, N_1, \ldots$ die jeweiligen Level von t. Laut Lemma 23 gilt $rang(N_i) < \infty$ für alle $i \in \mathbb{N}$. Falls t nicht gedächtnislos ist, dann gibt es ein kleinstes $i \in \mathbb{N}$, und zwei verschiedene Knoten $n_1, n_2 \in N_i$, deren Unterbäume verschieden sind, für die aber $t(n_1) = t(n_2)$ gilt. O.B.d.A. gelte $rang(n_1) \leq rang(n_2)$. Man sieht leicht, dass das Ersetzen des Unterbaums von n_1 an die Stelle von n_2 nicht die Tatsache zerstört, dass t akzeptierend ist. Dies wird von Lemma 23 gezeigt, denn im entstehenden Lauf haben alle Levels erst recht endlichen Rang.

Sei $t_0 := t$ und t_1 derjenige Lauf, der dadurch entsteht, dass diese Ersetzung auf t_0 solange durchgeführt wird, bis es auf dem i-ten Level keine Knoten mehr gibt, die die Gedächtnislosigkeit verletzen. Da jeder Level eines Lauf nur endlich viele Knoten hat, muss so ein t' existieren. Dieses Verfahren lässt sich nun ins Unendliche iterieren, wobei Läufe $t_0, t_1, \ldots$ erzeugt werden. Sei t^* der Limit dieser Konstruktion, d.h. der eindeutige Baum, der gemeinsame Präfixe mit allen t_i hat. Beachte, dass sogar gilt: Für jedes $level_i$ in t^* existiert ein $k \in \mathbb{N}$ so dass $level_i$ auch das i-te Level von allen t_j mit $j \geq k$ ist.

Wir behaupten, dass t^* ein akzeptierender und gedächtnisloser Lauf von $\mathcal{A}$ auf w ist. Dazu betrachten wir einen beliebigen Level $level_j$ aus t^*. Da $level_j$ in fast allen t_i ebenso vorkommt, kann es keine zwei Knoten enthalten, die die Gedächtnislosigkeit verletzen. Außerdem ist Rang von $level_j$ auch endlich, denn mit steigendem i kann dieser in den jeweiligen t_i nur abnehmen. Da dies für alle Level gilt, ist t^* insgesamt gedächtnislos. Laut Lemma 23 ist t^* aber auch ein akzeptierender Lauf. ∎

Satz 42 (Miyano-Hayashi)
Sei $\mathcal{A}$ ein ABA mit n Zuständen. Dann gibt es einen NBA $\mathcal{A}'$ mit höchstens 2^{2n} vielen Zuständen, so dass $L(\mathcal{A}') = L(\mathcal{A})$ gilt.

BEWEIS Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$. Wir konstruieren einen NBA $\mathcal{A}'$, der intuitiv zu jeden Schritt eine mögliche Ebene eines gedächtnislosen Laufs von $\mathcal{A}$ auf einem Eingabewort rät. Um zu überprüfen, ob dieser Lauf akzeptierend ist, sprich ob es keinen Pfad darin gibt, auf dem nur endlich viele Endzustände vorkommen, merkt sich $\mathcal{A}'$ in jedem Schritt außerdem noch diejenigen Zustände im aktuellen Level, von denen man aus noch einen Endzustand durchlaufen muss. Ist diese einmal leer, dann müssen wiederum alle Zustände des aktuellen Levels so verfolgt werden.

O.B.d.A. nehmen wir $q_0 \notin F$ an. Sei $\mathcal{A}' := (2^Q \times 2^Q, \Sigma, (\{q_0\}, \{q_0\}), \delta', F')$, wobei

$$\begin{aligned}
\delta'((P, \emptyset), a) &:= \{(P', P' \setminus F) \mid P' \models_{min} \bigwedge_{q \in P} \delta(p, a)\} \\
\delta'((P, O), a) &:= \{(P', O' \setminus F) \mid P' \models_{min} \bigwedge_{q \in P} \delta(p, a) \text{ und} \\
&\qquad\qquad O' \subseteq P' \text{ und } O' \models_{min} \bigwedge_{q \in O} \delta(q, a)\}
\end{aligned}$$

und $F' := 2^Q \times \{\emptyset\}$.

Die Behauptung über die Größenbeschränkung von $\mathcal{A}'$ ist leicht ersichtlich. Es bleibt noch die Korrektheit der Konstruktion, sprich $L(\mathcal{A}') = L(\mathcal{A})$ zu zeigen.

"$\supseteq$" Sei $w = a_0 a_1 \ldots \in L(\mathcal{A})$, d.h. es gibt einen akzeptierenden Lauf von $\mathcal{A}$ auf w. Laut Lemma 24 gibt es dann auch einen gedächtnislosen und akzeptierenden Lauf. Dieser lässt sich als DAG darstellen, so dass jeder Level dieses Lauf eine Teilmenge $P \subseteq Q$ bildet. Seien $P_0, P_1, \ldots$ diese Ebenen. Offensichtlich gilt $P_0 = \{q_0\}$. Außerdem gilt wegen Lemma 21 für alle $i \in \mathbb{N}$:

$$P_{i+1} \models \bigwedge_{q \in P_i} \delta(q, a_i)$$

Wir definieren nun Mengen O_i für $i \in \mathbb{N}$ induktiv wie folgt. $O_0 := \{q_0\}$ und, für $i \geq 1$:

$$O_i \;:=\; \begin{cases} P_i \;, & \text{falls } O_{i-1} = \emptyset \\ \bigcup\limits_{O' \subseteq P_{i-1}} \{O' \setminus F \mid O' \models \bigwedge\limits_{q \in O_{i-1}} \delta(q, a)\}, & \text{sonst} \end{cases}$$

Man sieht leicht mithilfe von Lemma 21, dass $(P_0, O_0), (P_1, O_1), \ldots$ ein Lauf von $\mathcal{A}'$ auf w ist. Es bleibt lediglich zu zeigen, dass dieser auch akzeptierend ist. Dazu beobachten wir zunächst, dass für alle $i \in \mathbb{N}$ gilt: wenn $O_i \neq \emptyset$, dann ist $\mathit{rang}(O_{i+1}) < \mathit{rang}(O_i)$, wobei wir die O_i als Level in dem Lauf auf w ansehen. Außerdem gilt $\mathit{rang}(O_i)$ nur, wenn $O_i = \emptyset$. Da der Rang eines jeden O_i aber endlich sein muss, denn $|O_i| < \infty$, und laut Lemma 23 hat jeder Knoten in O_i nur endlichen Rang, folgt daraus sofort, dass es unendlich viele i geben muss, so dass $O_i = \emptyset$ ist. Damit ist der Lauf aber akzeptierend.

"$\subseteq$" Angenommen $(P_0, O_0), (P_1, O_1), \ldots$ ist ein akzeptierender Lauf von $\mathcal{A}'$ auf $w = a_0 a_1 \ldots \in \Sigma^\omega$. Man sieht leicht, dass $P_0, P_1, \ldots$ die Level eines gedächtnislosen Laufs von $\mathcal{A}$ auf w repräsentieren. Es bleibt wieder zu zeigen, dass dieser akzeptierend ist. Nach Voraussetzung gibt es $i_0 < i_1 < \ldots$, so dass $O_{i_j} = \emptyset$ für alle $j \in \mathbb{N}$. Dann gilt aber $\mathit{rang}(P_{i_j+1}) < \infty$ für alle $j \in \mathbb{N}$, denn $O_{i_j+1} = P_{i_j+1}$ für alle $j \in \mathbb{N}$. Da aber auf jedes solche i_j wieder ein i_{j+1} folgt mit $O_{i_{j+1}} = \emptyset$ folgt, gibt es keinen Pfad, der von einem Knoten im Level $q \in P_{i_j+1}$ ausgeht und niemals einen Endzustand besucht.

Somit gibt es also unendlich viele Level mit endlichem Rang. Laut Lemma 23 ist der Lauf dann aber akzeptierend. ∎

3.11 Komplementierung von NBAs mittels Alternierung

Definition 32
Ein *alternierender co-Büchi-Automat* (AcoBA) ist ein Tupel $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$, welches definiert ist wie bei einem ABA. Ebenso ist der Lauf eines AcoBA auf einem Wort $w \in \Sigma^\omega$ definiert. Solch ein Lauf ist im Gegensatz zu dem Lauf eines ABA jedoch akzeptierend, wenn auf allen Pfaden nur endlich viele Zustände $q \notin F$ vorkommen.

Beachte: Die co-Büchi-Bedingung ist das Duale zur Büchi-Bedingung.

Lemma 25
Für jeden ABA $\mathcal{A}$ mit n Zuständen gibt es einen AcoBA $\overline{\mathcal{A}}$ mit höchstens n Zuständen, so dass $L(\overline{\mathcal{A}}) = \Sigma^\omega \setminus L(\mathcal{A})$ gilt. Die Umkehrung gilt ebenso.

Beweis Übung. ■

Korollar 43
Zu jedem NBA $\mathcal{A}$ mit n Zuständen gibt es einen AcoBA $\overline{\mathcal{A}}$ mit höchstens n Zuständen, so dass $L(\overline{\mathcal{A}}) = \Sigma^\omega \setminus L(\mathcal{A})$ gilt.

Wie bei ABAs kann man bei AcoBAs sich ebenfalls auf gedächtnislose Läufe, also insbesondere solche, die sich als DAG repräsentieren lassen, beschränken. Dies werden wir aber dieses Mal nur hinnehmen, ohne es explizit zu beweisen.

Lemma 26
Sei $\mathcal{A}$ ein AcoBA und $w \in \Sigma^\omega$. Es gilt: Wenn $w \in L(\mathcal{A})$, dann gibt es einen gedächtnislosen Lauf von $\mathcal{A}$ auf w.

Beweis Übung. ■

Im folgenden ordnen wir wieder einem Knoten in einem (gedächtnislosen) Lauf eines AcoBA einen Rang zu. Dies ist jedoch nicht dieselbe Definition des Ranges aus dem vorigen Abschnitt.

Definition 33
Sei t ein gedächtnisloser Lauf eines AcoBA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ auf einem Wort $w \in \Sigma^\omega$ – repräsentiert als DAG, dessen Level jeweils höchstens $|Q|$ viele Elemente haben. Zuerst definieren wir induktiv Mengen $R_0^t \supseteq R_1^t \supseteq R_2^t \supseteq \ldots$, von Knoten in t. R_0^t ist die Menge aller Knoten in t, und für alle $i \in \mathbb{N}$:

$$\begin{aligned} R_{2i+1}^t &:= R_{2i}^t \setminus \{n \mid \text{von } n \text{ aus sind nur endlich viele } R_{2i}^t - \text{Knoten erreichbar } \} \\ R_{2i+2}^t &:= R_{2i+1}^t \setminus \{n \mid \text{für alle von } n \text{ aus erreichbaren Knoten } n' \text{ gilt} \\ &\qquad\qquad n' \in R_{2i+1}^t \Rightarrow t(n') \in F\} \end{aligned}$$

Der Rang eines Knoten n, $rang(n)$ ist das minimale $i \in \mathbb{N}$, für das gilt: $n \in R_i^t \setminus R_{i+1}^t$.

Da alternierende Automaten so definiert wurden, dass jede Transition mindestens einen Nachfolgezustand enthält, hat jeder Knoten in einem Lauf mindestens einen Nachfolger. Daher ist der Rang eines Knotens immer mindestens 1. Interessanter ist jedoch eine Abschätzung nach oben. Es gilt, dass jeder Knoten in einem akzeptierenden Lauf endlichen Rang hat. Es gilt sogar die folgende, wesentlich stärkere Eigenschaft.

Lemma 27
Sei t ein gedächtnisloser Lauf eines AcoBA $\mathcal{A}$ mit n Zuständen auf einem beliebigen Wort $w \in \Sigma^\omega$. Dann gilt: $R_{2n+1}^t = \emptyset$.

Beweis Man zeigt leicht durch Induktion über i, dass gilt: Für alle $i \leq n$ existiert ein $l_i \in \mathbb{N}$, so dass für alle $j \geq l_i$ gilt: der j-te Level von t hat höchstens $n - i$ Knoten in R_{2i}^t. Der Induktionsanfang ist klar, da in der DAG-Repräsentation eines gedächtnislosen Laufes jeder Level höchstens n Knoten hat. Der Induktionsschritt wird durch Fallunterscheidung gezeigt. Es sollte klar sein, dass die Behauptung des Lemmas aus dieser Aussage folgt. ■

Daraus folgt offensichtlich, dass der Rang eines Knoten in einem akzeptierenden und gedächtnislosen Lauf eines AcoBA auf einem beliebigen Wort durch $2n$ beschränkt ist, wobei n die Anzahl der Zustände des Automaten ist.

Lemma 28
Sei t ein gedächtnisloser und akzeptierender Lauf eines AcoBA $\mathcal{A}$ auf einem beliebigen Wort $w \in \Sigma^\omega$. Dann gibt es auf jedem Pfad unendlich viele Zustände mit ungeradem Rang.

Beweis Übung. ∎

Satz 44 (Kupferman-Vardi)
Für jeden AcoBA $\mathcal{A}$ mit n Zuständen gibt es einen ABA $\mathcal{A}'$ mit höchstens $2n^2+1$ vielen Zuständen, so dass $L(\mathcal{A}') = L(\mathcal{A})$ gilt.

Beweis Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein AcoBA. Definiere $\mathcal{A}' := (Q', \Sigma, q_0', \delta', F')$ mit $Q' := \{\bot\} \cup Q \times \{1, \ldots, 2n\}$, $q_0' := (q_0, 2n)$, $F' := \{(q,i) \mid i \text{ ist ungerade }\}$ und

$$\begin{aligned} \delta(\bot, a) &:= \bot \\ \delta((q,i), a) &:= \begin{cases} \bot & \text{, falls } q \notin F \text{ und } i \text{ ist ungerade} \\ \delta(q,a)|_i & \text{, sonst} \end{cases} \end{aligned}$$

wobei $(\varphi \vee \psi)|_i := \varphi|_i \vee \psi|_i$, $(\varphi \wedge \psi)|_i := \varphi|_i \wedge \psi|_i$ und $q|_i := \bigvee_{j \leq i}(q,j)$.

Intuitive errät $\mathcal{A}'$ zu jedem Knoten in einem gedächtnislosen Lauf von $\mathcal{A}$ auf einem w dessen Rang und speichert diesen in der zweiten Komponente des Zustands ab. Es bleibt noch $L(\mathcal{A}') = L(\mathcal{A})$ zu zeigen.

"$\supseteq$" Man sieht leicht, dass man einen Lauf von $\mathcal{A}'$ auf einem $w \in \Sigma^\omega$ dadurch erhält, indem man in einem akzeptierenden Lauf von $\mathcal{A}$ auf diesem w jeden Knoten mit seinem Rang annotiert. Dass dies möglich ist, zeigt Lemma 27. Laut Lemma 28 kommen auf jedem Pfad unendlich viele Zustände mit ungeradem Rang vor. Damit ist der Lauf des ABA aber akzeptierend.

"$\subseteq$" Angenommen $\mathcal{A}'$ hat einen akzeptierenden Lauf auf einem $w \in \Sigma^\omega$. In diesem kann nirgendwo der Zustand $\bot$ vorkommen. Klar ist, dass die Projektion auf die jeweils erste Zustandskomponente einen Lauf von $\mathcal{A}'$ auf w liefert.

Außerdem beobachten wir, dass auf jedem Pfad des Laufs von $\mathcal{A}'$ die zweiten Zustandskomponenten nur absteigen können. Da es nur endlich viele gibt, gilt für jeden Pfad, dass er irgendwann nur noch Zustände in $Q \times \{i\}$ für ein $i \in \{1, \ldots, 2n\}$ besucht. Da der Lauf aber akzeptierend ist, muss i ungerade sein. Aufgrund der Definition von δ' kann also irgendwann auf diesem Pfad kein Zustand aus $Q \setminus F$ mehr vorkommen. Somit ist auf allen Pfaden die co-Büchi-Bedingung erfüllt und es gilt $w \in L(\mathcal{A})$. ∎

Korollar 45
Für jeden NBA $\mathcal{A}$ mit n Zuständen existiert ein NBA $\overline{\mathcal{A}}$ mit höchstens 2^{4n^2+2} vielen Zuständen, so dass $L(\overline{\mathcal{A}}) = \Sigma^\omega \setminus L(\mathcal{A})$ ist.

Beweis Sei $\mathcal{A}$ ein NBA mit n Zuständen. Dieser ist insbesondere ein ABA. Laut Lemma 25 existiert dann ein AcoBA $\overline{\mathcal{A}}$ mit n Zuständen, der dessen Komplement erkennt. Mit Satz 44 lässt dieser sich in einen ABA $\overline{\mathcal{A}}'$ mit $2n^2 + 1$ vielen Zuständen umwandeln. Schließlich liefert Satz 42 dafür einen NBA mit höchstens der geforderten Zustandszahl. ■

Korollar 45 liefert zwar eine asymptotisch schlechtere Komplementierungskonstruktion für NBAs als die von Safra aus Proposition 21. Diese hat jedoch den Vorteil, dass sie symbolisch implementiert werden kann.

3.12 Schwache Automaten

Definition 34
Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein Automat (nicht-det. / alternierend, Büchi / co-Büchi). Im Falle der Alternierung schreiben wir auch $q' \in \delta(q, a)$, falls q' syntaktisch in der positiv booleschen Formel $\delta(q, a)$ vorkommt.

Solch ein Automat heißt *schwach*, wenn es eine Partition $Q_1 \cup \ldots \cup Q_k$ seiner Zustandsmenge gibt, so dass gilt:

- Für alle $i = 1, \ldots, k$ ist $Q_i \subseteq F$ oder $Q_i \cap F = \emptyset$.
- Für alle $q, q' \in Q$ mit $q' \in \delta(q, a)$ für ein $a \in \Sigma$ gilt: Wenn $q \in Q_i$ und $q' \in Q_j$, dann ist $j \leq i$.

Mit WABA / WAcoBA / WNBA bezeichnen wir einen schwachen ABA / AcoBA / NBA.

Man beachte, dass die in Satz 44 konstrukierten ABAs in Wirklichkeit sogar WABAs sind. Somit existiert zu jedem AcoBA auch ein äquivalenter WABA. Es stellt sich die Frage, ob dies auch für ABAs und WAcoBAs gilt. Dies folgt aus den beiden folgenden Lemmas.

Lemma 29
Für jeden WABA mit n Zuständen existiert ein äquivalenter WAcoBA mit n Zuständen und umgekehrt.

Beweis Dies folgt trivialerweise aus der Tatsache, dass ein schwacher Automat ein Wort mit Büchi-Bedingung akzeptiert gdw. wenn er es mit der co-Büchi-Bedingung akzeptiert. Beachte, dass jeder Pfad in einem Lauf eines schwachen Automaten sich in einer starken Zusammenhangskomponente des Automaten verfängt. Darin sind entweder alle Zustände Endzustände oder keiner ist es. Somit kommen auf solch einem Pfad unendlich viele Endzustände vor gdw. nur endlich viele Nicht-Endzustände vorkommen. ■

Lemma 30
Für jeden WABA $\mathcal{A}$ mit n Zuständen existiert ein äquivalenter WABA $\overline{\mathcal{A}}$ mit höchstens n Zuständen, so dass $L(\overline{\mathcal{A}}) = \Sigma^\omega \setminus L(\mathcal{A})$ gilt.

Beweis Folgt sofort aus Lemmas 25 und 29. ■

Daraus folgt, dass schwache alternierende Büchi-Automaten gar nicht schwächer sind als normale alternierende Büchi-Automaten.

Satz 46
Für jeden ABA $\mathcal{A}$ mit n Zuständen existiert ein äquivalenter WABA $\mathcal{A}'$ mit höchstens $2n^2 + 1$ vielen Zuständen.

Dies steht im Gegensatz zu der Situation bei nicht-deterministischen Automaten.

Satz 47
Es gibt ω-reguläre Sprachen, die nicht von einem WNBA erkannt werden.

Ein Beispiel einer solchen Sprache ist $\{w \in \{a, b\}^\omega \mid |w|_a = \infty\}$.

Definition 35
Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein alternierender Büchi-Automat mit $Q = \{q_0, \ldots, q_n\}$. Dieser heißt *schleifenfrei* (oder auch *zählerfrei*, bzw. *sehr schwach*), wenn es eine lineare Ordnung $<$ auf seiner Zustandsmenge gibt, so dass für alle $q, q' \in Q$ gilt: wenn $q' \in \delta(q, a)$ für ein $a \in \Sigma$, dann gilt $q' \leq q$.
Wir bezeichnen einen solchen Automaten als VWABA.

Beachte, dass ein VWABA auch ein WABA ist. Die geforderte Partition erhält man, indem man jeden Zustand als eine Komponente betrachtet.

Im Gegensatz zu WABA erkennen VWABA weniger als die ω-regulären Sprachen, nämlich genau die stern-freien. Wir zeigen hier lediglich, dass VWABAs gleich mächtig zu LTL sind.

Satz 48
Für jede LTL-Formel φ existiert ein VWABA $\mathcal{A}_\varphi$ mit $L(\mathcal{A}_\varphi) = L(\varphi)$ und $|\mathcal{A}_\varphi| = O(|\varphi|)$.

Beweis Übung. ∎

Satz 49
Für jeden VWABA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ über einem Alphabet Σ existiert eine LTL-Formel $\varphi_\mathcal{A}$ mit $L(\varphi_\mathcal{A}) = L(\mathcal{A})$ und $|\varphi_\mathcal{A}| = O(|Q| \cdot 2^{|\Sigma| \cdot m})$, wobei $m := \max\{|\delta(q, a)| \mid q \in Q, a \in \Sigma\}$.

Beweis Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein VWABA mit $Q = \{q_0, \ldots, q_n\}$. O.B.d.A. nehmen wir an, dass $q_0 > q_1 > \ldots > q_n$ gilt, d.h. jeder Zustand ist vom Anfangszustand aus erreichbar.

Wir definieren nun für $i = n, \ldots, 0$ Formeln ψ_i, die genau diejenigen Sprachen definieren, die von $\mathcal{A}$ in Zustand q_i erkannt werden. Bei der Definition von ψ_i können wir davon ausgehen, dass ψ_j für alle $j > i$ bereits definiert ist. Die Konstruktion von ψ_i unterscheidet zwei Fälle.

1. Sei $q_i \notin F$. Die von Zustand q_i aus erkannte Sprache ist dieselbe wie die von X, wobei X rekursiv definiert ist über

$$X \;=\; \bigvee_{a \in \Sigma} a \wedge \bigcirc(\delta(q_i, a))'$$

wobei $(\varphi_1 \vee \varphi_2)' := \varphi_1' \vee \varphi_2'$, $(\varphi_1 \wedge \varphi_2)' := \varphi_1' \wedge \varphi_2'$, $q_i' := X$ und $q_j' := \psi_j$ für $j > i$. Beachte, dass es sich dabei um den kleinsten Fixpunkt handelt, denn $q_i \notin F$. Mithilfe der LTL-Äquivalenzen $\bigcirc(\varphi_1 \wedge \varphi_2) \equiv \bigcirc\varphi_2 \wedge \bigcirc\varphi_2$, $\bigcirc(\varphi_1 \vee \varphi_2) \equiv \bigcirc\varphi_2 \vee \bigcirc\varphi_2$ und der deMorgan'schen Regeln lässt sich die rechte Seite der Gleichung in disjunktiver Normalform bringen, so dass $\bigcirc$-Operatoren nur direkt vor atomaren Formeln vorkommen. Die rechte Seite ist also äquivalent zu einem $\alpha \vee (\beta \wedge \bigcirc X)$, so dass X nicht in α oder β vorkommt. Damit gilt dann $X \equiv \beta \mathtt{U} \alpha =: \psi_i$.

2. Sei $q_i \in F$. Ebenso lässt sich hier eine Gleichung $X \equiv \alpha \wedge (\beta \vee \bigcirc X)$ durch Umformen in konjunktive Normalform finden, die genau die von q_i aus erkannte Sprache beschreibt. Beachte, dass es sich hierbei um den größten Fixpunkt dieser Gleichung handelt. Dann gilt aber auch $X \equiv \mathtt{G}\alpha \vee (\alpha \mathtt{U} (\beta \wedge \alpha)) =: \psi_i$.

Die Größenbeschränkung ergibt sich aus der Tatsache, dass es insgesamt n viele Formeln der Form ψ_i gibt und jede aus einer Formel der Größe $O(\Sigma \cdot m)$ durch Umwandeln in dis-/konjunktive Normalform entsteht. Die Korrektheit der Konstruktion zeigt man leicht durch Induktion über die Anzahl der Zustände von $\mathcal{A}$. ∎