Automatentheorie

Mitschrift einer Vorlesung
von
Universitätsprofessor Dr.–Ing. Franz Baader
an der
Rheinisch–Westfälischen Technischen Hochschule Aachen
im WS94/95

10. Dezember 2001

Dies ist die Mitschrift einer Vorlesung; sie sollte nicht mit einem Lehrbuch verwechselt werden. Zum vollständigen Verstehen des Inhalts ist der Besuch der Vorlesung und das Lösen der Übungsaufgaben nötig. Ohne die mündlichen Erläuterungen der Vorlesung bleiben viele Zusammenhänge unerwähnt und die meisten Beweise lückenhaft. Ohne das selbständige Lösen der Übungsaufgaben ist ein tieferes Verständnis nicht möglich.

Obwohl dieses Skriptum gründlich Korrektur gelesen wurde, kann keine Fehlerfreiheit garantiert werden. Hinweise auf Fehler, dunkle Formulierungen und unverständliche Bezeichnungen werden deshalb gerne entgegengenommen, am besten persönlich vorbeikommen oder per e-mail. Eine aktuelle Liste der bisher entdeckten Fehler gibt es unter
`http://www-lti.informatik.rwth-aachen.de/Lehre/AT-errata-96.html`.

Ulrike Sattler, Jörn Richts 10. Dezember 2001

`script@cantor.informatik.rwth-aachen.de`

Inhaltsverzeichnis

Motivation und Einordnung **1**

1 Reguläre Sprachen, endliche Monoide und logische Formeln **6**

1.1 Reguläre Sprachen und endliche Automaten 6

1.2 Reguläre Sprachen und endliche Monoide 11

1.3 Reguläre Sprachen und logische Formeln 26

2 Verallgemeinert–definite Sprachen und quantorenfreie Formeln **30**

2.1 Verallgemeinert–definite Sprachen 30

2.2 Die zugehörige S–Varietät 32

2.3 Quantorenfreie Formeln 37

3 Sternfreie Sprachen **42**

3.1 Die Sprachklassen 42

3.2 Aperiodische Monoide 44

3.3 Formeln der Logik erster Stufe 46

4 Unendliche Wörter und Büchi–Automaten **62**

4.1 Büchi–Automaten und ω–reguläre Sprachen 64

4.2 Abschlußunter Komplement 74

4.3 Muller–Automaten 80

5 Unendliche Wörter und logische Formeln **83**

5.1 Die Logik S1S und ω–reguläre Sprachen 83

5.2 Schwächere Logiken und Presburger Arithmetik 94

5.3 Sternfreie Sprachen und unendliche Wörter 97

5.4 Dynamische Logik 99

6 Automaten auf endlichen Bäumen **113**

6.1 Endliche Bäume 113

6.2 Reguläre Baumsprachen 121

7 Automaten auf unendlichen Bäumen **132**

7.1 Büchi– und Rabin–Baumautomaten 133

7.2 Entscheidbarkeitsresultate 139

8 Baumautomaten und logische Formeln **149**

8.1 Die Logik S2S und ihre Varianten 149

8.2 Dynamische Logiken 153

Schlußwort **159**

Literaturverzeichnis **160**

Notationstabelle **162**

Index **166**

Motivation und Einordnung

Im Bereich Automatentheorie und formale Sprachen interessiert man sich für Klassen von Sprachen und deren Eigenschaften. Eine *formale Sprache* (oder kurz Sprache) ist eine Menge $L \subseteq \Sigma^*$, d.h. eine Menge von Wörtern über einem Alphabet Σ. Dabei ist Σ meist endlich. Eine *Klasse formaler Sprachen* $\mathcal{K}$ ordnet jedem endlichen Alphabet Σ eine Menge $\mathcal{K}_\Sigma \subseteq 2^{\Sigma^*}$ zu, d.h. eine Menge von Sprachen über Σ.

Für eine gegebene Klasse $\mathcal{K}$ interessiert man sich für die folgenden drei Fragestellungen:

Charakterisierungen Wie kann man die zur Klasse gehörenden Sprachen charakterisieren?

Gesucht sind also Eigenschaften E, die erfüllen:

$$L \in \mathcal{K}_\Sigma \quad \text{gdw} \quad L \subseteq \Sigma^* \text{ und } L \text{ hat Eigenschaft } E.$$

Man will dabei meist verschiedene äquivalente Charakterisierungen (Automaten, Grammatiken, ...) finden, da einige Charakterisierungen für gewisse Zwecke besser geeignet sind als andere.

Abschlußeigenschaften Unter welchen Operationen auf Sprachen (Durchschnitt, Vereinigung, Komplement, homomorphe Bilder, ...) ist die Klasse abgeschlossen?

Kenntnisse der Abschlußeigenschaften einer Klasse $\mathcal{K}$ erleichtern oft die Entscheidung von Fragen wie "$L \in \mathcal{K}_\Sigma$?" oder "Sind die Eigenschaften E_1 und E_2 äquivalent?"

Entscheidbarkeit Welche Fragestellungen sind für gegebene Sprachen der Klasse entscheidbar?

Z.B.: $w \in L$? $L \neq \emptyset$? $L_1 \subseteq L_2$?

Dabei geht man davon aus, daß die (im allgemeinen unendlichen) Sprachen durch eine der oben angesprochenen Charakterisierungen endlich beschrieben sind.

Die wichtigsten Sprachklassen sind in der Chomsky–Hierarchie zusammengefaßt:

Klasse	Charakterisierung
Typ 0	• allgemeine Chomsky–Grammatiken (Regeln $u \to v$, u enthält mindestens ein Nicht–Terminal) • akzeptiert von Turingmaschinen
Typ 1 kontext-sensitiv	• kontextsensitive Grammatiken (Regeln $u \to v$, $1 \leq \|u\| \leq \|v\|$) • akzeptiert von Turingmaschinen mit linearer Bandbeschränkung
Typ 2 kontext-frei	• kontextfreie Grammatiken (Regeln $X \to v$, X Nicht–Terminal) • akzeptierte von Kellerautomaten
Typ 3 regulär	• rechtslineare Grammatiken (Regeln $X \to uY$, $X \to u$, u Terminalwort) • akzeptiert von endlichen Automaten

Beispiel für Charakterisierung

Liefern deterministische Automaten (Maschinen) bereits die gesamte Sprachklasse?

Typ 0	Ja
Typ 1	Offen
Typ 2	Nein
Typ 3	Ja

Beispiel für Abschlußeigenschaft

Ist die Klasse unter Komplementbildung abgeschlossen, d.h. $L \in \mathcal{K}_\Sigma \rightsquigarrow \Sigma^* \setminus L \in \mathcal{K}_\Sigma$?

Typ 0	Nein, da das Komplement einer rekursiv aufzählbaren Menge im allgemeinen nicht rekursiv aufzählbar ist.
Typ 1	Ja (dies ist ein Resultat von 1987).
Typ 2	Nein (daraus folgt das "Nein" in der vorherigen Frage).
Typ 3	Ja (dies folgt aus dem "Ja" in der vorherigen Frage).

Beispiele für Entscheidungsprobleme

Sind das Elementproblem und das Äquivalenzproblem entscheidbar?

Klasse	$w \in L$?	$L_1 = L_2$?
Typ 0	unentscheidbar	unentscheidbar
Typ 1	entscheidbar	unentscheidbar
Typ 2	entscheidbar	unentscheidbar
Typ 3	entscheidbar	entscheidbar

Dieses Skript wird sich meist auf Typ 3 (reguläre) Sprachen konzentrieren. Dabei werden Unterklassen, andere Charakterisierungen und Verallgemeinerungen auf unendliche Wörter und Bäume betrachtet. Das Skript gliedert sich in drei Teile:

1. Reguläre Sprachen endlicher Wörter

Als alternative Charakterisierungen werden betrachtet:

- Algebraische Charakterisierung

  - Jeder Sprache kann ein Monoid (das syntaktische Monoid) zugeordnet werden.
  - Reguläre Sprachen sind genau die mit endlichem syntaktischen Monoid.

- Logische Charakterisierung
  - Logische Formeln können Sprachen definieren.
  - Es gibt eine Logik (Monadische Logik zweiter Stufe), deren Formeln genau die regulären Sprachen definieren.

Mit Hilfe dieser Charakterisierungen kann man Unterklassen der Klasse der regulären Sprachen beschreiben. Die wohl bekannteste Unterklasse ist die der *sternfreien Sprachen*:

- Diese werden genau durch Formeln der Logik erster Stufe definiert.
- Ihre syntaktischen Monoide sind genau die aperiodischen Monoide.

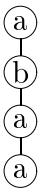
2. Sprachen unendlicher Wörter

Statt endlicher Wörter (endliche Folgen von Elementen des Alphabets) kann man auch unendliche Wörter (unendliche Folgen) als Eingabe für endliche Automaten verwenden. Dazu muß lediglich die Akzeptanzbedingung geändert werden. Während bei endlichen Wörtern nach Lesen des Wortes ein Endzustand erreicht sein muß, werden bei unendlichen Wörtern Bedingungen an die Menge der unendlich oft durchlaufenen Zustände gestellt.

Hier werden Abschlußeigenschaften, Entscheidbarkeit des Leerheitsproblems und der Zusammenhang zur Logik betrachtet. Daraus lassen sich für die Logik wichtige Entscheidbarkeitsresultate ableiten.

3. Baumsprachen

Wörter kann man als beschriftete Bäume mit Verzweigungsgrad 1 auffassen.

$abaa \quad \widehat{=}$ 

Man kann nun den Begriff des endlichen Automaten zu Baumautomaten verallgemeinern, die auch Bäume mit Verzweigungszahl größer als eins als Eingabe zulassen. Viele Resultate für reguläre Sprachen lassen sich auf den Fall von Baumsprachen übertragen. Auch hier ist der Zusammenhang zur Logik sehr interessant. Es werden sowohl endliche als auch unendliche Bäume betrachtet.

Hauptmotivation in allen drei Teilen ist die Anwendung in der Logik, insbesondere die sich aus der Automatentheorie für die Logik ergebenden Entscheidbarkeitsresultate. Dabei sind die Methoden, mit denen man zu diesen Resultaten gelangt, mindestens genauso wichtig wie die Resultate selbst. Es wird daher sehr viel Gewicht auf vollständige Beweise gelegt, auch wenn diese manchmal aufwendig sind.

Kapitel 1

Reguläre Sprachen, endliche Monoide und logische Formeln

Ziel dieses Kapitels ist es, einige Definitionen und Resultate über reguläre Sprachen aus der Vorlesung „Grundzüge der Theoretischen Informatik“ zu wiederholen und anschließend den Zusammenhang zu Monoiden und Formeln herzustellen.

1.1 Reguläre Sprachen und endliche Automaten

Die Notation ist hier weitgehend an das GTI–Skriptum [Tho90b] angelehnt.

Definition 1.1 Für ein endliches Alphabet Σ ist die Klasse der *regulären Sprachen* $\mathcal{R}eg_\Sigma$ über Σ die kleinste Klasse mit

- $\emptyset$, $\{\epsilon\}$ und $\{a\}$ für $a \in \Sigma$ sind in Reg_Σ (wobei ϵ das leere Wort ist),
- sind $L, L_1, L_2 \in Reg_\Sigma$, so auch $L_1 \cup L_2$, $L_1 \cdot L_2 = \{u \cdot v \mid u \in L_1 \text{ und } v \in L_2\}$, $L^* = \{u_1 \cdots u_n \mid n \geq 0 \text{ und } u_i \in L\}$.

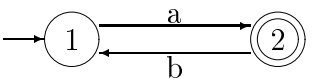
Wie üblich beschreiben wir reguläre Sprachen durch *reguläre Ausdrücke*, bei denen die Mengenklammern (und häufig auch der Konkatenationspunkt) wegfallen. Z.B.: $(ab)^*a$ beschreibt $(\{a\} \cdot \{b\})^* \cdot \{a\}$, d.h. die Sprache aller Wörter, die mit a beginnen und enden und bei denen sich a und b abwechseln.

Aus der Definition ergibt sich offenbar, daß reguläre Sprachen unter Vereinigung, Konkatenation und Kleene–Stern abgeschlossen sind. Um Abschluß unter Durchschnitt und Komplement zu zeigen, ist die Charakterisierung durch endliche Automaten besser geeignet.

Definition 1.2 Ein *(nicht–deterministischer) endlicher Automat* $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ besteht aus

- einer endlichen Menge Q von Zuständen,
- einem endlichen Alphabet Σ,
- einer Menge von Anfangszuständen $I \subseteq Q$,
- einer Übergangsrelation $\Delta \subseteq Q \times \Sigma \times Q$,
- einer Menge von Endzuständen $F \subseteq Q$.

Wie üblich kann man derartige Automaten graphisch darstellen.

Z.B.: 

$Q = \{1, 2\}$, $\Sigma = \{a, b\}$, $I = \{1\}$ (ausgedrückt durch

)

$\Delta = \{(1, a, 2); (2, b, 1)\}$, $F = \{2\}$ (ausgedrückt durch

)

Ein *Pfad* in dem Automaten ist eine Folge $q_0 a_1 q_1 a_2 \ldots a_n q_n$, so daß für $1 \leq i \leq n$ gilt: $(q_{i-1}, a_i, q_i) \in \Delta$. Abkürzend schreiben wir dafür $q_0 \xrightarrow{a_1 \ldots a_n}_{\mathcal{A}} q_n$. Der Pfad ist *erfolgreich*, wenn $q_0 \in I$ und $q_n \in F$ gilt.

Die von $\mathcal{A}$ *akzeptierte Sprache* ist

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid q_o \xrightarrow{w}_{\mathcal{A}} q_n \text{ ist erfolgreicher Pfad in } \mathcal{A}\}.$$

Eine Sprache $L \subseteq \Sigma^*$ heißt *erkennbar*, falls es einen endlichen Automaten $\mathcal{A}$ gibt mit $L(\mathcal{A}) = L$.

Der *Satz von Kleene* besagt, daß eine Sprache $L \subseteq \Sigma^*$ genau dann erkennbar ist, wenn sie regulär ist. Als Beispiel verwenden wir diese alternative Charakterisierung regulärer Sprachen, um deren Abschluß unter Durchschnitt zu zeigen.

Satz 1.3 Sind $L_1, L_2 \in Reg_\Sigma$, so auch $L_1 \cap L_2$.

Beweis: Es seien $\mathcal{A}_1 = (Q_1, \Sigma, I_1, \Delta_1, F_1)$ und $\mathcal{A}_2 = (Q_2, \Sigma, I_2, \Delta_2, F_2)$ Automaten mit $L_1 = L(\mathcal{A}_1)$ und $L_2 = L(\mathcal{A}_2)$. Wir definieren $\mathcal{A} := (Q_1 \times Q_2, \Sigma, I_1 \times I_2, \Delta, F_1 \times F_2)$, wobei

$$\Delta := \{((q_1, q_2), a, (q_1', q_2')) \mid (q_1, a, q_1') \in \Delta_1 \text{ und } (q_2, a, q_2') \in \Delta_2\}.$$

Man sieht leicht, daß gilt:

$$(q_1, q_2) \xrightarrow{w}_{\mathcal{A}} (q_1', q_2') \quad \text{gdw} \quad q_1 \xrightarrow{w}_{\mathcal{A}_1} q_1' \text{ und } q_2 \xrightarrow{w}_{\mathcal{A}_2} q_2'.$$

Nach Definition der Anfangs– und Endzustandsmengen in $\mathcal{A}$ ist daher $w \in L(\mathcal{A})$ gdw $w \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. ∎

Um den Abschluß unter Komplement zu zeigen, sind nicht–deterministische Automaten nicht sehr gut geeignet.

Beachte: Es sei $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ein *nicht–deterministischer* endlicher Automat. Dann gilt für $\overline{\mathcal{A}} := (Q, \Sigma, I, \Delta, Q \setminus F)$ im allgemeinen *nicht* $L(\overline{\mathcal{A}}) = \Sigma^* \setminus L(\mathcal{A})$.

Damit eine derartige Konstruktion funktioniert, benötigt man deterministische Automaten.

Definition 1.4 Der Automat $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ heißt *deterministisch*, falls gilt:

- $|I| = 1$, d.h. $I = \{q_0\}$,
- Δ ist funktional, d.h. für $q \in Q$ und $a \in \Sigma$ existiert genau ein $q' \in Q$ mit $(q, a, q') \in \Delta$.[1]

Statt der Übergangsrelation Δ verwendet man gewöhnlich bei deterministischen Automaten die zugehörige (totale) *Übergangsfunktion*

$$\begin{array}{rrcl} \delta: & Q \times \Sigma & \to & Q \\ \delta: & (q, a) & \mapsto & q' \text{ gdw } (q, a, q') \in \Delta. \end{array}$$

Die Funktion δ kann man auf Wörter erweitern durch $\delta(q, w) := q'$, wobei q' der (eindeutig bestimmte) Zustand ist, für den es einen Pfad $q \xrightarrow{w}_{\mathcal{A}} q'$ gibt. Es ist $L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$.

Durch die sogenannte *Potenzmengenkonstruktion* kann man jedem nicht–deterministischen Automaten effektiv einen deterministischen Automaten zuordnen, der dieselbe Sprache akzeptiert:

Es sei $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ nicht–deterministisch. Wir definieren $P(\mathcal{A}) := (2^Q, \Sigma, q_o, \delta', F')$ durch:

- $q_0 := I$,
- $\delta'(P, a) := \{q \in Q \mid \text{Es gibt } p \in P \text{ mit } (p, a, q) \in \Delta\}$,
- $F' := \{P \subseteq Q \mid P \cap F \neq \emptyset\}$.

Man zeigt leicht, daß $L(\mathcal{A}) = L(P(\mathcal{A}))$.

Satz 1.5 Ist $L \in Reg_\Sigma$, so auch $\overline{L} = \Sigma^* \setminus L$.

Beweis: Zu L existiert ein *deterministischer* Automat $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$, für den gilt: $w \in L$ gdw $\delta(q_0, w) \in F$. Offenbar ist damit $w \in \overline{L}$ gdw $\delta(q_0, w) \in Q \setminus F$. Daher ist $\overline{\mathcal{A}} = (Q, \Sigma, q_0, \delta, Q \setminus F)$ ein Automat für $\overline{L}$. ∎

[1] Die deterministischen Automaten sind in diesem Skriptum also auch immer vollständig. In nicht–vollständigen deterministischen Automaten gibt es für alle $q \in Q, a \in \Sigma$ höchstens ein $q' \in Q$ mit $(q, a, q') \in \Delta$.

Zu jeder regulären Sprache L existiert ein (bis auf Zustandsumbenennung) eindeutiger deterministischer Automat $\mathcal{A}$ minimaler Zustandsanzahl mit $L = L(\mathcal{A})$. Dieser *minimale Automat* kann wie folgt effektiv konstruiert werden. Es sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein deterministischer Automat für L.

1. Entferne unerreichbare Zustände, d.h. Zustände q, für die es kein $w \in \Sigma^*$ gibt mit $\delta(q_o, w) = q$.

2. Identifiziere äquivalente Zustände.

   Für $q \in Q$ sei $L_q(\mathcal{A}) := \{w \in \Sigma^* \mid \delta(q, w) \in F\}$. Man definiert nun

   $$q \sim_{\mathcal{A}} q' \text{ gdw } L_q(\mathcal{A}) = L_{q'}(\mathcal{A}).$$

   Die Relation $\sim_{\mathcal{A}}$ ist eine Äquivalenzrelation.[2] Faßt man äquivalente Zustände zu einem Zustand zusammen, so erhält man den minimalen Automaten $\mathcal{A}_L$.

Alternativ kann man den minimalen Automaten mit Hilfe der *Nerode–Rechtskongruenz* beschreiben. Für eine Sprache $L \subseteq \Sigma^*$ definieren wir

$$u \rho_L v \text{ gdw für alle } w \in \Sigma^* \text{ gilt } (uw \in L \text{ gdw } vw \in L).$$

Diese Relation ist eine Äquivalenzrelation und es gilt:

$$u \rho_L v \rightsquigarrow uw \rho_L vw \qquad \text{(Rechtskongruenz)}.$$

Der *Satz von Nerode* sagt aus, daß L genau dann regulär ist, wenn ρ_L endlichen Index hat (d.h. nur endlich viele Äquivalenzklassen besitzt). Diese Klassen können nun als Zustände eines Automaten aufgefaßt werden. Für $u \in \Sigma^*$ bezeichne $[u] = \{v \mid u \rho_L v\}$ die ρ_L–Klasse von u. Wir definieren $\mathcal{A}_L = (Q, \Sigma, q_0, \delta, F)$ mit

- $Q := \{[u] \mid u \in \Sigma^*\}$,
- $q_0 := [\epsilon]$,
- $\delta([u], a) := [ua]$,
- $F := \{[u] \mid u \in L\}$.

Für eine reguläre Sprache L ist $\mathcal{A}_L$ der minimale Automat für L.

[2] In Beispiel 1.14 wird gezeigt, wie diese Relation berechnet werden kann.

1.2 Reguläre Sprachen und endliche Monoide

Ein Monoid $(M, \bullet_M, 1_M)$ besteht aus einer nicht–leeren Trägermenge M, einer assoziativen, binären Operation $\bullet_M$ und einem Einselement $1_M \in M$, d.h. es gilt:

für alle $x, y, z \in M$ gilt	$(x \bullet_M y) \bullet_M z = x \bullet_M (y \bullet_M z)$	(Assoziativität),
für alle $x \in M$ gilt	$1_M \bullet_M x = x \bullet_M 1_M = x$	(Einselement).

Wir schreiben meist einfach M für das Monoid $(M, \bullet_M, 1_M)$. Der Index M wird ebenfalls meist weggelassen. Sind M, N Monoide, so heißt eine Abbildung $\phi : M \to N$ *Homomorphismus*, falls gilt:

- $\phi(x \bullet_M y) = \phi(x) \bullet_N \phi(y)$,
- $\phi(1_M) = 1_N$.

Ein Monoid $(M, \bullet_M, 1_M)$ heißt endlich, falls die Trägermenge M endlich ist.

Beispiel 1.6 Für ein Alphabet Σ ist die Menge Σ^* aller Wörter über Σ zusammen mit Konkatenation als Operation $\bullet$ und mit dem leeren Wort ϵ als Einselement ein Monoid.

Man nennt Σ^* *freies Monoid* über Σ, da es folgende universelle Eigenschaft besitzt: Für jedes Monoid M und jede Abbildung $f : \Sigma \to M$ gibt es genau einen Homomorphismus $\phi : \Sigma^* \to M$ mit $\phi|_\Sigma = f$.

Homomorphismen von Σ^* in ein Monoid M können verwendet werden, um Sprachen (d.h. Teilmengen von Σ^*) zu definieren.

Definition 1.7 Es sei M ein Monoid, Σ ein Alphabet und $\phi : \Sigma^* \to M$ ein Homomorphismus. Jede Teilmenge N von M definiert eine Teilmenge $\phi^{-1}(N) := \{w \in \Sigma^* \mid \phi(w) \in N\}$, das Urbild von N unter ϕ. Die Sprache $L \subseteq \Sigma^*$ wird von M *akzeptiert*, wenn es $N \subseteq M$ und $\phi : \Sigma^* \to M$ gibt mit $L = \phi^{-1}(N)$.

Satz 1.8 Für eine Sprache $L \subseteq \Sigma^*$ sind äquivalent:

1. L wird von einem endlichen Monoid akzeptiert.
2. L ist regulär.

Beweis:

"1 $\rightsquigarrow$ 2" Wir fassen hierzu M als endlichen Automaten auf. Es sei also M ein endliches Monoid und $\phi : \Sigma^* \to M$ ein Homomorphismus mit $L = \phi^{-1}(N)$ für $N \subseteq M$. Um zu zeigen, daß L regulär ist, konstruieren wir einen *deterministischen endlichen* Automaten $\mathcal{A}_M$, der L akzeptiert. Wir definieren $\mathcal{A}_M := (M, \Sigma, 1, \delta, N)$ mit der Übergangsfunktion $\delta(m, \sigma) := m \bullet \phi(\sigma)$. Wir zeigen zunächst:

Für alle $w \in \Sigma^*$ und alle $m \in M$ ist $\delta(m, w) = m \bullet \phi(w)$.

Beweis durch Induktion über $|w|$:

$$\begin{array}{lll} w = \epsilon : & \delta(m, \epsilon) = m = m \bullet 1 & \\ & = m \bullet \phi(\epsilon) & \\ w = v\sigma : & \delta(m, v\sigma) = \delta(\delta(m, v), \sigma) = \delta(m \bullet \phi(v), \sigma) & \\ & = (m \bullet \phi(v)) \bullet \phi(\sigma) = m \bullet (\phi(v) \bullet \phi(\sigma)) & \\ & = m \bullet \phi(v\sigma) & \end{array}$$

Damit ergibt sich $\delta(1, w) = 1 \bullet \phi(w) = \phi(w)$. Also gilt

$$\begin{array}{lll} w \in L = \phi^{-1}(N) & \text{gdw} & \phi(w) \in N \\ & \text{gdw} & \delta(1, w) \in N \\ & \text{gdw} & w \in L(\mathcal{A}_M). \end{array}$$

"2 $\rightsquigarrow$ 1": Es sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein deterministischer endlicher Automat mit $L = L(\mathcal{A})$. Offenbar definiert jedes $w \in \Sigma^*$ eine Funktion $\delta_w : Q \to Q : q \mapsto \delta(q, w)$. Es sei $M = Q^Q$ die Menge der Funktionen von Q nach Q. Da Q endlich ist, ist auch M endlich. Mit Komposition von Funktionen als binärer Operation und der Identitätsfunktion *id* als Einselement ist M ein Monoid. Die Komposition ist als $(\delta_1 \circ \delta_2)(q) := \delta_2(\delta_1(q))$ definiert (beachte die Reihenfolge). Man

sieht leicht, daß $\phi : \Sigma^* \to M : w \mapsto \delta_w$ ein Homomorphismus ist: $\phi(wv) = \delta_{wv} = \delta_w \circ \delta_v = \phi(w) \circ \phi(v)$ und $\phi(\epsilon) = id$. Wir definieren $N := \{\delta_w \in Q^Q \mid \delta_w(q_0) \in F\}$. Dann gilt

$$\begin{array}{lll} w \in L(\mathcal{A}) & \text{gdw} & \delta(q_0, w) = \delta_w(q_0) \in F \\ & \text{gdw} & \delta_w \in N \\ & \text{gdw} & \phi(w) \in N \\ & \text{gdw} & w \in \phi^{-1}(N), \end{array}$$

d.h. $L(\mathcal{A})$ wird von $M = Q^Q$ akzeptiert. ∎

Das Bild des im Beweis von "$2 \rightsquigarrow 1$" definierten Homomorphismus ϕ bezeichnen wir als Übergangsmonoid von $\mathcal{A}$:

Definition 1.9 Für einen endlichen deterministischen Automaten $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ist das *Übergangsmonoid* von $\mathcal{A}$ das Untermonoid $M_\mathcal{A} := \{\delta_w \in Q^Q \mid w \in \Sigma^*\}$ von $M = Q^Q$.

Das Übergangsmonoid des minimalen Automaten ist besonders interessant:

Definition 1.10 Für eine reguläre Sprache L heißt das Übergangsmonoid des minimalen Automaten für L *syntaktisches Monoid von* L. Wir bezeichnen dieses mit M_L.

Da der minimale Automat eindeutig durch die Sprache bestimmt ist, hängt M_L also nur von L ab. Es kann auch direkt, d.h. ohne Bezugnahme auf Automaten, von L ausgehend definiert werden.

Definition 1.11 Für eine (beliebige) Sprache $L \subseteq \Sigma^*$ ist die *syntaktische Kongruenz* $\sim_L$ auf Σ^* definiert durch:

$$\begin{array}{l} \text{Für alle } u, v \in \Sigma^* \text{ gilt } u \sim_L v \text{ gdw für alle } x, y \in \Sigma^* \text{ gilt} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad xuy \in L \text{ gdw } xvy \in L. \end{array}$$

Man zeigt leicht, daß $\sim_L$ eine Kongruenzrelation ist, d.h. eine Äquivalenzrelation, die zusätzlich erfüllt:

$$\text{Für alle } u, v, x \in \Sigma^* \text{ gilt } (u \sim_L v \rightsquigarrow (ux \sim_L vx \text{ und } xu \sim_L xv)).$$

Daher kann man das Quotientenmonoid $\Sigma^*/_{\sim_L}$ bilden mit

- Trägermenge $\{[w]_{\sim_L} \mid w \in \Sigma^*\}$, wobei $[w]_{\sim_L} := \{w' \mid w \sim_L w'\}$ ist,
- Operation $[u]_{\sim_L} \bullet [v]_{\sim_L} := [uv]_{\sim_L}$ (Die Repräsentantenunabhängigkeit folgt aus der Kongruenzeigenschaft von $\sim_L$.),
- Einselement $[\epsilon]_{\sim_L}$.

Satz 1.12 Sei $L \subseteq \Sigma^*$ regulär. Dann gilt $M_L \simeq \Sigma^*/_{\sim_L}$, d.h. $\Sigma^*/_{\sim_L}$ ist isomorph zum syntaktischen Monoid M_L.

Beweis: Sei L regulär und $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ der minimale Automat für L. Wir definieren:

$$\begin{array}{rcl} \psi : \Sigma^*/_{\sim_L} & \to & M_L \\ [u]_{\sim_L} & \mapsto & \delta_u \end{array}$$

Im folgenden zeigen wir, daß ψ ein Isomorphismus ist, indem wir zeigen, daß

1. ψ repräsentantenunabhängig (und damit wohldefiniert) ist,
2. ψ ein Homomorphismus ist,
3. ψ injektiv ist und
4. ψ surjektiv ist.

1. Zu zeigen: Diese Definition ist repräsentantenunabhängig, d.h. aus $u \sim_L v$ folgt $\psi(u) = \psi(v)$, d.h. $\delta_u = \delta_v$.

   Angenommen, $u \sim_L v$ und $\delta_u \neq \delta_v$. Dann gibt es $q \in Q$ mit

   $$\delta_u(q) = \delta(q, u) = q_1 \neq q_2 = \delta(q, v) = \delta_v(q).$$

Da $\mathcal{A}$ minimal ist, ist q erreichbar, d.h. es gibt ein $x \in \Sigma^*$ mit $q = \delta(q_0, x)$.

Wegen $u \sim_L v$ gilt nun aber für alle $y \in \Sigma^*$

$$xuy \in L \text{ gdw } xvy \in L.$$

Das bedeutet, für alle $y \in \Sigma^*$ gilt

$$\begin{aligned} \delta(q_1, y) &= \delta(q_0, xuy) \in F \text{ gdw} \\ \delta(q_0, xvy) &= \delta(q_2, y) \in F. \end{aligned}$$

Das bedeutet aber, daß $L_{q_1}(\mathcal{A}) = L_{q_2}(\mathcal{A})$ ist, und da $\mathcal{A}$ minimal ist, folgt daraus $q_1 = q_2$ im Widerspruch zur Annahme.

2. Zu zeigen: ψ ist ein Monoid–Homomorphismus, d.h.

   Für alle $[u]_{\sim_L}, [v]_{\sim_L} \in \Sigma^*/_{\sim_L}$ gilt $\psi([u]_{\sim_L} \bullet [v]_{\sim_L}) = \psi([u]_{\sim_L}) \circ \psi([v]_{\sim_L})$.

   Durch Nachrechnen sieht man leicht, daß

   $$\psi([u]_{\sim_L} \bullet [v]_{\sim_L}) = \psi([uv]_{\sim_L}) = \delta_{uv} = \delta_u \circ \delta_v = \psi([u]_{\sim_L}) \circ \psi([v]_{\sim_L}).$$

3. Zu zeigen: ψ ist injektiv, d.h. aus $\delta_u = \delta_v$ folgt $[u]_{\sim_L} = [v]_{\sim_L}$.

   Es sei $\delta_u = \delta_v$ und $x, y \in \Sigma^*$ beliebig. Wir zeigen, $xuy \in L$ impliziert $xvy \in L$ (Aus Symmetriegründen folgt dann automatisch die andere Richtung und wir haben $u \sim_L v$). Sei nun also $xuy \in L$. Dann ist $\delta(q_0, xuy) \in F$. Für $q_1 := \delta(q_0, x)$ gilt:

   $$\begin{aligned} \delta(q_0, xuy) &= \delta(q_1, uy) \\ &= \delta_y(\delta_u(q_1)) \\ &= \delta_y(\delta_v(q_1)) \text{ mit } \delta_u = \delta_v \\ &= \delta(q_1, vy) \\ &= \delta(q_0, xvy). \end{aligned}$$

   Wegen $\delta(q_0, xvy) \in F$ folgt $xvy \in L$.

4. Zu zeigen: ψ ist surjektiv. Da jedes $\delta_u \in M_L$ Bild von $[u]_{\sim_L}$ ist, ist ψ trivialerweise surjektiv. ∎

Für reguläre Sprachen haben wir also den Zusammenhang zwischen syntaktischer Kongruenz und syntaktischem Monoid hergestellt. Daraus folgt das nächste Korollar:

Korollar 1.13 L ist regulär gdw $\sim_L$ hat endlichen Index[3].

Beweis:

"$\rightsquigarrow$" Einfach, da $M_L \simeq \Sigma^*/_{\sim_L}$ als Übergangsmonoid eines endlichen Automaten endlich ist.

"$\leftsquigarrow$" Wir zeigen, daß L von $\Sigma^*/_{\sim_L}$ akzeptiert wird (und somit regulär ist, falls $\Sigma^*/_{\sim_L}$ endlich ist — siehe Satz 1.8). Dazu definieren wir

$$\begin{aligned} \phi : \Sigma^* &\to \Sigma^*/_{\sim_L} \\ u &\mapsto [u]_{\sim_L}, \end{aligned}$$

und zeigen $\phi^{-1}(\phi(L)) = L$. Damit dies gilt, muß aus $u \in L$ und $\phi(u) = \phi(v)$ folgen, daß $v \in L$ ist. $\phi(u) = \phi(v)$ gilt aber genau dann, wenn für alle $x, y \in \Sigma^*$ gilt: $xuy \in L$ gdw $xvy \in L$. Insbesondere für $x = \epsilon = y$ folgt $u \in L$ gdw $v \in L$. ∎

Beispiel 1.14 Es sei $L = \Sigma^*\sigma\tau\Sigma^*$ für $\Sigma = \{\sigma, \tau\}$. Offensichtlich ist $\mathcal{A}_1$ in Abbildung 1.1 ein nicht–deterministischer Automat für L. Die Potenzmengenkonstruktion (bei der wir uns auf die erreichbaren Mengen beschränken können) liefert uns mit $\mathcal{A}_2$ einen deterministischen Automaten für L, der aber noch nicht minimal sein muß. Dafür muß er noch minimiert werden. Man berechnet dazu die Äquivalenzklassen der folgenden Relation auf den Zuständen und faßt hinterher äquivalente Zustände zu einem neuen zusammen: $q \sim_{\mathcal{A}} q'$ gdw $L_q(\mathcal{A}) = L_{q'}(\mathcal{A})$.

Zur Bestimmung dieser Äquivalenzklassen berechnen wir zunächst $\sim_n$ für $n \geq 0$.

$$\begin{aligned} q \sim_0 q' \quad &\text{gdw} \quad (q \in F \text{ und } q' \in F) \text{ oder } (q \notin F \text{ und } q' \notin F), \\ q \sim_{n+1} q' \quad &\text{gdw} \quad q \sim_n q' \text{ und für alle } \sigma \in \Sigma \text{ gilt } \delta(q, \sigma) \sim_n \delta(q', \sigma). \end{aligned}$$

[3]Der Index einer Äquivalenzrelation ist die Anzahl ihrer Äquivalenzklassen

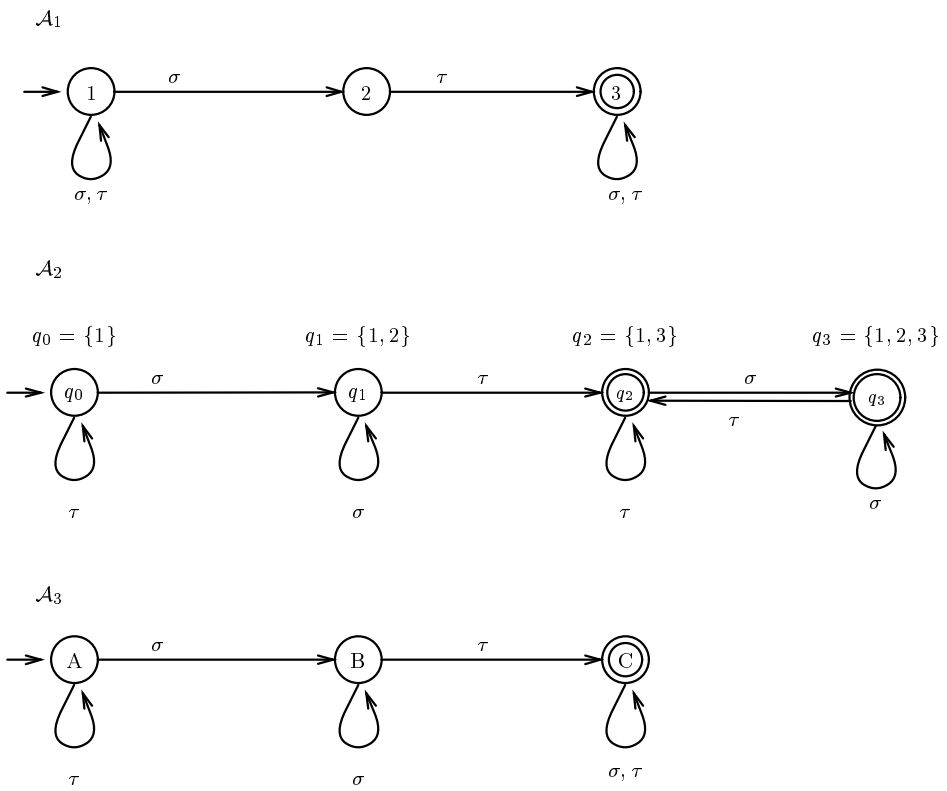


Abbildung 1.1: Die Automaten $\mathcal{A}_1, \mathcal{A}_2$ und $\mathcal{A}_3$

Da Q endlich ist, existiert ein k mit $\sim_k = \sim_{k+1}$ und man zeigt leicht, daß $\sim_k = \sim_{\mathcal{A}}$ ist. In $\sim_0$ bestehen die Äquivalenzklassen aus den Endzuständen $\{q_2, q_3\}$ und den Nicht–Endzuständen $\{q_0, q_1\}$. Bezüglich $\sim_1$ gibt es drei Äquivalenzklassen: Die Endzustände $\{q_2, q_3\}$, der erste Zustand $\{q_0\}$ und der zweite Zustand $\{q_1\}$. Da $\sim_2 = \sim_1$ gilt, folgt, daß der minimale Automat drei Zustände hat, die wir mit A, B und C in $\mathcal{A}_3$ bezeichnet haben.

Wir betrachten nun das zu $\mathcal{A}_3$ zugehörige Übergangsmonoid:

$$\delta_\epsilon = \frac{A\ B\ C}{A\ B\ C}, \quad \delta_\sigma = \frac{A\ B\ C}{B\ B\ C} = \delta_{\sigma\sigma}, \quad \delta_\tau = \frac{A\ B\ C}{A\ C\ C} = \delta_{\tau\tau},$$

$$\delta_{\sigma\tau} = \frac{A\ B\ C}{C\ C\ C} = \delta_{\tau\sigma\tau} = \delta_{\sigma\tau w} \text{ für alle } w \in \Sigma^*$$

$$\delta_{\tau\sigma} = \frac{A\ B\ C}{B\ C\ C} = \delta_{\tau\sigma\sigma}.$$

Es ist also $M_{\mathcal{A}_3} = \{\delta_\epsilon, \delta_\sigma, \delta_\tau, \delta_{\sigma\tau}, \delta_{\tau\sigma}\}$, wobei die oben genannten Identitäten die Multiplikationstafel liefern. Z.B. gilt $\delta_{\tau\sigma} \circ \delta_\tau = \delta_{\sigma\tau}$.

Beispiel 1.15 Nicht jedes endliche Monoid ist syntaktisches Monoid einer regulären Sprache: Sei $M = \{1, a, b, c\}$ mit der durch die folgende Multiplikationstafel gegebenen Operation.

$\cdot$	1	a	b	c
1	1	a	b	c
a	a	a	b	c
b	b	a	b	c
c	c	a	b	c

a, b, c sind sogenannte Rechtsnullen, d.h. für alle $x \in M$ gilt $x \cdot a = a$, $x \cdot b = b$, $x \cdot c = c$.

Wir werden nun zeigen, daß M kein syntaktisches Monoid ist:

Angenommen, M ist syntaktisches Monoid von L. Dann gibt es einen Isomorphismus $\phi : M \to \Sigma^*/_{\sim_L}$. Von $\sim_L$ wissen wir:

$$\begin{aligned} u \in L &\rightsquigarrow [u]_{\sim_L} \subseteq L, \\ u \notin L &\rightsquigarrow [u]_{\sim_L} \subseteq \overline{L}. \end{aligned}$$

Daher muß für $m \in \{a, b, c\}$ gelten:

$$\phi(m) \subseteq L \text{ oder } \phi(m) \subseteq \overline{L}.$$

Es gibt also zwei Elemente in $\{a, b, c\}$, die beide in L oder beide in $\overline{L}$ liegen. Wir betrachten nun exemplarisch den Fall $\phi(a) \subseteq L$ und $\phi(b) \subseteq L$. Alle anderen Fälle (wie z.B. $\phi(b) \subseteq \overline{L}, \phi(c) \subseteq \overline{L}$) können entsprechend behandelt werden.

Behauptung: $\phi(a) \subseteq L$ und $\phi(b) \subseteq L \rightsquigarrow \phi(a) = \phi(b)$ im Widerspruch zur Injektivität von ϕ.

Sei nun $\phi(a) = [u]_{\sim_L}, \phi(b) = [v]_{\sim_L}$. Wir müssen also zeigen, daß $u \sim_L v$ gilt (das heißt $\phi(a) = \phi(b)$). Seien dazu $x, y \in \Sigma^*$ gegeben.

1. Fall $[y]_{\sim_L} \neq [\epsilon]_{\sim_L}$ ($[y]_{\sim_L}$ ist also nicht das Einselement von $\Sigma^*/_{\sim_L}$). Dann gilt

$$\begin{aligned}
\phi^{-1}([xuy]_{\sim_L}) &= \phi^{-1}([x]_{\sim_L}[u]_{\sim_L}[y]_{\sim_L}) \\
&= \phi^{-1}([x]_{\sim_L})\phi^{-1}([u]_{\sim_L})\phi^{-1}([y]_{\sim_L}) \\
&= \phi^{-1}([y]_{\sim_L}) \\
&\quad \text{(da } \phi^{-1}([y]_{\sim_L}) \text{ nicht Einselement ist, ist es} \\
&\quad \text{Rechtsnull, d.h. es “frißt” alles links von sich)} \\
&= \phi^{-1}([x]_{\sim_L})\phi^{-1}([v]_{\sim_L})\phi^{-1}([y]_{\sim_L}) \\
&= \phi^{-1}([xvy]_{\sim_L}).
\end{aligned}$$

Daraus folgt $[xuy]_{\sim_L} = [xvy]_{\sim_L}$, d.h. $xuy \sim_L xvy$ und insbesondere gilt $xuy \in L$ gdw $xvy \in L$.

2. Fall $[y]_{\sim_L} = [\epsilon]_{\sim_L}$.

Dann gilt

$$\begin{aligned}
\phi^{-1}([xuy]_{\sim_L}) &= \phi^{-1}([x]_{\sim_L}[u]_{\sim_L}[y]_{\sim_L}) \\
&= \phi^{-1}([x]_{\sim_L})\phi^{-1}([u]_{\sim_L}) \\
&\quad (\text{da } \phi^{-1}([y]_{\sim_L}) \text{ Einselement ist}) \\
&= \phi^{-1}([u]_{\sim_L}) = a \\
&\quad (\text{da } \phi^{-1}([u]_{\sim_L}) = a \text{ Rechtsnull ist}) \\
\phi^{-1}([xvy]_{\sim_L}) &= \phi^{-1}([x]_{\sim_L}[v]_{\sim_L}[y]_{\sim_L}) \\
&= \phi^{-1}([x]_{\sim_L})\phi^{-1}([v]_{\sim_L}) \\
&= \phi^{-1}([v]_{\sim_L}) = b.
\end{aligned}$$

Da $\phi(a) \subseteq L$ und $\phi(b) \subseteq L$ folgt daraus sowohl $xuy \in L$ als auch $xvy \in L$.

Fassen wir die beiden Fälle zusammen, so folgt, daß für beliebige $x, y \in \Sigma^*$ gilt: $xuy \in L$ gdw $xvy \in L$. Aus der Vervollständigung des Beweises um die anderen Fälle folgt dann, daß, egal wie ϕ sich verhält, es nie injektiv sein kein. ∎

Der Zusammenhang zwischen regulären Sprachen und endlichen Monoiden läßt sich ausnutzen, um bestimmte Teilklassen der Klasse der regulären Sprachen über bestimmte Klassen endlicher Monoide zu definieren.

Definition 1.16 Es sei V eine Klasse endlicher Monoide. Wir definieren die zugehörige Klasse $L(V)$ von Sprachen durch

$$L(V)_\Sigma = \{L \subseteq \Sigma^* \mid \Sigma^*/_{\sim_L} \in V\}.$$

Eine Sprache $L \subseteq \Sigma^*$ ist also in $L(V)_\Sigma$, falls ihr syntaktisches Monoid in V ist. Mit Korollar 1.13 sind alle Sprachen aus $L(V)_\Sigma$ regulär. Um "vernünftige" Klassen von Sprachen zu erhalten (mit "vernünftigen" Abschlußeigenschaften), muß man von Klassen endlicher Monoide ausgehen, die vernünftige Abschlußeigenschaften haben — und überprüfen, daß sich diese Eigenschaften auf die Sprachklassen übertragen. Als besonders interessant haben sich die sogenannten M–Varietäten erwiesen:

Definition 1.17 Eine nicht–leere Klasse endlicher Monoide, die abgeschlossen ist unter Bildung von Untermonoiden, binärem direkten Produkt und homomorphen Bildern, heißt *M–Varietät*.

Erklärung zu den Abschlußeigenschaften:

Untermonoid: $N \subseteq M$ heißt Untermonoid von $(M, \bullet, 1)$, falls gilt: $1 \in N$ und $n, n' \in N \rightsquigarrow n \bullet n' \in N$. Ein Klasse V von Monoiden ist abgeschlossen unter Bildung von Untermonoiden, wenn gilt $M \in V$ und N Untermonoid von $M \rightsquigarrow N \in V$.

Homomorphes Bild: Ist $\phi : M_1 \to M_2$ ein surjektiver Homomorphismus, so ist M_2 homomorphes Bild von M_1. Ein Klasse V von Monoiden ist abgeschlossen unter homomorphen Bildern, wenn aus $M_1 \in V$ und M_2 homomorphes Bild von M_1 folgt, daß auch $M_2 \in V$.

Direktes Produkt: $M_1 \times M_2 = \{(m, n) \mid m \in M_1 \text{ und } n \in M_2\}$ ist das direkte Produkt zweier Monoide M_1, M_2 mit $(m, n) \bullet_{M_1 \times M_2} (m', n') := (m \bullet_{M_1} m', n \bullet_{M_2} n')$ und $1_{M_1 \times M_2} := (1_{M_1}, 1_{M_2})$. Ein Klasse V von Monoiden ist abgeschlossen unter direktem Produkt, wenn gilt $M_1, M_2 \in V \rightsquigarrow M_1 \times M_2 \in V$.

Beispiel 1.18 Sei V_G die Klasse aller endlichen Gruppen. Wir werden zeigen, daß V_G eine M–Varietät ist.

Daß das homomorphe Bild einer endlichen Gruppe wieder eine endliche Gruppe ist und das direkte Produkt zweier endlicher Gruppen wiederum eine endliche Gruppe ist, läßt sich leicht nachprüfen. Interessant ist der Abschluß unter Untermonoidbildung (Wir benötigen dazu die Endlichkeit: so ist z.B. $(\mathbb{N}, +, 0)$ Untermonoid von $(Z, +, 0)$, aber keine Gruppe!).

Sei also G eine endliche Gruppe und M ein Untermonoid von G. Wir zeigen, daß M auch eine Gruppe ist, d.h. für alle $n \in M$ gilt $n^{-1} \in M$. Dazu betrachten wir zu $n \in M$ folgende Abbildung:

$$\begin{array}{rcl} \phi_n : M & \to & M \\ m & \mapsto & m \bullet n. \end{array}$$

Da G Gruppe ist, kann man kürzen, d.h. $m \bullet n = m' \bullet n \rightsquigarrow m = m'$ und daraus folgt, daß ϕ_n injektiv ist. Da M endlich ist, ist ϕ_n auch surjektiv.

Deshalb gibt es ein $\tilde{n} \in M$ mit $\phi_n(\tilde{n}) = 1$, d.h. $n \bullet \tilde{n} = 1$. Die Konstruktion von ϕ_n zeigt also, daß das Inverse zu beliebigen $n \in M$ ebenfalls in M ist, also ist M Untergruppe von G und V_G eine M–Varietät.

M–Varietäten sind unter anderem deshalb interessant, weil sie "durch Gleichungen schließlich definiert" werden können (Erklärung folgt). Es sei X ein abzählbares Alphabet. Eine *Gleichung* ist von der Form $u \doteq v$ mit $u, v \in X^*$ (statt ϵ schreiben wir meist 1 in Gleichungen). Ein Monoid M *erfüllt* eine Gleichung $u \doteq v$, falls $\phi(u) = \phi(v)$ für jeden Homomorphismus $\phi : X^* \to M$ gilt. Z.B. ist $xy \doteq yx$ eine Gleichung $(x, y \in X)$, die von allen kommutativen Monoiden erfüllt wird: Sei M kommutatives Monoid und $\phi : X^* \to M$ Homomorphismus mit $\phi(x) = m,\ \phi(y) = n$. Dann gilt:

$$\phi(xy) = \phi(x) \bullet \phi(y) = m \bullet n \stackrel{!}{=} n \bullet m = \phi(y) \bullet \phi(x) = \phi(yx).$$

Definition 1.19 Es sei $U = (u_n \doteq v_n)_{n \geq 1}$ eine Folge von Gleichungen. Eine Klasse V von endlichen Monoiden wird *schließlich definiert* durch diese Folge U, falls für jedes endliche Monoid M gilt: $M \in V$ genau dann, wenn es für M ein $k_M \in \mathbb{N}$ gibt, so daß M alle Gleichungen $(u_n \doteq v_n)_{n \geq k_M}$ erfüllt.

Diese Folge U von Gleichungen muß natürlich nicht aus paarweise verschiedenen Gleichungen bestehen, sie kann z.B. auch aus einer einzigen bestehen. Der folgende Satz (ohne Beweis) beschreibt den oben genannten Zusammenhang zwischen M–Varietäten und Folgen von Gleichungen.

Satz 1.20 [Eilenberg, Schützenberger] Jede M–Varietät kann durch Gleichungen schließlich definiert werden und umgekehrt ist jede durch Gleichungen schließlich definierte Klasse endlicher Monoide eine M–Varietät.

Fortsetzung Beispiel 1.18 Die M–Varietät V_G aller endlichen Gruppen wird schließlich definiert durch

$$(x^{\bar{n}} \doteq 1)_{(n \geq 1)} \qquad \text{wobei } \bar{n} := kgV(1, \ldots, n).$$

Beweis:

1. Sei $G \in V_G$. Wir wissen: Für $k = |G|$ und alle $g \in G$ gilt: $g^k = 1$. Für alle $n \geq k$ ist k natürlich ein Teiler von $\bar{n} = kgV(1, \ldots, n)$. Deshalb erfüllt G also $g^{\bar{n}} = 1$ für alle $n \geq k$.

2. Erfüllt ein endliches Monoid G die Gleichung $g^k = 1$, so ist G eine endliche Gruppe, da es zu jedem $g \in G$ ein Inverses $g^{-1} = g^{k-1}$ gibt.

■

Aus den Abschlußeigenschaften einer M–Varietät V ergeben sich auch Abschlußeigenschaften für die Klasse regulärer Sprachen $L(V)$. Im folgenden werden wir exemplarisch den Abschluß unter Booleschen Operationen zeigen. Dazu benötigen wir zunächst folgendes Lemma.

Lemma 1.21 Sei V eine M–Varietät und $L \subseteq \Sigma^*$ eine Sprache, die von einem $M \in V$ akzeptiert wird. Dann ist auch $M_L = \Sigma^*/_{\sim_L} \in V$.

Beweis: Da M die Sprache L akzeptiert, gibt es einen Homomorphismus $\phi : \Sigma^* \to M$ und eine Menge $N \subseteq M$ mit $L = \phi^{-1}(N)$.

Ohne Einschränkung sei ϕ surjektiv (wenn es das nicht ist, dann nehmen wir das Untermonoid $M' = \phi(\Sigma^*) \subseteq M$, das ja auch in V ist, und $N' = N \cap M'$). Zu ϕ definieren wir nun die Äquivalenzrelation $\sim_\phi$ auf Σ^* durch

$$u \sim_\phi v \text{ gdw } \phi(u) = \phi(v)$$

und prüfen nach, daß $\sim_\phi$ tatsächlich eine Verfeinerung von $\sim_L$ ist ($\sim_\phi \subseteq \sim_L$). Das wird uns erlauben, einen surjektiven Homomorphismus von M nach M_L zu definieren, aus dessen Existenz dann folgt, daß $M_L \in V$ gilt.

Behauptung: $\sim_\phi \subseteq \sim_L$.

Sei $u \sim_\phi v$ und $x, y \in \Sigma^*$ beliebig und gelte $xuy \in L$. Dann ist $\phi(xuy) \in N$ und außerdem gilt:

$$\begin{aligned} \phi(xvy) &= \phi(x)\phi(v)\phi(y) \\ &= \phi(x)\phi(u)\phi(y) \text{ (wegen } u \sim_\phi v) \\ &= \phi(xuy) \in N. \end{aligned}$$

Damit ist dann auch $xvy \in L$. Das heißt, gilt $u \sim_\phi v$, dann auch $u \sim_L v$. Deshalb können wir nun folgenden Homomorphismus definieren:

$$\begin{aligned} \psi : M &\rightarrow \Sigma^*/_{\sim_L} \\ m &\mapsto [u]_{\sim_L} \text{ falls } \phi(u) = m. \end{aligned}$$

Da $\sim_\phi \subseteq \sim_L$ gilt, ist ψ repräsentantenunabhängig. Wegen der Surjektivität von ϕ gibt es zu jedem m ein u mit $\phi(u) = m$, also ist ψ wohldefiniert. Offensichtlich ist ψ ebenfalls surjektiv. Also folgt aus $M \in V$ und der Abgeschlossenheit von M–Varietäten unter homomorphen Bildern auch $M_L = \Sigma^*/_{\sim_L} \in V$. ∎

Nun also der versprochene Satz.

Satz 1.22 Sei V eine M–Varietät. Dann ist $L(V)_\Sigma$ abgeschlossen unter Vereinigung, Durchschnitt und Komplement.

Beweis: Es genügt natürlich, Abschluß unter Durchschnitt und Komplement zu zeigen.

Komplement: Man sieht leicht, daß eine Sprache und ihr Komplement dasselbe syntaktische Monoid haben: $M_L = M_{\Sigma^* \setminus L}$, da die Aussage $xuy \in L$ gdw $xvy \in L$ äquivalent ist zu $xuy \notin L$ gdw $xvy \notin L$. Das heißt, daß $u \sim_L v$ gdw $u \sim_{\Sigma^* \setminus L} v$ und deshalb $M_L = \Sigma^*/_{\sim_L} = \Sigma^*/_{\sim_{\Sigma^* \setminus L}} = M_{\Sigma^* \setminus L}$ ist.

Durchschnitt: Seien $L, L' \in L(V)_\Sigma$, d.h. $\Sigma^*/_{\sim_L}, \Sigma^*/_{\sim_{L'}} \in V$. Seien

$$\begin{aligned} \phi : \Sigma^* &\rightarrow \Sigma^*/_{\sim_L} \\ u &\mapsto [u]_{\sim_L} \\ \phi' : \Sigma^* &\rightarrow \Sigma^*/_{\sim_{L'}} \\ u &\mapsto [u]_{\sim_{L'}} \end{aligned}$$

die kanonischen Homomorphismen. Wir wissen (siehe Beweis von Korollar 1.13), daß für $N = \phi(L)$ und $N' = \phi'(L')$ gilt: $L = \phi^{-1}(N)$ und $L' = \phi^{-1}(N')$. Definieren wir

$$\begin{aligned} \phi : \Sigma^* &\rightarrow \Sigma^*/_{\sim_L} \times \Sigma^*/_{\sim_{L'}} \\ u &\mapsto ([u]_{\sim_L}, [u]_{\sim_{L'}}), \end{aligned}$$

dann sehen wir:

$$\begin{aligned}\phi^{-1}(N \times N') &:= \{u \in \Sigma^* \mid \phi(u) \in N \text{ und } \phi'(u) \in N'\} \\ &= \phi^{-1}(N) \cap \phi'^{-1}(N') \\ &= L \cap L'.\end{aligned}$$

Also akzeptiert $M = \Sigma^*/_{\sim_L} \times \Sigma^*/_{\sim_{L'}}$ die Sprache $L \cap L'$ und wegen der Abgeschlossenheit von V unter direktem Produkt ist $M \in V$. Lemma 1.21 liefert dann, daß das syntaktische Monoid $M_{L\cap L'}$ ebenfalls in V ist. ∎

Manchmal ist es günstiger, Halbgruppen statt Monoide zu betrachten (Erinnerung: In Halbgruppen fordert man nur eine assoziative binäre Operation, aber kein Einselement). Die bisher betrachteten Begriffe und Ergebnisse lassen sich auf (akzeptierende) Halbgruppen übertragen.

Die syntaktische Kongruenz $\sim_L$ ist auch eine Kongruenz auf der freien Halbgruppe $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. Für eine Sprache L ist die *syntaktische Halbgruppe* S_L (das S kommt von "semigroup") definiert durch $\Sigma^+/_{\sim_L}$. Diese erhält man ebenfalls über die Betrachtung des minimalen Automaten und dessen Übergangshalbgruppe $\{\delta_w \in Q^Q \mid w \in \Sigma^+\}$.

Eine *S–Varietät* ist eine unter (binären) direkten Produkten, homomorphen Bildern und Unterhalbgruppenbildung abgeschlossene Klasse endlicher Halbgruppen.

S–Varietäten können ebenfalls durch Gleichungen schließlich definiert werden, d.h. Satz 1.20 gilt auch in einer Halbgruppenvariante 1.20s, die dadurch entsteht, daß man in ihm "Monoide" durch "Halbgruppen" und "M–Varietäten" durch "S–Varietäten" ersetzt.

S–Varietäten werden betrachtet, da sie oft eine feinere Aufteilung liefern als M–Varietäten. Zum Beispiel wird die Gleichung $xy \doteq y$ nur von dem trivialen Monoid erfüllt ($1 = 1 \bullet m = m$ für alle $m \in M$). Aber es gibt nicht–triviale Halbgruppen, die diese Gleichung erfüllen.

Für eine S–Varietät V ist

$$L(V)_\Sigma = \{L \subseteq \Sigma^* \mid S_L \in V\}$$

die zugehörige Sprachklasse. Lemma 1.21 und Satz 1.22 gelten ebenfalls in ihren Halbgruppenvarianten 1.21s und 1.22s.

1.3 Reguläre Sprachen und logische Formeln

In diesem Abschnitt werden die grundlegenden Begriffe für die folgenden zwei Kapitel eingeführt, in denen wir den Zusammenhang herstellen zwischen einer bestimmte Teilklasse von Formeln der Logik erster und zweiter Stufe und den von ihnen beschriebenen Sprachklassen. Sollten Begriffe aus der Logik unbekannt sein, so findet man sie in [Sch92] erläutert.[4]

Wir verwenden Logik mit Gleichheit, d.h. es gibt ein binäres Prädikat "$\doteq$", das immer als Identität interpretiert wird. Als nicht–logische Symbole verwenden wir zunächst nur eine binäre Relation "$\dot{<}$" und endlich viele einstellige Prädikatssymbole $P_1, \ldots, P_k$. Außerdem betrachten wir nur *endliche* Interpretationen, in denen "$\dot{<}$" als *totale* Ordnung $<$ interpretiert wird.

Solche Interpretationen können als Wörter über $\Sigma = \{0,1\}^k$ aufgefaßt werden:

- $\mathrm{dom}(I) = \{d_1, \ldots, d_n\}$ wobei $d_1 < d_2 < \cdots < d_n$.
- Für $1 \leq i \leq n$ sei $\sigma_i = (b_{i1}, \ldots, b_{ik}) \in \Sigma$, wobei

$$b_{i_j} = \begin{cases} 1 & d_i \in P_j^I \\ 0 & \text{sonst.} \end{cases}$$

Dann entspricht I dem Wort $\sigma_1\sigma_2\cdots\sigma_n$. Umgekehrt liefert jedes Wort der Länge $m \geq 1$ über Σ (bis auf Isomorphie) genau eine Interpretation mit einem Interpretationsbereich der Mächtigkeit m.

Beispiel 1.23 Sei $k = 1$, d.h. $\Sigma = \{0,1\}$ und es gebe nur ein einstelliges Prädikatssymbol P_1. Dem Wort 010 entspricht eine Interpretation I mit $\mathrm{dom}(I) = \{d_1, d_2, d_3\}, d_1 < d_2 < d_3$ und $P_1^I = \{d_2\}$.

[4] Wie üblich lassen wir in logischen Formeln die Klammern häufig weg. Es gelten dann die folgenden Klammerungsregeln: Die logischen Junktoren $\Leftrightarrow$, $\Rightarrow$, $\vee$, $\wedge$, $\neg$ binden (in dieser Reihenfolge) mit zunehmender Stärke; der Skopus der Quantoren $\exists x.$ und $\forall y.$ reicht bis zur nächsten umgebenden schließenden Klammer (oder bis zum Ende der Formel). Die vollständige Klammerung der Formel $\forall x.(\exists y.\phi \vee \psi) \Leftrightarrow \chi \vee \neg\xi \wedge \theta \Rightarrow \zeta \wedge \exists y.\phi \vee \psi$ ist also $\forall x.((\exists y.(\phi \vee \psi)) \Leftrightarrow ((\chi \vee ((\neg\xi) \wedge \theta)) \Rightarrow (\zeta \wedge (\exists y.(\phi \vee \psi)))))$.

Anstelle von Interpretationen werden wir daher im folgenden stets Wörter über $\Sigma = \{0,1\}^k$ verwenden. Es macht daher Sinn, ein Wort $w \in \Sigma^*$ Modell einer Formel ϕ zu nennen ($w \models \phi$).

Definition 1.24 Sei ϕ eine geschlossene Formel der Logik erster Stufe, die die nicht–logischen Symbole $\dot{<}, P_1, \ldots, P_k$ verwendet. Sei $\Sigma = \{0,1\}^k$. Dann *definiert ϕ die Sprache*

$$L(\phi) = \{w \in \Sigma^+ \mid w \models \phi\}.$$

Da Interpretationen stets nicht–leeren Bereich haben, entspricht keine Interpretation dem leeren Wort ϵ. Deshalb enthält kein $L(\phi)$ das leere Wort. Diese Einschränkung ist aber unwesentlich, da z.B. eine Sprache L genau dann regulär ist, wenn $L \setminus \{\epsilon\}$ regulär ist.

Beispiel 1.25 Sei $k = 1$, d.h. $\Sigma = \{0,1\}$. Die Sprache 11^*0^* wird definiert durch die Formel

$$\begin{array}{l} \exists x. P_1(x) \wedge \\ \quad (\forall y. y \dot{\leq} x \Rightarrow P_1(y)) \wedge \\ \quad (\forall y. x \dot{<} y \Rightarrow \neg P_1(y)), \end{array}$$

wobei $x \dot{\leq} y$ eine Abkürzung für $(x \dot{<} y \vee x \doteq y)$ ist.

Satz 1.26 Boolesche Operatoren in Formeln entsprechen Booleschen Operatoren auf Sprachen. Mit anderen Worten gilt für geschlossene Formeln ϕ, θ:

$$\begin{array}{rcl} L(\neg\phi) & = & \Sigma^+ \setminus L(\phi), \\ L(\phi \wedge \theta) & = & L(\phi) \cap L(\theta), \\ L(\phi \vee \theta) & = & L(\phi) \cup L(\theta). \end{array}$$

Der Beweis ergibt sich direkt durch Einsetzen der Definitionen.

Bei der Beschreibung von Formeln sind folgende Abkürzungen hilfreich:

$Q_\sigma(x)$ Für jedes $\sigma \in \Sigma$ kürzt man mit $Q_\sigma(x)$ ab, daß an der Stelle x das Symbol σ steht. Für $k = 2, \Sigma = \{(0,0),(0,1),(1,0),(1,1)\}$ übersetzt man zum Beispiel $Q_{(1,1)}(x)$ mit $P_1(x) \wedge P_2(x)$, und $Q_{(1,0)}(x)$ mit $P_1(x) \wedge \neg P_2(x)$.

min(x) Durch die Formel $\min(x)$ drückt man aus, daß x der Wortanfang ist: $\min(x)$ entspricht $\neg\exists y.\, y \dot{<} x$.

max(x) Die Formel $\max(x)$ definiert entsprechend das Wortende: $\max(x)$ entspricht $\forall y.\, y \dot{\leq} x$.

succ(x, y) Die nächste Stelle nach x im Wort ist y: $\mathrm{succ}(x, y)$ entspricht $x \dot{<} y \wedge \neg\exists z.\, x \dot{<} z \wedge z \dot{<} y$.

***s*(x)** Anstelle der Formel $\mathrm{succ}(x, y)$ kann man auch eine Nachfolgerfunktion verwenden: Die Aussage $s(x) \doteq y$ entspricht $\mathrm{succ}(x, y) \vee (\max(x) \wedge x \doteq y)$.

min, max Entsprechend kann man auch statt der Prädikate $\min(x), \max(x)$ Konstanten $\min, \max$ einführen.

pred(x, y) Der Vorgänger läßt sich wie der Nachfolger definieren: $\mathrm{pred}(x, y)$ entspricht $y \dot{<} x \wedge \neg\exists z.\, y \dot{<} z \wedge z \dot{<} x$.

***p*(x)** Anstelle der Nachfolgerfunktion kann man wieder die Vorgängerfunktion verwenden: Die Aussage $p(x) \doteq y$ entspricht $\mathrm{pred}(x, y) \vee (\min(x) \wedge x \doteq y)$.

Den Gewinn an Übersichtlichkeit durch diese Abkürzungen illustriert das nächste Beispiel:

Beispiel 1.27 Die reguläre Sprache $a(ba)^*$ kann beschrieben werden durch:

$$
\begin{array}{l}
Q_a(\min) \wedge \\
(\forall x.\, \forall y.\, Q_a(x) \wedge \mathrm{succ}(x, y) \Rightarrow Q_b(y)) \wedge \\
(\forall x.\, \forall y.\, Q_b(x) \wedge \mathrm{succ}(x, y) \Rightarrow Q_a(y)) \wedge \\
Q_a(\max).
\end{array}
$$

Uns interessiert nun, welche Sprachen man durch Formeln der Logik erster Stufe beschreiben kann. Wir werden sehen, daß man nur reguläre Sprachen durch Formeln der Logik erster Stufe beschreiben kann — aber kann man jede reguläre Sprache durch Formeln der Logik erster Stufe beschreiben?

Beispiel 1.28 Die reguläre Sprache $L = a(aa)^*$ kann nicht durch eine Formel der Logik erster Stufe beschrieben werden. Wir werden später sehen, wie man das zeigt.

Erlaubt man Quantifizierung über einstellige Prädikate (und verläßt also die Logik erster Stufe), so kann man L durch folgende Formel definieren. Als Variablen für einstellige Prädikate verwenden wir im folgenden große Buchstaben und als Variablen für Objekte (Stellen im Wort) wie bisher kleine.

$L = a(aa)^*$ wird beschrieben durch:

$$\begin{aligned}\exists X.\, \exists Y.\, & X(\min) \wedge X(\max) \wedge \\ & (\forall x.\, \forall y.\, X(x) \wedge \mathrm{succ}(x,y) \Rightarrow Y(y)) \wedge \\ & (\forall x.\, \forall y.\, Y(x) \wedge \mathrm{succ}(x,y) \Rightarrow X(y)) \wedge \\ & (\forall x.\, X(x) \Leftrightarrow \neg Y(x)) \wedge \\ & (\forall x.\, Q_a(x)).\end{aligned}$$

Dabei steht X für die ungeraden Positionen im Wort und Y für die geraden. $X(\min)$, $X(\max)$ garantiert, daß das Wort mit derselben Parität aufhört, mit der es angefangen hat.

Wir werden später sehen, daß man durch Formeln, die derartige Quantifizierungen zweiter Stufe zulassen, genau die regulären Sprachen beschreiben kann. In Kapitel 3 werden wir die Klasse von Sprachen untersuchen, die genau durch Formeln der Logik erster Stufe (in denen also keine Quantifizierungen zweiter Stufe erlaubt sind) beschrieben werden können. Zum Aufwärmen betrachten wir in Kapitel 2 eine kleinere Klasse von Sprachen: Diejenigen, die durch quantorenfreie Formeln beschrieben werden können.

Kapitel 2

Verallgemeinert–definite Sprachen und quantorenfreie Formeln

Wir führen zunächst die Sprachklasse direkt ein und zeigen dann, wie man sie mit Hilfe von Halbgruppen und Formeln charakterisieren kann.

2.1 Verallgemeinert–definite Sprachen

Informell sind dies Sprachen, bei denen nur die ersten und letzten k Buchstaben eine Wortes interessant sind.

Definition 2.1 Die Klasse B_0 der *verallgemeinert–definiten Sprachen* ist wie folgt definiert: $L \subseteq \Sigma^*$ gehört zu $(B_0)_\Sigma$ genau dann, wenn es ein $k \geq 0$ gibt, so daß für alle $w \in L$ gilt:

$$\text{Ist } w = uv = v'u' \text{ mit } |u| = |u'| = k, \text{ so ist } u\Sigma^* \cap \Sigma^* u' \subseteq L.$$

Dabei bezeichnen $u\Sigma^*$ und $\Sigma^* u'$ die Wörter, die mit u beginnen bzw. mit u' aufhören.

Ob ein Wort w zu L gehört, hängt also nur von den ersten und letzten k Buchstaben ab.

Lemma 2.2 Für jedes endliche Alphabet Σ ist $(B_0)_\Sigma$ der Boolesche Abschluß der Sprachen $\{u\Sigma^* \mid u \in \Sigma^*\} \cup \{\Sigma^* u' \mid u' \in \Sigma^*\}$.

Beweis: Wir bezeichnen den Booleschen Abschluß der Sprachen $\{u\Sigma^* \mid u \in \Sigma^*\} \cup \{\Sigma^* u' \mid u' \in \Sigma^*\}$ mit B'_Σ und zeigen also $(B_0)_\Sigma = B'_\Sigma$.

"$\subseteq$" Es sei $L \in (B_0)_\Sigma$ und k die in Definition 2.1 geforderte Zahl.

1. $k = 0$:
   Dann ist entweder $L = \emptyset$ oder $L = \Sigma^*$. Es gilt $\emptyset \in B'_\Sigma$, da $\emptyset = u\Sigma^* \cap \overline{u\Sigma^*} \in B'_\Sigma$ für beliebiges u, und es gilt $\Sigma^* \in B'_\Sigma$, da $\Sigma^* = \epsilon\Sigma^* \in B'_\Sigma$.
2. $k > 0$:
   Wir zeigen

$$L = \bigcup_{\substack{|u|=k=|u'| \\ u\Sigma^* \cap \Sigma^* u' \subseteq L}} u\Sigma^* \cap \Sigma^* u' \cup \{v \in L \mid k > |v|\}.$$

   "$\supseteq$" ist trivial.

   "$\subseteq$" Es sei $w \in L$. Der Fall $|w| < k$ ist trivial. Sei also $|w| \geq k$. Damit existieren u, u', v, v' mit $|u| = |u'| = k$ und $w = uv = v'u'$. Aus $w \in L$ folgt nach Definition von $(B_0)_\Sigma$ bereits $u\Sigma^* \cap \Sigma^* u' \subseteq L$.

   Es bleibt zu zeigen, daß $\{v \in L \mid k > |v|\}$ im Booleschen Abschluß B'_Σ ist. Es gilt aber

$$\{v\} = v\Sigma^* \setminus v\Sigma\Sigma^* = v\Sigma^* \setminus (\bigcup_{\sigma \in \Sigma} v\sigma\Sigma^*) \in B'_\Sigma.$$

"$\supseteq$" Wir müssen zeigen, daß $w\Sigma^*, \Sigma^* w \in (B_0)_\Sigma$ und daß $(B_0)_\Sigma$ abgeschlossen ist unter den Booleschen Operationen.

1. Wir zeigen $w\Sigma^* \in (B_0)_\Sigma$. Wir wählen dazu $k = |w|$. Offenbar gilt für alle $w' = uv = v'u'$ (mit $|u| = |u'| = k$) $w' \in w\Sigma^*$ gdw $u = w$. Es ist daher $u\Sigma^* \cap \Sigma^* u' \subseteq w\Sigma^*$, d.h. die Bedingung in Definition 2.1 ist erfüllt und $w\Sigma^* \in (B_0)_\Sigma$.
2. Der Fall $\Sigma^* w$ kann entsprechend behandelt werden.
3. Wir zeigen, daß $(B_0)_\Sigma$ unter Vereinigung abgeschlossen ist. Es sei $L_1 \in (B_0)_\Sigma$, $L_2 \in (B_0)_\Sigma$ mit Zahlen k_1, k_2. Wir wählen $k = \max\{k_1, k_2\}$. Beachte: Ist $u = u_1 u_2$, so ist $u_1\Sigma^* \supseteq u\Sigma^*$ und $\Sigma^* u_2 \supseteq \Sigma^* u$. Damit folgt leicht, daß k das Gewünschte leistet.
4. Wir zeigen, daß $(B_0)_\Sigma$ unter Komplementbildung abgeschlossen ist. Es sei $L \in (B_0)_\Sigma$ und k die entsprechende Zahl. Diese ist auch passend für $\overline{L}$. Es sei $w = uv = v'u' \in \overline{L}$, $|u| = |u'| = k$. Wäre $u\Sigma^* \cap \Sigma^* u' \not\subseteq \overline{L}$, so gäbe es $w' \in u\Sigma^* \cap \Sigma^* u'$ mit $w' \in L$. Daraus würde aber $u\Sigma^* \cap \Sigma^* u' \subseteq L$ folgen, im Widerspruch zu $w \in \overline{L}$.
5. Da sich der Durchschnitt durch Vereinigung und Komplement ausdrücken läßt, folgt die Abgeschlossenheit von $(B_0)_\Sigma$ unter Durchschnitt unmittelbar. ∎

2.2 Die zugehörige S–Varietät

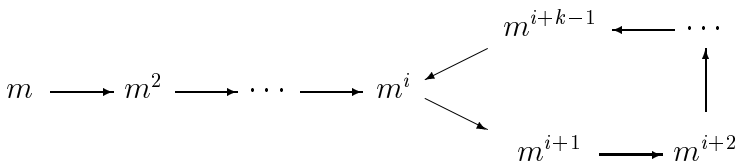
Um die S–Varietät für verallgemeinert–definite Sprachen zu definieren, benötigen wir zunächst den Begriff der idempotenten Elemente einer Halbgruppe.

Definition 2.3 Es sei S eine Halbgruppe. Ein Element $e \in S$ heißt *idempotent*, falls $e \bullet e = e$ gilt.

Offenbar ist ein Einselement idempotent. Es kann aber noch weitere Idempotente geben.

Satz 2.4 Es sei S eine *endliche* Halbgruppe und $m \in S$. Dann enthält die Menge $\{m, m^2, m^3, \dots\}$ ein Idempotentes.

Beweis: Da S endlich ist, existieren $i, k \geq 1$ mit $m^i = m^{i+k}$:



Offenbar existiert ein ℓ mit

- $i \leq \ell < i + k$ (sei $\ell = i + p$),
- $(\ell \bmod k) = 0$, d.h. es gibt ein ℓ' mit $\ell = k\ell'$.

Nun kann man zeigen, daß m^ℓ ein idempotentes Element ist:

$$m^\ell \bullet m^\ell = m^{i+p} \bullet m^{k\ell'} = m^{i+k\ell'} \bullet m^p = m^i \bullet m^p = m^\ell.$$

■

Beachte: Sind $m^i, m^{i+1}, \ldots, m^{i+k-1}$ alle verschieden (d.h. k ist minimal gewählt), so ist $\{m^i, m^{i+1}, \ldots, m^{i+k-1}\}$ eine zyklische Gruppe mit Einselement m^ℓ.

Korollar 2.5 Es sei S eine endliche Halbgruppe, $|S| \leq n$. Dann gilt für $\overline{n} = kgV(1, \ldots, n)$ und alle $m \in S$: $m^{\overline{n}}$ ist idempotent.

Beweis: Offenbar findet man im Beweis des Satzes 2.4 i und k mit $i+k-1 \leq n$. Es ist daher $\ell \leq n$ und somit ist ℓ ein Teiler von $\overline{n}$, d.h. $\overline{n} = \ell \cdot \ell'$. Es folgt $m^{\overline{n}} = (m^\ell)^{\ell'} = m^\ell$. ■

Definition 2.6 Die Klasse $\widehat{\mathbb{D}}$ besteht aus allen endlichen Halbgruppen, für die gilt: Für alle Idempotenten $e \in S$ ist $eSe = e$ (d.h. $\{eme \mid m \in S\} = \{e\}$).

Satz 2.7 $\widehat{\mathbb{D}}$ ist eine S–Varietät, die schließlich definiert ist durch

$$(*) \qquad x^{\overline{n}} y x^{\overline{n}} = x^{\overline{n}} \qquad (n \geq 1).$$

Beweis:

1. Es sei $S \in \widehat{\mathbb{D}}$. Mit Korollar 2.5 ist für $n \geq |S|$ und $m \in S$ stets $m^{\overline{n}}$ idempotent. Aus der Definition von $\widehat{\mathbb{D}}$ folgt für alle $m' \in S$: $m^{\overline{n}} m' m^{\overline{n}} = m^{\overline{n}}$, also erfüllt S die Gleichungen $(*)$ schließlich.
2. Gilt in S die Gleichung $(*)$ für ein n, so gilt für alle Idempotenten e und alle $m \in S$: $eme = e^{\overline{n}} m e^{\overline{n}} = e^{\overline{n}} = e$, also ist $eSe = e$. ∎

Lemma 2.8 Es sei $S \in \widehat{\mathbb{D}}$.

1. Für $n > |S|$ und $m_1, \ldots, m_n \in S$ ist $m_1 \bullet \cdots \bullet m_n$ idempotent.
2. Sind $e, f \in S$ idempotent und ist $m \in S$, so gilt $emf = ef$.

Beweis:

1. Betrachte $m_1, m_1m_2, m_1m_2m_3, \ldots, m_1 \cdots m_n$. Wegen $n > |S|$ gibt es $i < j$ mit $m_1 \cdots m_i = m_1 \cdots m_i m_{i+1} \cdots m_j$. Mit Satz 2.4 gibt es ein ℓ mit $(m_{i+1} \cdots m_j)^\ell =: e$ idempotent. Damit ist

$$\begin{aligned} m_1 \cdots m_n &= m_1 \cdots m_i m_{i+1} \cdots m_j m_{j+1} \cdots m_n \\ &= m_1 \cdots m_i (m_{i+1} \cdots m_j)^\ell m_{j+1} \cdots m_n \\ &= m_1 \cdots m_i \underbrace{(m_{i+1} \cdots m_j)^\ell}_{e} \underbrace{m_{j+1} \cdots m_n m_1 \cdots m_i}_{\text{in } S} \bullet \\ &\quad \underbrace{(m_{i+1} \cdots m_j)^\ell}_{e} m_{j+1} \cdots m_n \\ &= m_1 \cdots m_n m_1 \cdots m_n. \end{aligned}$$

2. $e(mf) = efe(mf) = e(femf) = ef$. Wir haben zuerst $e = efe$ und dann $f(em)f = f$ angewendet. Beide Gleichungen gelten nach Definition 2.6. ∎

Wir können jetzt den Zusammenhang zwischen $\widehat{\mathbb{D}}$ und B_0 zeigen.

Satz 2.9

$$L(\widehat{\mathbb{D}}) = B_0,$$

d.h. für beliebige $L \subseteq \Sigma^*$ gilt: $S_L = \Sigma^+/_{\sim_L} \in \widehat{\mathbb{D}} \Leftrightarrow L \in (B_0)_\Sigma$.

Beweis:

"$\supseteq$" Sei $L \in B_0$, d.h. mit Lemma 2.2 ist L im Booleschen Abschluß der Sprachen $\{u\Sigma^* \mid u \in \Sigma^*\} \cup \{\Sigma^* u' \mid u' \in \Sigma^*\}$. Da nach Satz 1.22s $L(\widehat{\mathbb{D}})$ unter Booleschen Operationen abgeschlossen ist, genügt es für $u\Sigma^*$ und $\Sigma^* u'$ zu zeigen, daß sie in $L(\widehat{\mathbb{D}})$ liegen. Wir beschränken uns hier auf den Fall $L = u\Sigma^*$, der andere ist symmetrisch.

Sei also $L = u\Sigma^*$, $|u| = n$.

1. Fall $n = 0$, d.h. $L = \Sigma^*$. Dann ist $\sim_L = \Sigma^* \times \Sigma^*$, d.h. S_L besteht aus einem einzigen Element, das folglich idempotent ist. Offenbar ist damit $S_L \in \widehat{\mathbb{D}}$.

2. Fall $n > 0$. Wir behaupten, daß für alle $w \in \Sigma^*$ mit $|w| \geq n$ und alle $v \in \Sigma^*$ gilt:

$$wv \sim_L w, \tag{2.1}$$

denn für alle $x, y \in \Sigma^*$ gilt offenbar

$$\begin{array}{lll} xwvy \in L = u\Sigma^* & \text{gdw} & xwvy \text{ beginnt mit einem } u \\ & \text{gdw} & xw \text{ beginnt mit einem } u \\ & & \quad (\text{da } |w| \geq |u| = n) \\ & \text{gdw} & xwy \text{ beginnt mit einem } u \\ & \text{gdw} & xwy \in L = u\Sigma^*. \end{array}$$

Nun nehmen wir ein beliebiges idempotentes Element $e \in S_L$, und betrachten ein $x \in \Sigma^*$ mit $e = [x]_{\sim_L} \in S_L = \Sigma^+/_{\sim_L}$. Wegen $|x| \geq 1$ ist $|x^n| \geq n$ und x erfüllt die Voraussetzung zu obiger Behauptung. Also gilt für beliebige $m = [y]_{\sim_L} \in S_L$

$$em = e^n m = [x^n]_{\sim_L} \bullet [y]_{\sim_L} = [x^n y]_{\sim_L} \overset{\text{Gl. 2.1}}{=} [x^n]_{\sim_L} = e^n = e.$$

Insbesondere folgt daraus für idempotente e: $eme = ee = e$, woraus schließlich $S_L \in \widehat{\mathbb{D}}$ folgt.

"$\subseteq$" Sei nun $S_L \in \widehat{\mathbb{D}}$, $n = |S_L| + 1$. Mit Lemma 2.8 (1) gilt für alle Wörter u der Länge n, daß $[u]_{\sim_L}$ idempotent ist:

$$[u]_{\sim_L} = [\sigma_1 \dots \sigma_n]_{\sim_L} = [\sigma_1]_{\sim_L} \bullet \cdots \bullet [\sigma_n]_{\sim_L}.$$

Es sei nun $x \in L$ mit $|x| \geq 2n$. Dann ist $x = uvu'$ mit $|u| = |u'| = n$ und $[u]_{\sim_L}, [u']_{\sim_L}$ sind idempotent. Mit Lemma 2.8 (2) gilt dann für alle Wörter $w \in \Sigma^*$:

$$\begin{aligned} [u]_{\sim_L} \bullet [w]_{\sim_L} \bullet [u']_{\sim_L} &= [u]_{\sim_L} \bullet [u']_{\sim_L} \\ &= [u]_{\sim_L} \bullet [v]_{\sim_L} \bullet [u']_{\sim_L}. \end{aligned}$$

Also ist $uwu' \sim_L uvu'$, und da $x = uvu' \in L$ vorausgesetzt war, gilt $uwu' \in L$ für alle $w \in \Sigma^*$. D.h. $u\Sigma^*u' \subseteq L$. Damit läßt sich L schreiben als:

$$L = \bigcup_{\substack{u\Sigma^*u' \subseteq L \\ |u|=|u'|=n}} u\Sigma^*u' \cup \{w \in L \mid |w| < 2n\}.$$

Für jedes Wort w ist das Singleton $\{w\} \in (B_0)_\Sigma$, da $\{w\} = w\Sigma^* \setminus \bigcup_{\sigma\in\Sigma} w\sigma\Sigma^*$. Bleibt zu zeigen, daß sich jedes der (endlich vielen!) $u\Sigma^*u'$ mit Hilfe Boolescher Operatoren aus $u\Sigma^*$ und Σ^*u darstellen läßt:

$$u\Sigma^*u' = (u\Sigma^* \cap \Sigma^*u') \setminus \{w \mid |w| < 2n\}.$$

Damit ist also $L \in (B_0)_\Sigma$. ∎

Korollar 2.10 Sei $L \subseteq \Sigma^*$ eine reguläre Sprache (gegeben durch einen regulären Ausdruck, endlichen Automaten, etc.). Dann kann man effektiv entscheiden, ob $L \in (B_0)_\Sigma$ gilt oder nicht.

Beweis: Wir wissen, daß zu L effektiv die syntaktische Halbgruppe S_L konstruiert werden kann. (Nimm die Übergangshalbgruppe $\{\delta_u \mid u \in \Sigma^+\}$ des minimalen Automaten.) Es bleibt zu prüfen: für alle Idempotente $e \in S_L$ gilt $eS_Le = e$. Da S_L laut Voraussetzung endlich ist, ist auch dies effektiv möglich. ∎

2.3 Quantorenfreie Formeln

In diesem Abschnitt betrachten wir Formeln, die als nicht–logische Symbole wiederum die binären Relationen "$\dot{<}$","$\doteq$" und die unären Relationen $P_1, \ldots, P_k$ verwenden. Zusätzlich erlauben wir die Verwendung der im ersten Kapitel definierten Konstanten $\min, \max$ und der Vorgänger– bzw. Nachfolgerfunktionen $s(x), p(x)$. Die Interpretation der Symbole sei wie in Kapitel 1.3 eingeschränkt. Interpretationen werden wieder als Wörter über $\Sigma = \{0,1\}^k$ betrachtet, und für $\sigma \in \Sigma$ verwenden wir wieder die Formel $Q_\sigma(x)$ als Abkürzung dafür, daß σ an der Stelle x steht.

Bemerkung: Wir beschäftigen uns mit quantorenfreien Formeln, in denen aber die Symbole $\min, \max, s(x), p(x)$ auftauchen dürfen. In Kapitel 1.3 hatten wir diese Symbole durch Formeln beschrieben, in denen Quantoren auftauchen. Daher ist es äußerst wichtig, daß wir diese Symbole explizit zur Verfügung haben. Die hier betrachteten Formeln bezeichnen wir als quantorenfrei, da wir diese oben erwähnten Symbole nicht als Abkürzungen (für Formeln mit Quantoren) auffassen.

Satz 2.11 Es sei $\Sigma = \{0,1\}^k$. Für $L \subseteq \Sigma^*$ sind äquivalent:

1. $L \in (B_0)_\Sigma$.
2. $L \setminus \{\epsilon\} = L(\phi)$ für eine quantorenfreie geschlossene Formel über den nicht–logischen Symbolen "$\dot{<}$", "$\doteq$",$P_1, \ldots, P_k, \min, \max, s, p$.

Bemerkung: In quantorenfreien geschlossenen Formeln kommen keine Variablen vor! Da $\{\epsilon\} \in (B_0)_\Sigma$ ist, gilt $L \in (B_0)_\Sigma$ genau dann, wenn $L \setminus \{\epsilon\} \in (B_0)_\Sigma$.

Beweis:
"$1 \rightsquigarrow 2$" Da Disjunktion von Formeln der Vereinigung von Sprachen entspricht und Negation dem Komplement, genügt es mit Lemma 2.2 Sprachen der Form $u\Sigma^*, \Sigma^* u'$ zu betrachten.

Für $u\Sigma^*$ konstruieren wir die zugehörige Formel wie folgt:

Sei $u = \sigma_1 \cdots \sigma_n$ mit $n \geq 1$. Dann ist

$$Q_{\sigma_1}(\min) \wedge Q_{\sigma_2}(s(\min)) \wedge Q_{\sigma_3}(s(s(\min))) \wedge \ldots$$
$$\ldots \wedge Q_{\sigma_n}(s^{n-1}(\min)) \wedge s^{n-2}(\min) \dot{<} \max.$$

Das letzte Literal ist nötig, um eine Interpretation der Kardinalität n oder größer zu erzwingen (da $s(\max) \doteq \max$). Es entfällt für $n = 1$.

Für $\Sigma^* u$ mit u wie oben sieht die zugehörige Formel so aus:

$$Q_{\sigma_n}(\max) \wedge Q_{\sigma_{n-1}}(p(\max)) \wedge \ldots \wedge Q_{\sigma_1}(p^{n-1}(\max)) \wedge \min \dot{<} p^{n-2}(\max)$$

Sei $u = \epsilon$. Dann ist $u\Sigma^* = \Sigma^* = \Sigma^* u$ und $\Sigma^* \setminus \{\epsilon\} = \Sigma^+$. Daher beschreibt eine beliebige, stets wahre Formel (Tautologie) L, z.B. $\min \doteq \min$.

"$2 \rightsquigarrow 1$" Sei ϕ eine quantorenfreie geschlossene Formel über den zur Verfügung stehenden Symbolen (ohne Variablen!). Alle Terme sind also aufgebaut aus $\min, \max, s, p$.

Behauptung: Diese können so normalisiert werden, daß sie nur noch von der Gestalt

$$\begin{array}{lll} s^n(\min) & \text{für ein} & n \geq 0 \text{ oder} \\ p^n(\max) & \text{für ein} & n \geq 0 \end{array}$$

sind. Dazu verwendet man, daß wir s, p so konstruiert haben, daß $s(\max) = \max$ und $p(\min) = \min$ gilt und mit Ausnahme der Extremalpunkte $\max, \min$ stets $s(p(d)) = d$ und $p(s(d)) = d$ gilt. Wir können daher ohne Einschränkung sagen, daß die Formel ϕ Boolesche Kombination von atomaren Formeln der folgenden Gestalt ist:

$$P_i(t), \quad \neg P_i(t), \quad t \doteq t', \quad t \dot{<} t', \quad t \dot{\leq} t'$$

für normalisierte Terme t, t' (Beachte: $\neg(t \doteq t') = t \dot{<} t' \vee t' \dot{<} t$ und $\neg(t \dot{<} t') = t' \dot{\leq} t$.)

- Eine atomare Formel der Form $P_i(s^n(\min))$ wird erfüllt von
  - bestimmten Wörtern w der Länge $|w| < n$. Da aber endliche Mengen von Wörtern in $(B_0)_\Sigma$ sind, machen uns diese "kurzen" Wörter kein Problem.

– Wörtern der Länge $\geq n$. Da es sich hier nicht mehr zwangsläufig um eine endliche Wortmenge handelt, muß sie genauer betrachtet werden: Ein Wort $w = \sigma_1 \ldots \sigma_m,\ m \geq n$ erfüllt $P_i(s^n(\min))$ gdw sein $(n+1)$ter Buchstabe $\sigma_{n+1} \in \{0,1\}^k$ an der $i$ten Komponente eine 1 hat. Die Menge von Wörtern mit dieser Eigenschaft läßt sich beschreiben durch:

$$\bigcup_{\substack{|u| = n \\ \sigma \text{ hat } i\text{te Komp. } 1}} u\sigma\Sigma^*. \tag{2.2}$$

Da jede der Mengen $u\sigma\Sigma^*$ in $(B_0)_\Sigma$ ist, ist auch ihre endliche Vereinigung in $(B_0)_\Sigma$.

- Entsprechendes gilt für atomare Formeln der Form $\neg P_i(s^n(\min))$: ersetze im Ausdruck 2.2 "hat $i$te Komponente 1" durch "hat $i$te Komponente 0".

- Für atomare Formeln der Form $P_i(p^n(\max))$ gilt ebenfalls bezüglich Wörtern der Länge $< n$ das oben gesagte, und die Menge der Wörter mit Länge $\geq n$ läßt sich beschreiben durch:

$$\bigcup_{\substack{|u| = n \\ \sigma \text{ hat } i\text{te Komp. } 1}} \Sigma^*\sigma u. \tag{2.3}$$

- Für $\neg P_i(p^n(\max))$ ersetze im Ausdruck 2.3 "hat $i$te Komponente 1" durch "hat $i$te Komponente 0".

- Formeln der Form $t \doteq t'$, $t \dot{<} t'$, $t \doteq t'$ sind entweder von allen Wörtern erfüllt, oder sie schränken lediglich die Länge der sie erfüllenden Wörter ein. Die Abbildung 2.1 illustriert die aufgeführten Fälle. (Zur Erinnerung: $s(\max) \doteq \max$, $p(\min) \doteq \min$.) Die atomare Formel

  – $s^n(\min) \doteq s^m(\min)$ mit $n < m$ erzwingt, daß alle sie erfüllende Wörter w Länge $|w| \leq n+1$ haben. Die Menge dieser Wörter ist endlich, und endliche Wortmengen sind in $(B_0)_\Sigma$.

- $s^n(\min) \dot{<} s^m(\min)$ mit $n < m$ erzwingt, daß alle sie erfüllende Wörter w Länge $|w| > n + 1$ haben. Diese Wortmenge ist zwar unendlich, aber trotzdem in $(B_0)_\Sigma$, da ihr Komplement endlich ist (dieses ist dann in $(B_0)_\Sigma$, und damit auch die Menge selbst).
- $p^n(\max) \dot{\leq} s^m(\min)$ wird von Wörtern w erfüllt, deren Länge $|w| \leq n + m + 1$ ist. Endliche Wortmengen sind in $(B_0)_\Sigma$.

Alle anderen Fälle können entsprechend behandelt werden. ∎

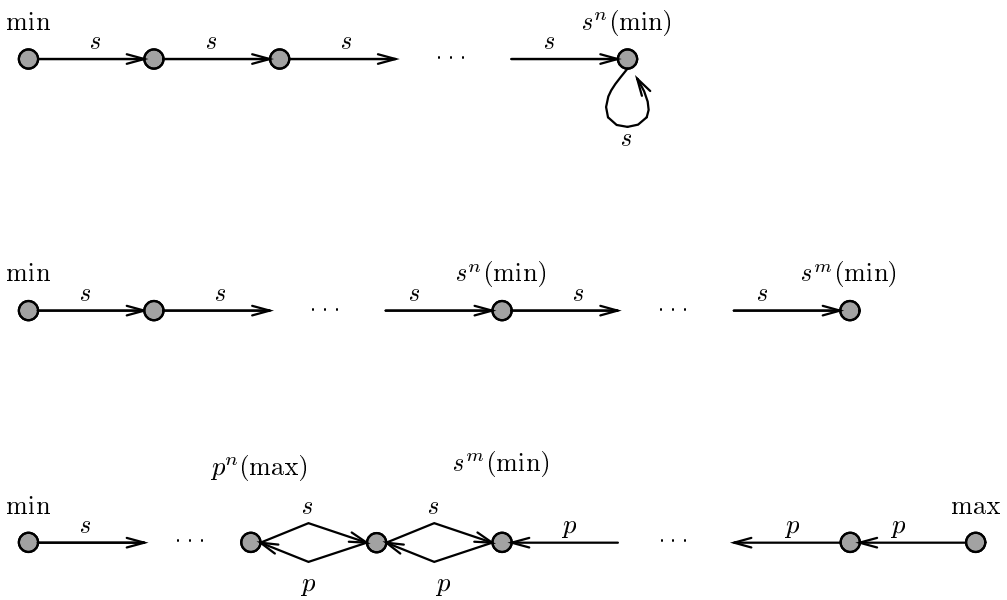


Abbildung 2.1: Beschränkung der Wortlängen durch Formeln $s^n(\min) \doteq s^m(\min)$, $s^n(\min) \dot{<} s^m(\min)$, $p^n(\max) \dot{\leq} s^m(\min)$

Kapitel 3

Sternfreie Sprachen

In diesem Kapitel betrachten wir die Sprachen, die genau durch Formeln der Logik erster Stufe beschrieben werden können.

3.1 Die Sprachklassen

Die Klasse der regulären Sprachen erhält man, indem man mit den *endlichen Sprachen* beginnt und dann den Abschluß unter

- Vereinigung ($L_1 \cup L_2$),
- Konkatenation ($L_1 \cdot L_2$) und
- Stern[1] (L^*)

bildet.

Würde man hier den Stern verbieten, so könnte man keine unendlichen Mengen erzeugen. Wir haben gesehen, daß reguläre Sprachen auch unter den Booleschen Operatoren Komplement und Durchschnitt abgeschlossen sind. Das Komplement endlicher Sprachen ist offenbar unendlich. Wir nehmen nun

[1] Mit "Stern" meinen wir im folgenden immer den Kleene–Stern.

Komplement und Durchschnitt als Abschlußoperatoren hinzu und verbieten dafür den Stern.

Definition 3.1 Für ein endliches Alphabet Σ ist die Klasse der *sternfreien Sprachen* SF_Σ über Σ die kleinste Klasse mit

- alle endlichen Sprachen über Σ sind in SF_Σ;
- sind $L, L_1, L_2 \in SF_\Sigma$, so auch $L_1 \cdot L_2$, $L_1 \cup L_2$, $L_1 \cap L_2$ und $\overline{L} = \Sigma^* \setminus L$.

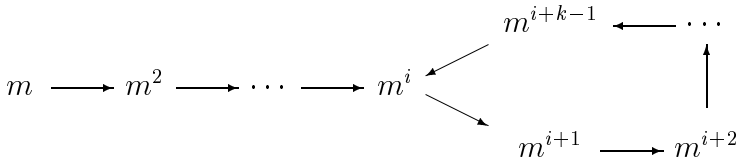
Beispiel 3.2

1. Σ^* ist offensichtlich sternfrei, da $\Sigma^* = \overline{\emptyset}$.
2. Ist $\Delta \subseteq \Sigma$, so ist $\Delta^* \in SF_\Sigma$, da
$$\Delta^* = \Sigma^* \setminus (\Sigma^* \cdot (\Sigma \setminus \Delta) \cdot \Sigma^*) = \overline{\emptyset} \cap \overline{\overline{\emptyset} \cdot (\Sigma \cap \overline{\Delta}) \cdot \overline{\emptyset}}.$$
3. Für $\Sigma = \{a, b\}$ ist $a(ba)^* \in SF_\Sigma$, da
$$a \cdot (ba)^* = a \cdot (\Sigma^* \setminus (a\Sigma^* \cup \Sigma^* b \cup \Sigma^* aa\Sigma^* \cup \Sigma^* bb\Sigma^*))$$
und wir in 1. gesehen haben, daß $\Sigma^* \in SF_\Sigma$.

Das dritte Beispiel zeigt, daß auch Sprachen, bei denen die offensichtliche Beschreibung den Stern verwendet, trotzdem sternfrei sein können. Kann man einer regulären Sprache (gegeben durch einen regulären Ausdruck mit Stern oder einen Automaten) ansehen, ob sie sternfrei ist? Konkreter: Wie zeigt man, daß $a(aa)^*$ *nicht* sternfrei ist? Man kann dazu die Charakterisierung der sternfreien Sprachen durch endliche Monoide verwenden.

3.2 Aperiodische Monoide

In Kapitel 2 haben wir gesehen, daß für ein Element m einer endlichen Halbgruppe die Menge $\{m, m^2, m^3, \dots\}$ stets ein Idempotentes enthält:



Bei aperiodischen Monoiden kann hier stets $k = 1$ gewählt werden.

Definition 3.3 Ein endliches Monoid M heißt *aperiodisch*, falls es ein $n \geq 1$ gibt mit $m^{n+1} = m^n$ für alle $m \in M$.

Die *Klasse der aperiodischen Monoide Ap* wird also schließlich definiert durch die Folge von Gleichungen $(x^{n+1} = x^n)_{n\geq 1}$. (Beachte dazu: Ist $m^{n+1} = m^n$, so gilt für alle $n' \geq n$ auch $m^{n'+1} = m^{n'}$.) Wir wissen daher, daß die Klasse der aperiodischen Monoide eine M–Varietät bildet.

Ist $m^i = m^{i+k}$ und sind $m^i, m^{i+1}, \dots, m^{i+k-1}$ paarweise verschieden, so ist $\{m^i, m^{i+1}, \dots, m^{i+k-1}\}$ bezüglich der Monoidoperation eine (zyklische) Gruppe. Das Einselement dieser Gruppe ist aber im allgemeinen nicht das Einselement des Monoids.

Definition 3.4 Es sei $(M, \bullet, 1)$ ein Monoid. Eine Teilmenge $G \subseteq M$ heißt *Gruppe in M*, falls G eine Unterhalbgruppe von M ist (d.h. $m, m' \in G \rightsquigarrow m \bullet m' \in G$), die bezüglich der Operation $\bullet$ in M eine Gruppe ist.

Das Einselement dieser Gruppe ist ein Idempotentes von M, kann aber verschieden vom Einselement 1 von M sein. Bei aperiodischen Monoiden M sind die zyklischen Gruppen $\{m^i, \dots, m^{i+k-1}\}$ stets trivial, d.h. sie haben Kardinalität 1. Dies gilt für alle Gruppen in M:

Satz 3.5 Ein endliches Monoid M ist genau dann aperiodisch, wenn es nur triviale Gruppen enthält.

Beweis:

"$\leadsto$" Es sei M aperiodisch und n so, daß $m^{n+1} = m^n$ für alle $m \in M$ gilt. Angenommen, $G \subseteq M$ ist eine nicht–triviale Gruppe in M, d.h. $|G| > 1$. Dann enthält G neben seinem Einselement e noch ein g mit $g \neq e$. Aus $g^{n+1} = g^n$ folgt aber $g = e$, da man in Gruppen kürzen kann.

"$\leadsto$" Es sei $m \in M$. Wir betrachten wieder $\{m, m^2, \ldots\}$. Es sei $k \geq 1$ minimal, so daß es ein $i \geq 1$ gibt mit $m^{i+k} = m^i$. Da $\{m^i, m^{i+1}, \ldots, m^{i+k-1}\}$ eine Gruppe in M ist, folgt $k = 1$. Es gibt also für jedes $m \in M$ ein $i_m \geq 1$ mit $m^{i_m+1} = m^{i_m}$. Offenbar folgt für alle $j \geq i_m$, daß $m^{j+1} = m^j$ gilt. Daher liefert $n := max\{i_m \mid m \in M\}$ die gewünschte Zahl mit $m^{n+1} = m^n$ für alle $m \in M$. ∎

Unser Interesse für aperiodische Monoide rührt daher, daß diese genau den sternfreien Sprachen entsprechen:

Satz 3.6 [Schützenberger] $L(Ap) = SF$, d.h. für $L \subseteq \Sigma^*$ gilt:

$$M_L \in Ap \quad \text{gdw} \quad L \in SF_\Sigma.$$

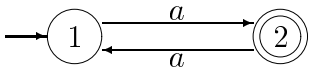
Der Beweis (insbesondere von "$\leadsto$") ist sehr aufwendig (siehe z.B. [Eil76, Vol. B]) und wird hier nicht geführt.

Korollar 3.7 Es sei $L \subseteq \Sigma^*$ eine reguläre Sprache (gegeben durch regulären Ausdruck, endlichen Automaten, ...). Dann kann man effektiv entscheiden, ob $L \in SF_\Sigma$ gilt oder nicht.

Beweis: Konstruiere das syntaktische Monoid und überprüfe, ob dieses aperiodisch ist. ∎

Beispiel 3.8 Es sei $\Sigma = \{a\}$. Dann ist $a(aa)^* \notin SF_\Sigma$.

Beweis: Der *minimale Automat* für $L = a(aa)^*$ ist $\mathcal{A}_L$:



Offenbar ist $\mathcal{A}_L$ deterministisch und die Zustände 1 und 2 sind nicht äquivalent. Die Trägermenge des Übergangsmonoids besteht aus den Funktionen $\delta_\epsilon = \{1 \mapsto 1; 2 \longmapsto 2\}$, $\delta_a = \{1 \mapsto 2; 2 \mapsto 1\}$ ($\delta_{aa} = \delta_\epsilon$). Damit ist $M_L = \{\delta_\epsilon, \delta_a\}$ mit $\delta_a \circ \delta_a = \delta_\epsilon$, d.h. M_L ist eine (zyklische) Gruppe der Ordnung 2. Damit ist M_L selbst eine nicht–triviale Gruppe in M_L. ∎

3.3 Formeln der Logik erster Stufe

Nachdem wir eben gesehen haben, daß sternfreie Sprachen den aperiodischen Monoiden entsprechen, werden wir nun zeigen, daß die sternfreien Sprachen genau durch Formeln der Logik erster Stufe definiert werden können.

Satz 3.9 Für $L \subseteq \Sigma^+$ sind äquivalent:

1. $L \in SF_\Sigma$.
2. $L = L(\phi)$ für eine geschlossene Formel ϕ der Logik erster Stufe über den nicht–logischen Symbolen $\doteq$, $\dot{<}$, Q_a ($a \in \Sigma$).

Beachte: An Stelle von $P_1, \ldots, P_n$ verwenden wir hier (ohne Einschränkung) direkt Q_a für $a \in \Sigma$ als Prädikatssymbole.

Der Beweis von "$1 \rightsquigarrow 2$" ist relativ einfach, während die andere Richtung sehr aufwendig ist.

3.3.1 Beweis von "$1 \rightsquigarrow 2$" des Satzes 3.9

Sternfreie Sprachen entstehen aus endlichen Sprachen durch Anwendung der Booleschen Operationen und der Konkatenation.

Endliche Sprachen

Offenbar genügt es, Einermengen $\{w\}$ für $w \in \Sigma^+$ zu betrachten. Für $w = a_1 \cdots a_n \in \Sigma^+$ sei

$$\begin{aligned}\phi_w \quad := \exists x_1 \ldots \exists x_n. \quad & Q_{a_1}(x_1) \wedge \ldots \wedge Q_{a_n}(x_n) \wedge \\ & \bigwedge_{j=1}^{n-1} \left(x_j \dot{<} x_{j+1} \wedge \neg \exists z. x_j \dot{<} z \wedge z \dot{<} x_{j+1} \right) \wedge \\ & \neg(\exists z. z \dot{<} x_1 \vee x_n \dot{<} z).\end{aligned}$$

Offensichtlich ist $L(\phi_w) = \{w\}$.

Boolesche Operatoren

Boolesche Operatoren auf Sprachen entsprechen nach Satz 1.26 den logischen Junktoren $\wedge$, $\vee$, $\neg$.

Konkatenation

Die Konkatenation $L_1 \cdot L_2$ entspricht im Prinzip dem existentiellen Quantor ("es gibt eine Trennstelle, so daß vor ihr ψ_{L_1} gilt und hinter ihr ψ_{L_2}"). Betrachten wir hier zunächst ein Beispiel:

$L_1 := a^+$ und $L_2 := b^+$ werden beschrieben durch $\phi_1 = \forall x. Q_a(x)$ und $\phi_2 = \forall x. Q_b(x)$. Die Sprache $L_1 \cdot L_2$ wird beschrieben durch

$$\exists z. \left((\forall x. x \dot{\leq} z \Rightarrow Q_a(x)) \wedge (\forall x. x \dot{>} z \Rightarrow Q_b(x)) \wedge \exists y. z \dot{<} y \right).$$

Der Quantor in ϕ_1 wird also auf die Elemente $\dot{\leq} z$ "relativiert" und der in ϕ_2 auf die Elemente $\dot{>} z$. Allgemein gelten für die *Relativierung* $\phi^{\dot{\leq} z}$ einer Formel ϕ auf Elemente $\dot{\leq} z$ die folgenden Gleichungen:

$$\begin{aligned}(\psi_1 \wedge \psi_2)^{\dot{\leq} z} &= \psi_1^{\dot{\leq} z} \wedge \psi_2^{\dot{\leq} z}, \quad (\exists x. \psi(x))^{\dot{\leq} z} = \exists x. x \dot{\leq} z \wedge (\psi(x))^{\dot{\leq} z}, \\ (\psi_1 \vee \psi_2)^{\dot{\leq} z} &= \psi_1^{\dot{\leq} z} \vee \psi_2^{\dot{\leq} z}, \quad (\forall x. \psi(x))^{\dot{\leq} z} = \forall x. x \dot{\leq} z \Rightarrow (\psi(x))^{\dot{\leq} z}, \\ (\neg\psi)^{\dot{\leq} z} &= \neg(\psi^{\dot{\leq} z}).\end{aligned}$$

Dabei komme ohne Einschränkung z nicht in den Formeln vor. Entsprechend ist $\phi^{\dot{>}z}$ definiert. Dann gilt: Ist $L_1 = L(\phi_1)$ und $L_2 = L(\phi_2)$, so ist $L_1 \cdot L_2 = L(\exists z. \phi_1^{\dot{<}z} \wedge \phi_2^{\dot{>}z} \wedge \exists y. z \dot{<} y)$.

Das schließt den Beweis von "$1 \rightsquigarrow 2$" ab.

3.3.2 Beweis von "$2 \rightsquigarrow 1$" des Satzes 3.9

Um zu zeigen, daß jede geschlossene Formel der Logik erster Stufe eine sternfreie Sprache definiert, betrachten wir die *Quantorentiefe* der Formel, d.h. die maximale Schachtelungstiefe von Quantoren in der Formel. Offenbar enthält jede geschlossene Formel mindestens einen Quantor (da es keine Terme ohne Variablen in unserer Sprache gibt). Da Negation vorhanden ist, können wir ohne Einschränkung davon ausgehen, daß alle Quantoren existentiell sind. Da die Junktoren $\wedge, \vee, \neg$ den Booleschen Operatoren auf Sprachen entsprechen, genügt es, sich auf Formeln der Form $\exists x. \phi(x)$ zu konzentrieren.

Induktionsanfang: $\exists x. \phi(x)$ hat Quantorentiefe 1

Dann enthält ϕ keine Quantoren (und damit keine anderen Variablen), d.h. ϕ ist Boolesche Kombination von Formeln

- $Q_a(x)$ oder $\neg Q_a(x)$,
- $x \dot{<} x$ oder $\neg(x \dot{<} x)$,
- $x \doteq x$ oder $\neg(x \doteq x)$.

Ohne Einschränkung sei diese Kombination in disjunktiver Normalform. Diese kann noch weiter normalisiert werden:

- Ersetze $x \dot{<} x$, $\neg(x \doteq x)$ durch false, d.h. lösche das Disjunkt.
- Ersetze $\neg(x \dot{<} x), x \doteq x$ durch true, d.h. entferne sie aus jedem Disjunkt.

Bleibt nach dieser Normalisierung true (oder false) übrig, so ist die definierte Sprache Σ^* (oder $\emptyset$) und damit insbesondere sternfrei.

Ansonsten kommen nur noch $Q_a(x)$ bzw. $\neg Q_a(x)$ in der normalisierten Formel vor. Dann kann man weiter normalisieren:

Enthält ein Disjunkt $Q_a(x)$, so

- lösche dieses Disjunkt, falls es auch $\neg Q_a(x)$ oder $Q_b(x)$ für $a \neq b$ enthält,
- sonst lösche alle $\neg Q_b(x)$ für $a \neq b$ aus dem Disjunkt.

Damit haben wir nur noch Disjunkte

- $Q_a(x)$,
- $\neg Q_{a_1}(x) \wedge \ldots \wedge \neg Q_{a_k}(x)$.

Wegen der Äquivalenz von $(\exists x.\, \phi \vee \psi)$ und $(\exists x.\, \phi) \vee (\exists x.\, \psi)$ brauchen wir nur Formeln der Form $\exists x.Q_a(x)$ und $\exists x.\neg Q_{a_1}(x) \wedge \ldots \wedge \neg Q_{a_k}(x)$ zu betrachten.

Die zugehörigen Sprachen $L(\exists x.Q_a(x)) = \Sigma^* a \Sigma^*$ und $L(\exists x.\neg Q_{a_1}(x) \wedge \ldots \wedge \neg Q_{a_k}(x)) = \Sigma^* \cdot (\Sigma \setminus \{a_1, \ldots, a_k\}) \cdot \Sigma^*$ sind sternfrei.

Damit ist die Quantorentiefe 1 abgehandelt.

Induktionsschritt

Es sei $\exists x.\, \phi(x)$ von Quantorentiefe $n+1$. Wir benötigen einige Notation und zwei Sätze, bevor wir weitermachen können.

Definition 3.10 Mit $L_{k,n}$ bezeichnen wir die Formeln der Logik erster Stufe (über den nicht–logischen Symbolen $\doteq$, $\dot{<}$, Q_a für $a \in \Sigma$), die k freie Variable und Quantorentiefe $\leq n$ haben.

Zum Beispiel gehört $\phi = \exists x.y \dot{<} x \wedge x \dot{<} z \wedge Q_a(x)$ zu $L_{2,1}$. Um diese Formel zu interpretieren, genügt ein Wort (z.B. baa) nicht. Man muß auch angeben, wie die freien Variablen y und z zu interpretieren sind (z.B. y durch 1 und z durch 3, d.h. durch die erste bzw. dritte Position in baa). Wir sagen $(baa, 1\ 3)$ erfüllt die Formel ϕ. Allgemein werden also Formeln aus $L_{k,n}$ durch Tupel $(w, \vec{s})$ interpretiert, wobei $w \in \Sigma^+$ und $\vec{s} = s_1 \cdots s_k$ mit $s_i \in \{1, \ldots, |w|\}$. $(w, \vec{s}) \models \phi$ ("$(w, \vec{s})$ erfüllt $\phi \in L_{k,n}$") ist in der offensichtlichen Weise definiert. Für $k = 0$ läßt man die leere Sequenz im allgemeinen weg.

Definition 3.11 Für $n \geq 0$ und $k \geq 0$ definieren wir

$$(w, \vec{s}) \equiv_{k,n} (v, \vec{t}) \quad \text{gdw} \quad \begin{array}{l} \text{für alle } \phi \in L_{k,n} \text{ gilt} \\ (w, \vec{s}) \models \phi \text{ gdw } (v, \vec{t}) \models \phi. \end{array}$$

Offenbar ist $\equiv_{k,n}$ eine Äquivalenzrelation. Für $k = 0$ schreiben wir statt $\equiv_{0,n}$ einfach $\equiv_n$ und $\{\epsilon\}$ definieren wir als eine eigene $\equiv_{0,n}$–Äquivalenzklasse. Um den Beweis von "$2 \rightsquigarrow 1$" des Satzes abzuschließen, benötigen wir zwei Sätze, die wir später in Unterabschnitt 3.3.3 beweisen werden.

Satz 3.12 Für alle $n \geq 0$ und $k \geq 0$ gibt es eine *endliche* Teilmenge $\Gamma_{k,n}$ von $L_{k,n}$, so daß gilt: Jedes Element von $L_{k,n}$ ist logisch äquivalent zu einem Element von $\Gamma_{k,n}$.

Logisch äquivalent heißt, daß die Formeln durch genau dieselben Tupel $(w, \vec{s})$ erfüllt werden. Eine einfache Folgerung aus dem Satz ist, daß jede Relation $\equiv_{k,n}$ endlichen Index hat.

Korollar 3.13 Für alle $n \geq 0$ und $k \geq 0$ hat $\equiv_{k,n}$ nur endlich viele Äquivalenzklassen.

Beweis: Die Klasse von $(w, \vec{s})$ ist eindeutig bestimmt durch die Teilmenge $\Gamma \subseteq \Gamma_{k,n}$ mit $\Gamma = \{\phi \in \Gamma_{k,n} \mid (w, \vec{s}) \models \phi\}$. Da es mit Satz 3.12 nur endlich viele solche Teilmengen gibt, existieren nur endlich viele Klassen. ∎

Der Begriff der durch eine Formel definierten Sprache kann auf den Fall $k > 0$ wie folgt ausgedehnt werden: Für $\phi \in L_{k,n}$ ist $L(\phi) := \{(w, \vec{s}) \mid (w, \vec{s}) \models \phi\}$.

Korollar 3.14 Für alle $n \geq 0$ und $k \geq 0$ und alle $\equiv_{k,n}$–Klassen W gibt es $\phi_W \in L_{k,n}$ mit $W = L(\phi_W)$.

Beweis:

$$W = [(w, \vec{s})]_{\equiv_{k,n}} = L(\bigwedge_{\substack{\phi \in \Gamma_{k,n} \\ (w,\vec{s}) \models \phi}} \phi \wedge \bigwedge_{\substack{\psi \in \Gamma_{k,n} \\ (w,\vec{s}) \not\models \psi}} \neg\psi)$$

Wie man sich leicht klarmachen kann, gilt diese Gleichung wegen der Definition von $\equiv_{k,n}$ und Satz 3.12. ∎

Korollar 3.15 Für alle $n \geq 0$ und $k \geq 0$ und alle $\phi \in L_{k,n}$ ist ϕ äquivalent zu einer endlichen Disjunktion von Formeln ϕ_W für $\equiv_{k,n}$–Klassen W.

Beweis:

$$\phi \text{ ist logisch äquivalent zu } \bigvee_{\substack{W = [(w,\vec{s})]_{\equiv_{k,n}} \\ (w,\vec{s}) \models \phi}} \phi_W.$$

Diese Disjunktion ist endlich, da $\equiv_{k,n}$ endlichen Index hat. ∎

Der zweite wichtige Satz sagt etwas über die Beziehung zwischen $\equiv_{0,n}$ und $\equiv_{1,n}$ aus und wird uns den Induktionsschritt ermöglichen:

Satz 3.16 Sei $n \geq 0$, $u, v, u', v' \in \Sigma^*$ und $a \in \Sigma$. Dann folgt aus $u \equiv_n u'$ und $v \equiv_n v'$ auch

$$(uav, |u| + 1) \equiv_{1,n} (u'av', |u'| + 1).$$

Mit diesem Satz, der in Abschnitt 3.3.3 bewiesen wird, können wir nun den Induktionsschritt abschließen.

Sei also $\exists x.\, \phi(x)$ von Quantorentiefe $n + 1$. Damit hat $\phi(x)$ Quantorentiefe n und eine freie Variable, ist also in $L_{1,n}$. Mit Korollar 3.15 wissen wir, daß es eine endliche Menge $\mathcal{W}$ von $\equiv_{1,n}$–Klassen gibt mit

$$\phi(x) \text{ ist logisch äquivalent zu } \bigvee_{W \in \mathcal{W}} \phi_W(x).$$

Daher sind auch die folgenden drei Formeln logisch äquivalent:

$$\exists x.\, \phi(x) \;\equiv\; \exists x.\, \bigvee_{W \in \mathcal{W}} \phi_W(x) \;\equiv\; \bigvee_{W \in \mathcal{W}} \exists x.\, \phi_W(x)$$

Wir betrachten daher jede einzelne Teilformel der Form $\exists x.\phi_W(x)$ separat und behandeln erst hinterher die Disjunktion durch Vereinigung.

Das folgende Lemma liefert uns die Sprache zu einer Formel der Form $\phi_W(x)$ als endliche Vereinigung von Konkatenationen sternfreier Sprachen.

Lemma 3.17

$$L(\exists x.\, \phi_W(x)) = \bigcup_{\substack{U=[u_0]_{\equiv_{0,n}},\ V=[v_0]_{\equiv_{0,n}} \\ a \in \Sigma \text{ mit} (u_0 a v_0, |u_0|+1) \in W}} UaV$$

Beweis:

"$\subseteq$" Es gilt

$$\begin{array}{lll} w \in L(\exists x.\, \phi_W(x)) & \text{gdw} & w = uav \text{ und } (uav, |u|+1) \models \phi_W(x) \\ & \text{gdw} & w = uav \text{ und } (uav, |u|+1) \in W. \end{array}$$

Damit ist $w = uav \in [u]_{\equiv_{0,n}} a [v]_{\equiv_{0,n}}$, also ist w in der rechten Seite enthalten.

"$\supseteq$" Aus $(u_0 a v_0, |u_0|+1) \in W = L(\phi_W(x))$ folgt $u_0 a v_0 \in L(\exists x.\, \phi_W(x))$. Es bleibt zu zeigen, daß für alle $u \equiv_{0,n} u_0$ und alle $v \equiv_{0,n} v_0$ gilt:

$$uav \in L(\exists x.\, \phi_W(x)).$$

Mit Satz 3.16 gilt

$$(u_0 a v_0, |u_0|+1) \equiv_{1,n} (uav, |u|+1),$$

was uns $(uav, |u|+1) \in W$ liefert. D.h. $(uav, |u|+1) \models \phi_W(x)$. Daraus folgt $uav \models \exists x.\, \phi_W(x)$. ∎

Mit Korollar 3.14 sind die $\equiv_{0,n}$ Klassen U, V aus Lemma 3.17 von der Form $U = L(\phi_U)$, $V = L(\phi_V)$ für $\phi_U, \phi_V \in L_{0,n}$. Die Induktion liefert $U, V \in SF_\Sigma$[2] und damit ist jedes $UaV \in SF_\Sigma$. Da $\equiv_{0,n}$ endlichen Index hat und Σ endlich ist, läßt sich also $L(\exists x.\, \phi_W(x))$ als endliche Vereinigung sternfreier Sprachen darstellen.

Dies schließt den Beweis von Satz 3.9 ab. ∎

Nun müssen noch die verwendeten Sätze bewiesen werden.

3.3.3 Beweis der Sätze 3.12 und 3.16

Wir werden zunächst Satz 3.12 beweisen.

Satz 3.12 Für alle $n \geq 0$ und $k \geq 0$ gibt es eine *endliche* Teilmenge $\Gamma_{k,n}$ von $L_{k,n}$, so daß gilt: Jedes Element von $L_{k,n}$ ist logisch äquivalent zu einem Element von $\Gamma_{k,n}$.

Beweis: Sei $\phi \in L_{k,n}$. Ohne Einschränkung können wir davon ausgehen, daß ϕ Boolesche Kombination ist von

- Elementen aus $L_{k,n-1}$ (Falls in einer dieser Teilformeln τ weniger als k freie Variablen vorkommen, können wir durch Anhängen von z.B. $\wedge(x \doteq x)$ für diejenigen x, die nicht in τ, aber in ϕ frei vorkommen, eine logisch äquivalente Teilformel finden, die in $L_{k,n-1}$ ist.) und von

- Formeln der Form $\exists x.\, \psi(x, y_1, \ldots, y_k)$, wobei $\psi(x, y_1, \ldots, y_k) \in L_{k+1,n-1}$ gilt (auch hier ist gegebenenfalls wieder das „Auffüllen“ fehlender freier Variablen nötig). Durch die Quantifizierung $\exists x.$ ist die gesamte Formel dann natürlich in $L_{k,n}$.

Wir verwenden nun eine Induktion über die Quantorentiefe n.

[2] Oder $U = \{\emptyset\}$ bzw. $V = \{\emptyset\}$. In diesem Fall kann U bzw. V in der Konkatenation einfach weggelassen werden.

Induktionsanfang ($n = 0$)**:** Sei $k \geq 0$ beliebig. Dann ist jede Formel $\psi(y_1, \ldots, y_k) \in L_{k,0}$ Boolesche Kombination von Formeln der Form

$$Q_a(y_i) \text{ für } a \in \Sigma,\ y_i \dot{<} y_j,\ y_i \doteq y_j. \tag{3.1}$$

Ohne Einschränkung genügt es offenbar, Formeln mit diesen freien Variablen zu betrachten (andere erhält man durch Umbenennung). Da Σ endlich ist, gibt es nur endlich viele Formeln der Form 3.1 und daher auch nur endlich viele Boolesche Kombinationen dieser Formeln, die nicht äquivalent sind. Diese bilden dann die endliche Teilmenge $\Gamma_{k,0} \subset L_{k,0}$.

Induktionsschritt ($n - 1 \to n$)**:** Eine Formel $\phi \in L_{k,n}$ ist Boolesche Kombination von Elementen aus $L_{k,n-1}$ und Formeln der Form $\exists x.\psi(x, y_1, \ldots, y_k)$ mit $\psi(x, y_1, \ldots, y_k) \in L_{k+1,n-1}$. Also ist ϕ äquivalent zu einer Booleschen Kombination von Elementen aus $\Gamma_{k,n-1}$ und Formeln $\exists x.\gamma(x, y_1, \ldots, y_k)$ mit $\gamma(x, y_1, \ldots, y_k) \in \Gamma_{k+1,n-1}$. Nach Induktionsannahme wissen wir bereits, daß die Mengen $\Gamma_{k+1,n-1}$ und $\Gamma_{k,n-1}$ endlich sind. ∎

Zum Beweis des Satzes 3.16 verwenden wir eine spieltheoretische Charakterisierung der Relation $\equiv_{k,n}$.

Definition 3.18 [Ehrenfeucht–Fraïssé Spiele] Es sei Σ ein endliches Alphabet. Wir betrachten zwei *Spieler* I und II, die auf zwei Wörtern $u, v \in \Sigma^+$ spielen. Ein *Zug* besteht darin, eine Position in u oder v zu wählen. Der Spieler I fängt an, dann wird abwechselnd gezogen. Macht I einen Zug in u, so muß II seinen nächsten Zug in v machen. Zieht I in v, muß II in u ziehen (I hat bei jedem seiner Züge die Wahl, in welchem Wort er ziehen möchte, II hat nie die Wahl. Es ist nicht verboten, bereits gezogene Positionen nochmals zu ziehen, aber dies ist für keinen der Spieler sinnvoll, falls er gewinnen will und es noch nicht gezogene Positionen gibt).

Ein *Spiel der Länge* n besteht aus n Zügen von I und den n Antwortzügen von II. Es seien $(i_1, j_1), \ldots, (i_n, j_n)$ die *gezogenen Positionen* eines solchen Spiels, wobei i_ν ein Zug in u ist und j_ν einer in v, unabhängig davon, wer gezogen hat.

Spieler II *hat gewonnen*, wenn für $u = u_1 \cdots u_p$ und $v = v_1 \cdots v_q$ mit $u_i, v_j \in \Sigma$ gilt:

- $u_{i_\nu} = v_{j_\nu}$ für $\nu = 1, \ldots, n$, d.h. in jedem Antwortzug ν von II wurde dasselbe Symbol aus Σ gezogen wie durch I.
- $i_\nu < i_{\nu'}$ gdw $j_\nu < j_{\nu'}$, d.h. die relative Lage der Züge ist gleich. Wurde also von I im $(\nu+1)$ten Zug eine Position links von der im νten Zug gewählt, dann wurde auch von II im $(\nu+1)$ten Zug eine Position links von der im νten Zug gewählt.

Sonst hat Spieler II verloren und Spieler I gewonnen.

Für das Verhalten der Spieler bedeutet dies, daß Spieler I versucht, seine Züge so zu machen, daß zwischen u und v unterschieden werden kann, und Spieler II versucht, das zu verhindern.

Beispiel 3.19 Sei $\Sigma = \{a\}$ (d.h. nur die Ordnung zwischen den gezogenen Positionen ist relevant), $n = 3$, $u = a^6$ und $v = a^7$. Die ersten beiden Zugpaare seien folgendermaßen abgelaufen:

	1	2	3	4	5	6	7
$u =$	a	a	a	a	a	a	
		I,2		II,1			
$v =$	a	a	a	a	a	a	a
		II,2		I,1			

Spieler II hat den ersten Zug von I mit Position 4 beantwortet; hätte er stattdessen Position 5 oder 6 genommen, könnte er auf Züge von I in v auf 6 und 7 nämlich nicht mehr entsprechend reagieren.

Den zweiten Zug von Spieler I mußte er mit Position 2 beantworten um seine Gewinnchance zu erhalten. Man sieht leicht, daß nun Spieler II gewinnen kann: Egal, was Spieler I im dritten und letzten Zug macht, Spieler II kann entsprechend reagieren. Allerdings kann Spieler I immer in 3 Zügen siegen: Beginnt I wieder mit Position 4 in v, so muß II diesen mit 3 oder 4 in u beantworten. Macht er dann seinen zweiten Zug in der rechten Hälfte von

v, so kann dieser zwar noch korrekt von II beantwortet werden, aber den dritten könnte er dann immer so ziehen, daß Spieler II verliert, also z.B. (4,4),(6,6),(7,?) oder (4,4),(6,5),(5,?).

Beispiel 3.20 Sei $\Sigma = \{a, b\}$, $n = 3$

	1	2	3	4
$u =$	b	b	a	b
		I,2	I,1	
$v =$	b	a	b	b
	II,2	II,1		

Spieler I zieht im ersten Zug auf Position 3 in u; Spieler II muß auf 2 in v ziehen. Danach zieht I auf 2 in u; Spieler II muß in v auf Position 1 ziehen. Wenn nun I auf Position 1 in u zieht, kann II keinen geeigneten Zug mehr finden und hat verloren.

Definition 3.21 Es sei $n \geq 1$ und $u, v \in \Sigma^+$. Wir sagen, daß Spieler II eine *Gewinnstrategie* für Spiele der Länge n auf u, v hat, wenn II alle möglichen Züge von I stets so beantworten kann, daß II gewinnt.

Um Induktionsargumente einsetzen zu können, betrachtet man auch Spiele, die schon ein Stück angefangen haben: Es seien $u, v \in \Sigma^+$, $\vec{s} = s_1 \cdots s_k \in \{1, \ldots, |u|\}^k$ und $\vec{t} = t_1 \cdots t_k \in \{1, \ldots, |v|\}^k$. Dann beschreibt das Paar $(u, \vec{s})$ und $(v, \vec{t})$ ein Spiel, indem schon k Züge gezogen wurden. Ein Spiel der Länge n darauf ist eine Fortsetzung um n weitere Züge. Spieler II hat dieses Fortsetzungsspiel gewonnen, wenn er das Gesamtspiel gewonnen hat.

Definition 3.22 Es seien $(u, \vec{s})$, $(v, \vec{t})$ mit $u, v \in \Sigma^+$, $\vec{s} \in \{1, \ldots, |u|\}^k$, $\vec{t} \in \{1, \ldots, |v|\}^k$ gegeben.

$(u, \vec{s}) \sim_{k,n} (v, \vec{t})$ gdw Spieler II hat für das Fortsetzungsspiel der Länge n auf $(u, \vec{s})$ und $(v, \vec{t})$ eine Gewinnstrategie.

Für $k = 0$ erweitern wir wieder $\sim_{0,n}$ auf Σ^*, indem wir $\{\epsilon\}$ zu einer $\sim_{k,n}$–Klasse machen.

Lemma 3.23 Die Relation $\sim_{k,n}$ ist eine Äquivalenzrelation.

Beweis: Wir betrachten die drei Bedingungen für eine Äquivalenzrelation.

Reflexivität $(u, \vec{s}) \sim_{k,n} (u, \vec{s})$, weil Spieler II auf die gleiche Position wie I im jeweils anderen Wort ziehen kann und dadurch gewinnt.

Symmetrie Aus $(u, \vec{s}) \sim_{k,n} (v, \vec{t})$ folgt $(v, \vec{t}) \sim_{k,n} (u, \vec{s})$, da $(u, \vec{s})$ und $(v, \vec{t})$ im Spiel symmetrisch verwendet werden.

Transitivität Es ist zu zeigen: Aus $(u, \vec{s}) \sim_{k,n} (v, \vec{t})$ und $(v, \vec{t}) \sim_{k,n} (w, \vec{r})$ folgt $(u, \vec{s}) \sim_{k,n} (w, \vec{r})$. Es sei GS1 die Gewinnstrategie, die $(u, \vec{s}) \sim_{k,n} (v, \vec{t})$ liefert und GS2 diejenige, die $(v, \vec{t}) \sim_{k,n} (w, \vec{r})$ liefert. Die Gewinnstrategie für II auf $(u, \vec{s})$ und $(w, \vec{r})$ sieht wie folgt aus:

- Zieht I in u, so überlegt sich II zunächst mit GS1 seinen entsprechenden Zug in v und wählt auf diesen Zug hin mit GS2 seinen tatsächlichen Zug in w.
- Zieht I in w, wird entsprechend zuerst mit GS2 in v und dann mit GS1 in u gezogen.

Man kann sich leicht überlegen, daß dies tatsächlich eine Gewinnstrategie für II auf $(u, \vec{s})$ und $(w, \vec{r})$ ist. ∎

Für die Relationen $\sim_{k,n}$ kann man nun eine Aussage, die der von Satz 3.16 für $\equiv_{k,n}$ entspricht, sehr leicht zeigen.

Lemma 3.24 Sei $n \geq 0$ und $u, v, u', v' \in \Sigma^*$ und $a \in \Sigma$. Dann folgt aus $u \sim_{0,n} u'$ und $v \sim_{0,n} v'$ auch $(uav, |u| + 1) \sim_{1,n} (u'av', |u'| + 1)$.

Beweis: Wir betrachten zunächst den Fall $u \neq \epsilon \neq v$. Es sei GS1 die Gewinnstrategie für $u \sim_{0,n} u'$ und GS2 für $v \sim_{0,n} v'$. Zieht Spieler I in u (bzw. u'), so antwortet II mit GS1 in u' (bzw. u). Zieht Spieler I in v (bzw. v'), so antwortet II mit GS2 in v' (bzw. v). Zieht I auf $|u| + 1$ in uav (bzw. $|u'| + 1$ in $u'av'$), so antwortet II mit $|u'| + 1$ in $u'av'$ (bzw. $|u| + 1$ in uav). Man sieht leicht, daß dies eine Gewinnstrategie für II liefert. Die Fälle mit $u = \epsilon = u'$ oder $v = \epsilon = v'$ können entsprechend behandelt werden. ∎

Um Satz 3.16 zu beweisen, genügt es nun zu zeigen, daß die Relationen $\sim_{k,n}$ und $\equiv_{k,n}$ übereinstimmen. Wir geben zunächst ein intuitives Argument für diese Übereinstimmung: Offenbar sind die Chancen für II, bei einem Spiel auf u, v zu gewinnen um so größer, je ähnlicher u und v sind. Wenn $u \equiv_n v$ gilt, bedeutet das, daß man u und v nicht durch Formeln der Quantorentiefe n unterscheiden kann. Der Zusammenhang zwischen Quantorentiefe und Anzahl der Züge ist auch einleuchtend: $\exists x.\phi(x)$ sagt aus, daß es eine Position mit bestimmten Eigenschaften gibt. Ein Zug wählt eine Position.

Lemma 3.25 Für alle $n, k \geq 0$ gilt:

$$\sim_{k,n} = \equiv_{k,n},$$

d.h. $(u, \vec{s}) \sim_{k,n} (v, \vec{t})$ gdw $(u, \vec{s}) \equiv_{k,n} (v, \vec{t})$.

Beweis: durch Induktion über n.

Induktionsanfang $n = 0$**:** Es seien $(u, \vec{s})$ und $(v, \vec{t})$ gegeben.

"$\supseteq$" Es gelte also $(u, \vec{s}) \equiv_{k,0} (v, \vec{t})$. Ohne Einschränkung seien die erlaubten freien Variablen aus $\{y_1, \ldots, y_k\}$. Für $i = 1, \ldots, k$ entsprechen s_i, t_i den Variablen y_i. Sei $u = a_1 \cdots a_p$ und $v = b_1 \cdots b_q$. Für alle atomaren Formeln $y_i \dot{<} y_j$, $y_i \doteq y_j$ und $Q_a(y_i)$ gilt wegen $(u, \vec{s}) \equiv_{k,0} (v, \vec{t})$

$$\left.\begin{array}{lcl lcl} (u,\vec{s}) & \models & y_i \dot{<} y_j & \text{gdw} & (v,\vec{t}) & \models & y_i \dot{<} y_j \\ (u,\vec{s}) & \models & y_i \doteq y_j & \text{gdw} & (v,\vec{t}) & \models & y_i \doteq y_j \\ (u,\vec{s}) & \models & Q_a(y_i) & \text{gdw} & (v,\vec{t}) & \models & Q_a(y_i) \end{array}\right\} (*)$$

Das heißt aber

$$\left.\begin{array}{lcl} s_i < s_j & \text{gdw} & t_i < t_j \\ s_i = s_j & \text{gdw} & t_i = t_j \\ a_{s_i} = a & \text{gdw} & b_{t_i} = a \end{array}\right\} (**)$$

Dies bedeutet, daß II das Spiel (mit 0 zusätzlichen Zügen) gewonnen hat.

"$\subseteq$" Aus $(u, \vec{s}) \sim_{k,0} (v, \vec{t})$ folgt $(**)$ (die mittlere Zeile folgt aus der ersten, da $<$ eine totale Ordnung ist). Damit gilt auch $(*)$. Also erfüllen $(u, \vec{s})$ und $(v, \vec{t})$ dieselben atomaren Formeln und damit auch dieselben Booleschen Kombinationen dieser atomaren Formeln.

Induktionsschritt $n \to n+1$**:**

"$\subseteq$" Es gelte also $(u, \vec{s}) \sim_{k,n+1} (v, \vec{t})$. Es sei $\phi(y_1, \ldots, y_k) \in L_{k,n+1}$ mit $(u, \vec{s}) \models \phi$ gegeben. Es genügt zu zeigen, daß daraus $(v, \vec{t}) \models \phi$ folgt. Da Boolesche Operatoren unproblematisch sind, genügt es, Formeln ϕ der Form $\phi = \exists y_{k+1}.\psi(y_1, \ldots, y_k, y_{k+1})$ für $\psi \in L_{k+1,n}$ zu betrachten.

Wegen $(u, \vec{s}) \sim_{k,n+1} (v, \vec{t})$ kann II den ersten Zug von I geeignet beantworten. Daher gibt es zu jedem $s_{k+1} \in \{1, \ldots, |u|\}$ ein $t_{k+1} \in \{1, \ldots, |v|\}$ mit

$$(u, \vec{s}s_{k+1}) \sim_{k+1,n} (v, \vec{t}t_{k+1}).$$

Induktion liefert, daß es für alle s_{k+1} ein t_{k+1} gibt mit

$$(u, \vec{s}s_{k+1}) \equiv_{k+1,n} (v, \vec{t}t_{k+1}). \qquad (*)$$

Wegen $(u, \vec{s}) \models \exists y_{k+1}.\psi(y_1, \ldots, y_k, y_{k+1})$ gibt es ein $s_{k+1} \in \{1, \ldots, |u|\}$ mit

$$(u, \vec{s}s_{k+1}) \models \psi(y_1, \ldots, y_k, y_{k+1}). \qquad (**)$$

Da $\psi \in L_{k+1,n}$ können wir $(*)$ auf $(**)$ anwenden, d.h. es gibt ein t_{k+1}, so daß gilt:

$$(v, \vec{t}t_{k+1}) \models \psi(y_1, \ldots, y_k, y_{k+1}).$$

Daraus folgt nun

$$(v, \vec{t}) \models \exists y_{k+1}.\psi(y_1, \ldots, y_k, y_{k+1}).$$

"$\supseteq$" Es gelte $(u, \vec{s}) \not\sim_{k,n+1} (v, \vec{t})$. Wir müssen $(u, \vec{s}) \not\equiv_{k,n+1} (v, \vec{t})$ zeigen. Gesucht ist also eine Formel $\phi \in L_{k,n+1}$, die von einem der beiden Tupel $(u, \vec{s})$, $(v, \vec{t})$ erfüllt wird, aber nicht von dem anderen.

Wegen $(u, \vec{s}) \not\sim_{k,n+1} (v, \vec{t})$ gibt es einen ersten Zug von I, den II nicht so beantworten kann, daß II danach eine Gewinnstrategie hat. Ohne Einschränkung sei dieser von I in u (der andere Fall kann entsprechend behandelt werden). Es gibt also ein $s_{k+1} \in \{1, \ldots, |u|\}$, so daß für alle $t_{k+1} \in \{1, \ldots, |v|\}$ gilt:

$$(u, \vec{s}s_{k+1}) \not\sim_{k+1,n} (v, \vec{t}t_{k+1}).$$

Mit der Induktionsvoraussetzung gilt

$$(u, \vec{s}s_{k+1}) \not\equiv_{k+1,n} (v, \vec{t}t_{k+1}).$$

Für alle $t_{k+1} \in \{1, \ldots, |v|\}$ gibt es daher eine Formel $\psi_{t_{k+1}}(y_1, \ldots, y_{k+1}) \in L_{k+1,n}$ mit

$$(u, \vec{s}s_{k+1}) \models \psi_{t_{k+1}} \quad \text{und} \tag{$*$}$$
$$(v, \vec{t}t_{k+1}) \not\models \psi_{t_{k+1}}. \tag{$**$}$$

Aus $(*)$ folgt, daß für $\phi_{t_{k+1}} = \exists y_{k+1}.\psi_{t_{k+1}}$ gilt $(u, \vec{s}) \models \phi_{t_{k+1}}$, wobei offenbar $\phi_{t_{k+1}} \in L_{k,n+1}$. Aus $(**)$ kann man aber leider *nicht* folgern, daß $(v, \vec{t}) \not\models \phi_{t_{k+1}}$. Wir wissen nur, daß t_{k+1} nicht für y_{k+1} in $\psi_{t_{k+1}}$ eingesetzt werden kann. Über andere $t \in \{1, \ldots, |v|\}$ ist aber nichts bekannt.

Wir betrachten daher statt $\phi_{t_{k+1}}$ die Formel

$$\phi := \exists y_{k+1}. \bigwedge_{1 \le t \le |v|} \psi_t. \tag{3.2}$$

Wegen $(*)$ folgt nun $(u, \vec{s}) \models \phi$, und wegen $(**)$ gibt es für jedes t ein Konjunkt, das nicht erfüllt ist, also $(v, \vec{t}) \not\models \phi$. Da $\phi \in L_{k,n+1}$, folgt $(u, \vec{s}) \not\equiv_{k,n+1} (v, \vec{t})$. ∎

Beachte: Lemma 3.25 gilt auch für unendliche Wörter. Um das zu beweisen, muß aber die Konjunktion in Gleichung 3.2 trotz unendlicher Länge von v endlich sein. Dazu kann man Satz 3.12 verwenden, der ja alle Formeln mit Quantorentiefe n und k freien Variablen auf eine endliche Menge äquivalenter Formeln reduziert.

Mit Lemma 3.25 und Lemma 3.24 ist nun Satz 3.16 bewiesen, und damit ist der Beweis des Satzes 3.9 abgeschlossen: Die sternfreien Sprachen sind also genau die durch Logik erster Stufe[3] beschreibbaren Sprachen.

[3]Wie immer in diesem Kapitel meinen wir damit Logik erster Stufe mit Gleichheit über den nicht–logischen Symbolen $\dot{<}$ (ein zweistelliges Relationssymbol) und $P_1, \ldots, P_k$ (einstellige Relationssymbole).

Da nicht alle regulären Sprachen sternfrei sind, zeigt dies, daß es reguläre Sprachen gibt, die nicht durch Logik erster Stufe beschrieben werden können. Ein Beispiel für eine solche Sprache wurde bereits erwähnt: $a(aa)^*$. Wenn man zusätzlich noch Quantifizierung über einstellige Prädikate zuläßt, kann man alle regulären Sprachen beschreiben (siehe Beispiel 1.28). Wir zeigen diesen Zusammenhang hier nicht, da wir später den entsprechenden für Sprachen unendlicher Wörter zeigen werden.

Aus der Sicht der Logik ist Satz 3.9 interessant, da er ein Entscheidbarkeitsresultat für die Theorie der totalen Ordnungen $\mathcal{TOT}$ liefert. $\mathcal{TOT}$ beschreibt eine transitive, irreflexive, totale Ordnung $<$

$$\begin{aligned}\mathcal{TOT} \quad = \quad & \{\ \forall x.\forall y.\forall z.\, x \dot{<} y \wedge y \dot{<} z \Rightarrow x \dot{<} z, \\ & \ \ \forall x.\neg(x \dot{<} x), \\ & \ \ \forall x.\forall y.\, x \dot{<} y \vee x \doteq y \vee y \dot{<} x\ \}\end{aligned}$$

Satz 3.26 Sei ϕ eine geschlossene Formel der Logik erster Stufe mit Gleichheit, die höchstens die nicht–logischen Symbole $\dot{<}, P_1, \ldots P_n$ enthält (wobei P_i einstellige Prädikatsymbole sind). Dann ist es entscheidbar, ob $\mathcal{TOT} \cup \{\phi\}$ ein endliches Modell hat.

Beweis: Wir haben gesehen, daß $L(\phi)$ eine sternfreie Sprache ist. Der Beweis von Satz 3.9 ist konstruktiv, d.h. man kann zu ϕ effektiv eine Beschreibung durch einen sternfreien Ausdruck (der Boolesche Operatoren und Konkatenation verwendet) für $L(\phi)$ angeben. Das liegt im wesentlichen daran, daß man die endlichen Mengen $\Gamma_{k,n}$ in Satz 3.12 effektiv berechnen kann und dadurch die Relationen $\equiv_{k,n}$ entscheidbar werden.

$\mathcal{TOT} \cup \{\phi\}$ hat ein endliches Modell genau dann, wenn die Sprache $L(\phi)$ nicht leer ist ($\mathcal{TOT}$ beschreibt ja gerade die von uns vorausgesetzte Ordnung $<$ auf dem Interpretationsbereich). Man kann den sternfreien Ausdruck effektiv in einen regulären Ausdruck umwandeln, und für reguläre Ausdrücke ist entscheidbar, ob sie die leere Sprache beschreiben oder nicht. ∎

Häufig möchte man aber auch wissen, ob eine solche Formel ein Modell hat, bei dem $\dot{<}$ als die gewöhnliche Ordnung auf den natürlichen Zahlen interpretiert wird (also ob sie insbesondere ein unendliches Modell hat). Daher werden wir im folgenden Kapitel Teilmengen unendlicher Wörter und Automaten auf unendlichen Wörtern betrachten.

Kapitel 4

Unendliche Wörter und Büchi–Automaten

Bevor wir unendliche Wörter formal einführen, betrachten wir nochmals endliche Wörter. Es sei Σ ein endliches Alphabet. Ein endliches Wort $u \in \Sigma^+$ ist eine Folge $u = a_0 a_1 a_2 \cdots a_k$ von Elementen $a_i \in \Sigma$. Man kann daher u auffassen als eine Abbildung $u : \{0, \ldots, k\} \to \Sigma$ mit $u(i) = a_i$. Das heißt also: *endliche Wörter* über Σ sind Abbildungen von einem Anfangssegment $\{0, \ldots, k\}$ der natürlichen Zahlen nach Σ. Da wir uns bei den natürlichen Zahlen nur für die Ordnung interessieren und nicht für arithmetische Operationen, schreiben wir dafür ω (= Ordnungstyp der natürlichen Zahlen, siehe [Ros82], 1. Kapitel).

Definition 4.1

1. Ein *unendliches Wort über* Σ ist eine Abbildung $\alpha : \omega \to \Sigma$.

2. Mit Σ^ω bezeichnen wir die *Menge aller unendlichen Wörter* über Σ.

3. Mengen unendlicher Wörter heißen ω*–Sprachen.*

Wir schreiben derartige Wörter häufig als $\alpha = a_0 a_1 a_2 a_3 \cdots$ wobei $\alpha(i) = a_i$.

Beispiel 4.2 Für $\Sigma = \{a, b\}$ können wir α mit

$$\alpha(i) = \begin{cases} a & \text{falls } i \text{ ungerade} \\ b & \text{falls } i \text{ gerade} \end{cases}$$

als $\alpha = ababab\cdots$ schreiben.

Wir betrachten nun einige Operationen für unendliche Wörter und Mengen von unendlichen Wörtern.

Segmente: Für $\alpha : \omega \to \Sigma$ bezeichne

- $\alpha(m, n)$ das endliche Wort $\alpha(m)\cdots\alpha(n)$ $(m \leq n)$,
- $\alpha(m, \omega)$ das unendliche Wort $\alpha(m)\alpha(m+1)\cdots$.

Konkatenation: Die Konkatenation eines endlichen mit einem unendlichen Wort: Ist $w = a_0 \cdots a_m$ ein endliches Wort und $\alpha = \alpha(0)\alpha(1)\alpha(2)\cdots$ ein unendliches Wort, so ist $w \cdot \alpha$ das unendliche Wort

$$a_0 \cdots a_m \alpha(0)\alpha(1)\alpha(2)\cdots .$$

Beachte: Ein unendliches Wort auf der linken Seite des Konkatenationssymbols macht keinen Sinn. Wie üblich wird Konkatenation auch auf Mengen von Wörtern erweitert.

Unendliche Iteration: Es sei $L \subseteq \Sigma^*$ eine Menge endlicher Wörter:

$$L^\omega := \{\alpha \in \Sigma^\omega \mid \alpha = w_0 w_1 w_2 \cdots \text{ mit } w_i \in L \setminus \{\epsilon\}\}\,.$$

Beispiel: $L = \{ab\}$: $L^\omega = \{abababab\cdots\}$. Wir schreiben dann häufig $L^\omega = (ab)^\omega$.

Limes: Es sei $L \subseteq \Sigma^*$ eine Menge endlicher Wörter.

$$\lim L := \{\alpha \in \Sigma^\omega \mid \text{ es gibt unendlich viele } n \text{ mit } \alpha(0, n) \in L\}.$$

Beispiel 4.3 Im folgenden sei $\Sigma = \{a, b\}$.

1. Sei $L = a^*b$, dann ist $\lim L = \emptyset$, da ein unendliches Wort höchstens ein Element von L als Anfangssegment haben kann. Dies liegt daran, daß L ein Präfix–Code ist, d.h. kein Wort in L ist Anfangsstück eines anderen Wortes in L.

2. Sei $L = ba^*$, dann ist $\lim L = \{baaa\cdots\}$.

3. Sei $L = (a^*bb^*)^*$, dann ist $\lim L = \{\alpha \in \Sigma^\omega \mid$ nach jedem Vorkommen von a in α kommt noch ein $b\}$.

4.1 Büchi–Automaten und ω–reguläre Sprachen

Als Automaten zum Akzeptieren von Mengen unendlicher Wörter (ω–Sprachen) betrachten wir "normale" endliche Automaten. Der Unterschied liegt in der Akzeptanzbedingung.

Definition 4.4 Ein *Büchi–Automat* ist ein (nicht–deterministischer) endlicher Automat $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$, d.h.

- Q ist eine nicht–leere, endliche Menge,
- Σ ist ein endliches Alphabet,
- $I \subseteq Q$ ist eine Menge von Anfangszuständen,
- $\Delta \subseteq Q \times \Sigma \times Q$ ist die Übergangsrelation,
- $F \subseteq Q$ ist eine Menge von Endzuständen.

Da wir uns jetzt für unendliche Wörter interessieren, betrachten wir *unendliche Pfade* im Automaten: $q_0 \xrightarrow{a_0}_{\mathcal{A}} q_1 \xrightarrow{a_1}_{\mathcal{A}} q_2 \xrightarrow{a_2}_{\mathcal{A}} q_3 \xrightarrow{a_3}_{\mathcal{A}} q_4 \cdots$. Die Beschriftung eines unendlichen Pfades ist ein unendliches Wort $a_0a_1a_2a_3\cdots$. Da es bei unendlichen Pfaden keinen letzten Zustand gibt, müssen erfolgreiche Pfade anders definiert werden als in Definition 1.2. Der unendliche Pfad $q_0 \xrightarrow{a_0}_{\mathcal{A}} q_1 \xrightarrow{a_1}_{\mathcal{A}} q_2 \xrightarrow{a_2}_{\mathcal{A}} q_3 \xrightarrow{a_3}_{\mathcal{A}} \cdots$ heißt *erfolgreich*, falls

1. $q_0 \in I$ und
2. es gibt unendlich viele i mit $q_i \in F$.

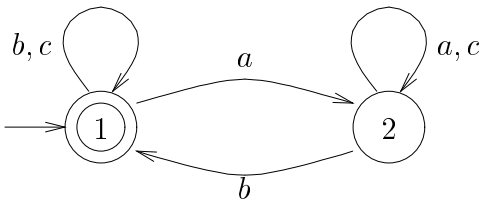
Die *von $\mathcal{A}$ akzeptierte ω–Sprache* ist

$$L_\omega(\mathcal{A}) := \{\alpha \in \Sigma^\omega \mid \alpha \text{ ist Beschriftung eines erfolgreichen Pfades}\}.$$

Solche Sprachen nennen wir *Büchi–erkennbar*.

Beispiel 4.5 Im folgenden sei $\Sigma = \{a, b, c\}$.

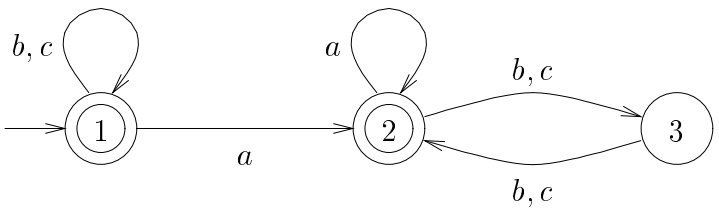
1. $\mathcal{A}_1$:



$$L_\omega(\mathcal{A}_1) = \{\alpha \in \Sigma^\omega \mid \text{nach jedem } a \text{ in } \alpha \text{ kommt noch ein } b\},$$

denn a führt in Zustand 2 über und der akzeptierende Zustand 1 wird nur durch Lesen eines b wieder erreicht: Wenn nach irgendeinem a kein b mehr auftritt, kann Zustand 1 nur endlich oft durchlaufen werden.

2. $\mathcal{A}_2$:



$$L_\omega(\mathcal{A}_2) = \{\alpha \in \Sigma^\omega \mid \text{zwischen zwei } a \text{ in } \alpha \text{ befindet sich eine gerade Anzahl von } b \text{ und } c\}$$

Um die von Büchi–Automaten akzeptierten Sprachen genauer zu untersuchen, verwenden wir die folgende Abkürzung: Es sei $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ein Büchi–Automat und $p, q \in Q$. Dann ist

$$L_{p,q} := \{w \in \Sigma^* \mid w \text{ ist Beschriftung eines (endlichen) Pfades von } p \text{ nach } q\}.$$

Offenbar sind die Sprachen $L_{p,q}$ regulär. Im folgenden zeigen wir, wie sich eine von einem Büchi–Automaten akzeptierte Sprache mit Hilfe dieser regulären Sprachen $L_{p,q}$ darstellen läßt.

Lemma 4.6 Sei $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ein Büchi–Automat. Dann gilt

$$L_\omega(\mathcal{A}) = \bigcup_{i \in I, f \in F} L_{i,f} \cdot L_{f,f}^\omega.$$

Beweis:

"$\subseteq$" Es sei $\alpha = a_0 a_1 a_2 a_3 \cdots \in L_\omega(\mathcal{A})$. Nach Definition gibt es einen unendlichen Pfad $q_0 \xrightarrow{a_0}_{\mathcal{A}} q_1 \xrightarrow{a_1}_{\mathcal{A}} q_2 \xrightarrow{a_2}_{\mathcal{A}} q_3 \xrightarrow{a_3}_{\mathcal{A}} \cdots$ mit $q_0 \in I$ und unendlich vielen i mit $q_i \in F$. Da F endlich ist, gibt es ein $f \in F$ und $i_1 < i_2 < i_3 < \cdots$ mit $q_{i_v} = f$. Es gilt somit $a_0 \cdots a_{i_1} \in L_{q_0,f}$ und $a_{i_v+1} \cdots a_{i_{v+1}} \in L_{f,f}$, d.h. $\alpha \in L_{i,f} \cdot L_{f,f}^\omega$.

"$\supseteq$" Es sei $\alpha = w_0 w_1 w_2 w_3 \cdots$ mit $w_0 \in L_{i,f}$, $w_i \in L_{f,f} \setminus \{\epsilon\}$ $(i \geq 1)$. Damit ist der Pfad $i \xrightarrow{w_0}_{\mathcal{A}} f \xrightarrow{w_1}_{\mathcal{A}} f \xrightarrow{w_2}_{\mathcal{A}} f \xrightarrow{w_3}_{\mathcal{A}} \cdots$ ein unendlicher, erfolgreicher Pfad und $\alpha \in L_\omega(\mathcal{A})$. ∎

Das nächste Lemma beschreibt Abschlußeigenschaften Büchi–erkennbarer Sprachen.

Lemma 4.7

1. Ist $U \subseteq \Sigma^*$ regulär, so ist U^ω Büchi–erkennbar.
2. Ist $U \subseteq \Sigma^*$ regulär und $L \subseteq \Sigma^\omega$ Büchi–erkennbar, dann ist auch $U \cdot L$ Büchi–erkennbar.
3. Sind L_1, L_2 Büchi–erkennbar, so sind auch $L_1 \cup L_2$ und $L_1 \cap L_2$ Büchi–erkennbar.

Beweis:

1. Wir wissen: Ist U regulär, so auch $U \setminus \{\epsilon\}$. Zur Erinnerung:

$$U^\omega = \{\alpha \in \Sigma^\omega \mid \alpha = u_0 u_1 u_2 u_3 \cdots \text{ und } u_i \in U \setminus \{\epsilon\}\}$$

Wir werden nun einen endlichen Automaten für U etwas modifizieren und dann daraus einen Büchi–Automaten $\mathcal{B}$ bauen, der U^ω akzeptiert.

Sei also $\mathcal{A} = (\Sigma, Q, I, \Delta, F)$ ein endlicher Automat für U. Dann definieren wir zunächst $\mathcal{A}'$, der $U \setminus \{\epsilon\}$ akzeptiert und genau einen Anfangszustand hat, der zudem von keinem anderen Zustand mehr erreicht werden kann: Es sei $q_0 \notin Q$, definiere

$$\begin{aligned} \mathcal{A}' &:= (\Sigma, Q \cup \{q_0\}, \{q_0\}, \Delta', F), \qquad \text{wobei} \\ \Delta' &:= \Delta \cup \{(q_0, a, q) \mid \exists i \in I : (i, a, q) \in \Delta\}. \end{aligned}$$

Den Büchi–Automaten $\mathcal{B}$ für U^ω definieren wir dann wie folgt:

$$\begin{aligned} \mathcal{B} &:= (\Sigma, Q \cup \{q_0\}, \{q_0\}, \Delta'', \{q_0\}), \qquad \text{wobei} \\ \Delta'' &:= \Delta' \cup \{(q, a, q_0) \mid \exists f \in F : (q, a, f) \in \Delta'\}. \end{aligned}$$

Der Nachweis, daß $L_\omega(\mathcal{B}) = U^\omega$ gilt, bietet sich als Übung an und wird hier nicht geführt.

2. Sei $\mathcal{A}$ ein endlicher Automat für U und $\mathcal{B}$ ein Büchi–Automat für L:

$$\begin{aligned} \mathcal{A} &= (Q_1, \Sigma, I_1, \Delta_1, F_1), \\ \mathcal{B} &= (Q_2, \Sigma, I_2, \Delta_2, F_2). \end{aligned}$$

Ohne Einschränkung sei $Q_1 \cap Q_2 = \emptyset$. Ein Automat $\mathcal{C}$ mit $L_\omega(\mathcal{C}) = U \cdot L$ ist dann definiert durch

$$\begin{aligned} \mathcal{C} &= (Q_1 \cup Q_2, \Sigma, I', \Delta', F_2), \\ \Delta' &= \Delta_1 \cup \Delta_2 \cup \\ &\quad \{(q, a, q') \mid \text{Es gibt } f \in F_1 \text{ mit } (q, a, f) \in \Delta_1 \text{ und } q' \in I_2\} \\ I' &= I_1 \cup \begin{cases} \emptyset & \text{falls } I_1 \cap F_1 = \emptyset, \text{ d.h. } \epsilon \notin L(\mathcal{A}) \\ I_2 & \text{sonst.} \end{cases} \end{aligned}$$

Man sieht leicht: $L_\omega(\mathcal{C}) = U \cdot L$.

3. Sei $\mathcal{A}$ ein Büchi–Automat für L_1 und $\mathcal{B}$ ein Büchi–Automat für L_2, mit

$$\begin{aligned} \mathcal{A} &= (Q_1, \Sigma, I_1, \Delta_1, F_1), \\ \mathcal{B} &= (Q_2, \Sigma, I_2, \Delta_2, F_2), \end{aligned}$$

wobei wir ohne Einschränkung davon ausgehen, daß $Q_1 \cap Q_2 = \emptyset$ ist.

Dann ist $\mathcal{C} = (Q_1 \cup Q_2, \Sigma, I_1 \cup I_2, \Delta_1 \cup \Delta_2, F_1 \cup F_2)$ ein (sicher nicht deterministischer) Büchi–Automat für $L_1 \cup L_2$.

Ein Büchi–Automat, der $L_1 \cap L_2$ akzeptiert, ist etwas aufwendiger: Sei

$$\mathcal{D} = (Q_1 \times Q_2 \times \{0,1,2\}, \Sigma, I_1 \times I_2 \times \{0\}, \Delta, F)$$

und

$$\begin{aligned} \Delta \quad = \{((q_1,q_2,i),a,(p_1,p_2,j)) \mid \quad & (q_1,a,p_1) \in \Delta_1,\ (q_2,a,p_2) \in \Delta_2 \text{ und} \\ & j = 1 \quad \text{falls } i = 0 \text{ und} \quad p_1 \in F_1, \\ & j = 2 \quad \text{falls } i = 1 \text{ und} \quad p_2 \in F_2, \\ & j = 0 \quad \text{falls } i = 2 \\ & j = i \quad \text{sonst.}\} \end{aligned}$$

Das heißt, man beginnt in der dritten Komponente mit 0. Wenn man das erste Mal ein $f \in F_1$ durchläuft, wird die dritte Komponente auf 1 gesetzt. Wenn man dann ein $f' \in F_2$ durchläuft, erhöht man die dritte Komponente auf 2 und setzt sie direkt im nächsten Schritt wieder auf 0. Durchläuft man nun unendlich oft Elemente aus F_1 in der ersten Komponente und unendlich oft Elemente aus F_2 in der zweiten, so erreicht man auch unendlich oft einen Zustand mit 2 in der dritten Komponente. Deshalb definieren wir $F := Q_1 \times Q_2 \times \{2\}$. ∎

Der nächste Satz gibt eine wichtige Charakterisierung Büchi–erkennbarer Sprachen.

Satz 4.8 [Büchi, 1962]

1. Eine ω–Sprache $L \subseteq \Sigma^\omega$ ist Büchi–erkennbar genau dann, wenn es reguläre Sprachen $U_1, \ldots U_n, V_1, \ldots, V_n \subseteq \Sigma^*$ gibt mit

$$L = \bigcup_{i=1}^{n} U_i \cdot V_i^\omega.$$

2. Man kann dabei ohne Einschränkung davon ausgehen, daß $V_i \cdot V_i \subseteq V_i$ gilt.

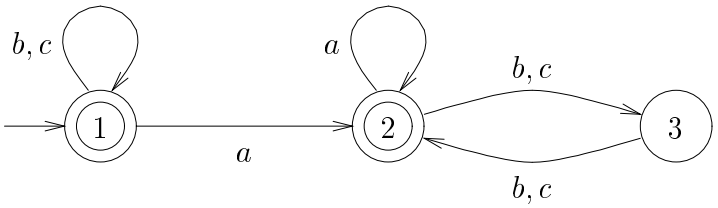
Beweis:

"$\rightsquigarrow$" Folgt für Teil 1 und 2 unmittelbar aus Lemma 4.6.

"$\leftsquigarrow$" Folgt aus 4.7. ∎

Wegen dieses engen Zusammenhanges mit regulären Sprachen nennt man Büchi–erkennbare Sprachen auch *ω–regulär*. Sind die Sprachen U_i, V_i durch reguläre Ausdrücke gegeben, so nennt man die Darstellung $L = \bigcup_{i=1}^{n} U_i \cdot V_i^\omega$ der ω–regulären Sprache L einen *ω–regulären Ausdruck für L*.

Beispiel 4.9 Betrachten wir den Automaten $\mathcal{A}_2$ aus Beispiel 4.5:



$$
\begin{aligned}
L_{1,1} &= (b \cup c)^*, \\
L_{1,2} &= (b \cup c)^* \cdot a \cdot L_{2,2}, \\
L_{2,2} &= (a \cup ((b \cup c) \cdot (b \cup c)))^*,
\end{aligned}
$$

$$
\begin{aligned}
L_\omega(\mathcal{A}_2) &= L_{1,1} \cdot L_{1,1}^\omega \cup L_{1,2} \cdot L_{2,2}^\omega \\
&= L_{1,1}^\omega \cup L_{1,2} \cdot L_{2,2}^\omega \\
&= ((b \cup c)^*)^\omega \cup (b \cup c)^* \cdot a \cdot L_{2,2} \cdot L_{2,2}^\omega \\
&\quad \text{offenbar gilt } (U^*)^\omega = U^\omega, \text{ und } U \cdot U^\omega = U^\omega \text{ daher:} \\
&= (b \cup c)^\omega \cup (b \cup c)^* \cdot a \cdot L_{2,2}^\omega \\
&= (b \cup c)^\omega \cup (b \cup c)^* \cdot a \cdot ((a \cup (b \cup c) \cdot (b \cup c))^*)^\omega \\
&= (b \cup c)^\omega \cup (b \cup c)^* \cdot a \cdot (a \cup (b \cup c) \cdot (b \cup c))^\omega.
\end{aligned}
$$

Eine weitere Konsequenz aus Lemma 4.6 ist, daß man entscheiden kann, ob ein Büchi–Automat eine nicht–leere Sprache akzeptiert: Dies ist genau dann der Fall, wenn es ein $i \in I$ und ein $f \in F$ gibt mit $L_{i,f} \neq \emptyset$ und $L_{f,f} \setminus \{\epsilon\} \neq \emptyset$. Da $L_{i,f}, L_{f,f} \setminus \{\epsilon\}$ reguläre Sprachen sind, die durch den gegebenen Automaten (bei passender Wahl der Anfangs– und Endzustände) akzeptiert werden, ist deren Leerheit entscheidbar.

Satz 4.10

1. Das Leerheitsproblem für Büchi–Automaten ist entscheidbar.
2. Ist L eine Büchi–erkennbare ω–Sprache, so enthält L ein schließlich periodisches Wort.

Beweis:

1. Ist bereits gezeigt, siehe oben.
2. Ist $L_{i,f} \neq \emptyset$ und $L_{f,f} \setminus \{\epsilon\} \neq \emptyset$ mit $i \in I$ und $f \in F$, so gibt es ein $u \in L_{i,f}$ und ein $v \in L_{f,f} \setminus \{\epsilon\}$ und es gilt

$$\alpha = uvvv \cdots \in L_{i,f} \cdot (L_{f,f} \setminus \{\epsilon\})^\omega \subseteq L.$$

   Das unendliche Wort α leistet dann das Gewünschte. ∎

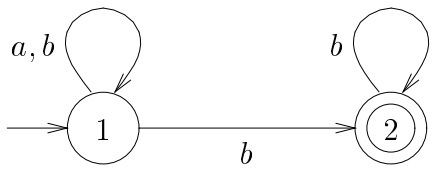
Für reguläre Sprachen kann man das Äquivalenzproblem ($L(\mathcal{A}_1) = L(\mathcal{A}_2)$?) auf das Leerheitsproblem reduzieren, da $L_1 = L_2$ genau dann gilt, wenn $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$ gilt und da reguläre Sprachen unter Booleschen Operatoren abgeschlossen sind. Bei ω–reguläre Sprachen fehlt uns für diese Reduktion allerdings noch der Abschluß unter Komplement. Schauen wir uns deshalb an, wie wir diese Abschlußeigenschaft für reguläre Sprachen L gezeigt haben:

1. Durch die Potenzmengenkonstruktion auf einem endlichen Automat für L erhalten wir einen deterministischen (und vollständigen) Automaten, der immer noch L erkennt.

2. Durch Vertauschen von Endzuständen mit Nicht–Endzuständen erhalten wir einen Automaten, der $\overline{L}$ erkennt.

Bei Büchi–Automaten funktioniert weder Punkt 1 noch Punkt 2. Dazu ein Beispiel einer ω–Sprache, die zwar von einem Büchi–Automaten, aber nicht von einem deterministischen Büchi–Automaten akzeptiert wird:

Beispiel 4.11 Sei $\Sigma = \{a, b\}$. Der folgende nicht–deterministische Büchi–Automat akzeptiert die Sprache $(a \cup b)^* b^\omega$.



Behauptung: $(a \cup b)^* b^\omega$ kann nicht von einem deterministischen Automaten erkannt werden.

Beweis: Angenommen, $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ist ein deterministischer Büchi–Automat mit $L_\omega(\mathcal{A}) = (a \cup b)^* b^\omega = L$. Also ist $\delta : Q \times \Sigma \to Q$ eine Funktion. Da $ab^\omega \in L$ gilt, gibt es ein $k_1 > 0$ und $f_1 \in F$ mit

$$q_0 \xrightarrow{ab^{k_1}}_{\mathcal{A}} f_1.$$

Nun ist aber auch $ab^{k_1}ab^\omega \in L$ und $\mathcal{A}$ deterministisch. Deshalb gibt es $k_2 > 0$ und $f_2 \in F$ mit

$$q_0 \xrightarrow{ab^{k_1}}_{\mathcal{A}} f_1 \xrightarrow{ab^{k_2}}_{\mathcal{A}} f_2.$$

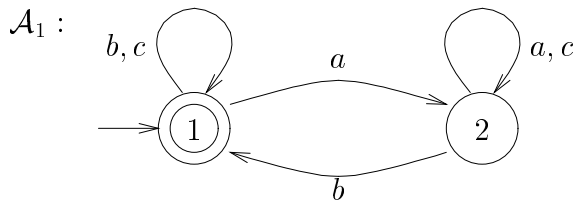
Iteriert man dieses Argument, so erhält man $k_1, k_2, k_3, \ldots \in \mathbb{N}$ und $f_1, f_2, f_3, \ldots \in F$ mit

$$q_0 \xrightarrow{ab^{k_1}}_{\mathcal{A}} f_1 \xrightarrow{ab^{k_2}}_{\mathcal{A}} f_2 \xrightarrow{ab^{k_3}}_{\mathcal{A}} f_3 \xrightarrow{ab^{k_4}}_{\mathcal{A}} f_4 \cdots .$$

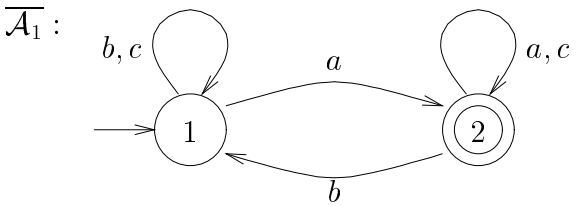
Damit ist dann aber auch $\alpha = ab^{k_1}ab^{k_2}ab^{k_3}ab^{k_4} \cdots \in L_\omega(\mathcal{A})$, aber $\alpha \notin L$ im Widerspruch zur Annahme, daß $L_\omega(\mathcal{A}) = L$ gilt. ∎

Das nächste Beispiel zeigt, daß auch bei deterministischen Büchi–Automaten das Vertauschen von End– mit Nicht–Endzuständen nicht dem Komplementieren der ω–Sprachen entspricht.

Beispiel 4.12 Betrachte $\mathcal{A}_1$ aus Beispiel 4.5.



$\mathcal{A}_1$ ist deterministisch. $L_\omega(\mathcal{A}_1) = \{\alpha \mid$ nach jedem a in α kommt noch ein $b\}$. $\overline{\mathcal{A}_1}$ entsteht aus $\mathcal{A}_1$ durch Vertauschen von Endzuständen und Nicht–Endzuständen.



$L_\omega(\overline{\mathcal{A}_1}) \cap L_\omega(\mathcal{A}_1)$ ist nicht leer; z.B. ist $(ab)^\omega \in L_\omega(\overline{\mathcal{A}_1}) \cap L_\omega(\mathcal{A}_1)$.

Wie Beispiel 4.11 zeigt, ist die Klasse der von *deterministischen* Büchi–Automaten akzeptierten Sprachen kleiner als die Klasse der ω–regulären Sprachen (also die Sprachen, die von nicht–deterministischen Büchi–Automaten akzeptiert werden). Der nächste Satz charakterisiert diese Klasse.

Satz 4.13 Für $L \subseteq \Sigma^\omega$ sind äquivalent:

1. L wird von einem deterministischen Büchi–Automaten akzeptiert.
2. $L = \lim U$ für eine reguläre Sprache $U \subseteq \Sigma^*$.

Beweis: Es sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein deterministischer endlicher Automat. Für $\alpha \in \Sigma^\omega$ sind äquivalent:

i) $\alpha \in L_\omega(\mathcal{A})$.

ii) Es gibt unendlich viele verschiedene Anfangsstücke von α, die in $L(\mathcal{A})$ liegen, d.h. $\alpha \in \lim L(\mathcal{A})$.

Denn:

i)$\rightsquigarrow$ii) Sei $\alpha \in L_\omega(\mathcal{A})$. Dann gibt es $f_1, f_2, \ldots \in F$ und $u_0, u_1, u_2, \ldots \in \Sigma^+$ mit $q_0 \xrightarrow{u_0}_{\mathcal{A}} f_1 \xrightarrow{u_1}_{\mathcal{A}} f_2 \xrightarrow{u_2}_{\mathcal{A}} f_3 \xrightarrow{u_3}_{\mathcal{A}} \cdots$. Es folgt

$$u_0, u_0u_1, u_0u_1u_2, u_0u_1u_2u_3, \ldots \in L(\mathcal{A}).$$

ii)$\rightsquigarrow$i) Es sei $\{u_0, u_0u_1, u_0u_1u_2, u_0u_1u_2u_3, \ldots\} \subseteq \Sigma^+, u_i \in \Sigma^+$ die unendliche Menge der Anfangsstücke von α mit $u_0 \cdots u_i \in L(\mathcal{A})$. Da $\mathcal{A}$ deterministisch ist, heißt das, daß es $f_1, f_2, \ldots \in F$ gibt mit $q_0 \xrightarrow{u_0}_{\mathcal{A}} f_1 \xrightarrow{u_1}_{\mathcal{A}} f_2 \xrightarrow{u_2}_{\mathcal{A}} f_3 \xrightarrow{u_3}_{\mathcal{A}} \cdots$. Daher ist $\alpha = u_0u_1u_2u_3 \cdots \in L_\omega(\mathcal{A})$.

Beachte: Wäre $\mathcal{A}$ nicht deterministisch, so könnte es zwei verschiedene Pfade mit Beschriftung u_0 geben: $q_0 \xrightarrow{u_0}_{\mathcal{A}} f_1$ und $q_0 \xrightarrow{u_0}_{\mathcal{A}} q_1 \xrightarrow{u_1}_{\mathcal{A}} f_2$ mit $q_1 \notin F$ und $f_1 \not\xrightarrow{u_2}_{\mathcal{A}} f_2$, wie in Beispiel 4.11.

Aus "i) gdw ii)" folgt offenbar "1. gdw 2." des Satzes. ∎

Korollar 4.14 Die Klasse der von deterministischen Büchi–Automaten akzeptierten Sprachen ist *nicht* unter Komplement abgeschlossen.

Beweis: In Beispiel 4.11 haben wir gesehen, daß $L = (a \cup b)^*b^\omega$ nicht von einem deterministischen Büchi–Automaten akzeptiert wird. Man sieht leicht, daß $\overline{L} = \{a, b\}^\omega \setminus L$ die ω–Sprache ist, welche alle unendlichen Wörter über $\{a, b\}$ enthält, die unendlich viele a enthalten. Daher ist $\overline{L} = \lim(b^*a)^*$, also von einem deterministischen Büchi–Automat erkennbar. ∎

4.2 Abschluß unter Komplement

Die von Büchi–Automaten erkennbaren Sprachen (d.h. die ω–regulären Sprachen) sind unter Komplement abgeschlossen. Der Beweis hierfür ist aber viel aufwendiger als im Fall der regulären Sprachen.

Hauptsatz 4.15 Ist $L \subseteq \Sigma^\omega$ ω–regulär, so auch $\Sigma^\omega \setminus L$.

Um dies zu beweisen, stellen wir L und $\Sigma^\omega \setminus L$ als endliche Vereinigung von ω–Sprachen $U \cdot V^\omega$ für reguläre Sprachen $U, V \subseteq \Sigma^*$ dar. Diese Sprachen werden die Klassen einer gewissen Kongruenzrelation $\sim_{\mathcal{A}}$ von endlichem Index sein.

Es sei dazu $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ein Büchi–Automat mit $L_\omega(\mathcal{A}) = L$. Wir schreiben $p \xrightarrow{w,F} q$ um auszudrücken, daß es in $\mathcal{A}$ einen Pfad von p nach q mit Beschriftung $w \in \Sigma^*$ gibt, der mindestens einen Zustand aus F enthält. Die Mengen

$$L^F_{p,q} := \{w \in \Sigma^* \mid p \xrightarrow{w,F} q\}$$

sind offenbar reguläre Sprachen:

$$L^F_{p,q} = \bigcup_{f \in F} L_{p,f} \cdot L_{f,q}.$$

Definition 4.16 Die Äquivalenzrelation $\sim_{\mathcal{A}}$ auf Σ^* ist definiert durch

$$\begin{array}{lll} u \sim_{\mathcal{A}} v & \text{gdw} & \text{für alle } p, q \in Q \text{ gilt} \\ & & 1.\ p \xrightarrow{u} q \quad \text{gdw} \quad p \xrightarrow{v} q \quad \text{und} \\ & & 2.\ p \xrightarrow{u,F} q \quad \text{gdw} \quad p \xrightarrow{v,F} q. \end{array}$$

Lemma 4.17 Die Relation $\sim_{\mathcal{A}}$ ist eine Kongruenzrelation, die endlichen Index hat (d.h. nur endlich viele Klassen).

Beweis:

1. Offenbar ist $\sim_{\mathcal{A}}$ eine Äquivalenzrelation (da "gdw" reflexiv, transitiv und symmetrisch ist). Damit $\sim_{\mathcal{A}}$ eine Kongruenzrelation ist, müssen wir zeigen: $u \sim_{\mathcal{A}} v \rightsquigarrow xuy \sim_{\mathcal{A}} xvy$.

   Es gelte also $u \sim_{\mathcal{A}} v$. Es gelte außerdem $p \xrightarrow{xuy,F} q$. Dann gibt es p', q' mit $p \xrightarrow{x} p' \xrightarrow{u} q' \xrightarrow{y} q$. Es folgt eine Fallunterscheidung danach, in welchem der drei Schritte Zustände aus F benutzt werden.

   1. Fall: $p \xrightarrow{x,F} p'$, d.h. $f \in F$ kommt im Pfad $p \xrightarrow{x} p'$ vor. Wegen $u \sim_{\mathcal{A}} v$ folgt aus $p' \xrightarrow{u} q'$ auch $p' \xrightarrow{v} q'$ und daher $p \xrightarrow{x,F} p' \xrightarrow{v} q' \xrightarrow{y} q$, also gilt auch $p \xrightarrow{xvy,F} q$.

   2. Fall: Der Beweis für $q' \xrightarrow{y,F} q$ erfolgt analog.

   3. Fall: $p' \xrightarrow{u,F} q'$. Wegen $u \sim_{\mathcal{A}} v$ folgt $p' \xrightarrow{v,F} q'$ und damit

   $$p \xrightarrow{x} p' \xrightarrow{v,F} q' \xrightarrow{y} q.$$

   Also folgt stets aus $p \xrightarrow{xuy,F} q$ auch $p \xrightarrow{xvy,F} q$. Die anderen Fälle kann man entsprechend behandeln: Die Rückrichtung des zweiten Teils der Definition ist symmetrisch zum bereits gezeigten, und der erste Teil ist diesem zweiten Teil ähnlich.

2. Die $\sim_{\mathcal{A}}$–Klasse eines Wortes w ist eindeutig bestimmt durch das Paar von Mengen $(\{(p,q) \mid p \xrightarrow{w} q\}, \{(p,q) \mid p \xrightarrow{w,F} q\})$. Es gibt daher maximal $2^{|Q\times Q|} \cdot 2^{|Q\times Q|}$ viele verschiedene $\sim_{\mathcal{A}}$–Klassen.

Wie sieht nun die $\sim_{\mathcal{A}}$–Klasse $[w]$ eines Wortes w aus?

Lemma 4.18 Für $w \in \Sigma^*$ gilt

1. $[w] = \bigcap_{\substack{p,q\in Q \\ w\in L_{p,q}}} L_{p,q} \cap \bigcap_{\substack{p,q\in Q \\ w\notin L_{p,q}}} \overline{L_{p,q}} \cap \bigcap_{\substack{p,q\in Q \\ w\in L^F_{p,q}}} L^F_{p,q} \cap \bigcap_{\substack{p,q\in Q \\ w\notin L^F_{p,q}}} \overline{L^F_{p,q}}$.

2. Insbesondere ist daher $[w] \subseteq \Sigma^*$ eine reguläre Sprache.

Beachte: Der Durchschnitt über eine leere Indexmenge wird hier als Σ^* angenommen, also

$$\bigcap_{\substack{p,q\in Q \\ w\in L^F_{p,q}}} L^F_{p,q} = \Sigma^*, \qquad \text{falls } w \notin L^F_{p,q} \text{ für alle } p,q \in Q.$$

Beweis: 2. folgt unmittelbar aus 1. Wir zeigen im folgenden also nur 1.

"$\subseteq$" Sei $u \in [w]$, d.h. $u \sim_{\mathcal{A}} w$. Ist $w \in L_{p,q}(\overline{L_{p,q}}, L^F_{p,q}, \overline{L^F_{p,q}})$ so folgt wegen $u \sim_{\mathcal{A}} w$ auch $u \in L_{p,q}(\overline{L_{p,q}}, L^F_{p,q}, \overline{L^F_{p,q}})$.

"$\supseteq$" Sei u in dem Durchschnitt auf der rechten Seite. Wir müssen $u \in [w]$ zeigen, d.h. $p \xrightarrow{u} q$ gdw $p \xrightarrow{w} q$ und $p \xrightarrow{u,F} q$ gdw $p \xrightarrow{w,F} q$.

- Aus $p \xrightarrow{w} q$ folgt $w \in L_{p,q}$ und damit $u \in L_{p,q}$, d.h. $p \xrightarrow{u} q$.
- Aus $p \not\xrightarrow{w} q$ folgt $w \in \overline{L_{p,q}}$ und damit $u \in \overline{L_{p,q}}$, d.h. $p \not\xrightarrow{u} q$.
- Für $\xrightarrow{w,F}$ erfolgen die Beweise analog. ∎

Hauptsatz 4.15 ergibt sich sehr leicht aus dem folgenden Satz.

Satz 4.19 Es sei $\mathcal{A}$ ein Büchi–Automat.

1. Für jedes Paar U, V von $\sim_{\mathcal{A}}$–Klassen gilt:
   (a) Ist $UV^\omega \cap L_\omega(\mathcal{A}) \neq \emptyset$, so ist $UV^\omega \subseteq L_\omega(\mathcal{A})$,
   (b) Ist $UV^\omega \cap \overline{L_\omega(\mathcal{A})} \neq \emptyset$, so ist $UV^\omega \subseteq \overline{L_\omega(\mathcal{A})}$.
2. Für jedes $\alpha \in \Sigma^\omega$ existieren $\sim_{\mathcal{A}}$–Klassen U, V mit $\alpha \in UV^\omega$.

Wir überlegen zunächst, warum hieraus folgt, daß $\overline{L_\omega(\mathcal{A})} = \Sigma^\omega \setminus L_\omega(\mathcal{A})$ ω–regulär ist.

Aus 1. und 2. des Satzes folgt, daß

$$L_\omega(\mathcal{A}) = \bigcup_{\substack{U,V \sim_{\mathcal{A}}\text{–Klassen} \\ UV^\omega \subseteq L_\omega(\mathcal{A})}} UV^\omega,$$

$$\overline{L_\omega(\mathcal{A})} = \bigcup_{\substack{U,V \sim_{\mathcal{A}}\text{–Klassen} \\ UV^\omega \subseteq \overline{L_\omega(\mathcal{A})}}} UV^\omega.$$

Da alle $\sim_{\mathcal{A}}$–Klassen U, V regulär sind und $\sim_{\mathcal{A}}$ endlichen Index hat, ist mit Satz 4.8 $\overline{L_\omega(\mathcal{A})}$ ω–regulär. Beachte: Der Punkt 2 dieses Satzes ist für "$\subseteq$" nötig!

Man kann den Satz 4.19 durch ad–hoc–Argumente beweisen. Eleganter ist aber die Verwendung eines kombinatorischen Resultats, das auch in anderen Bereichen sehr nützlich ist.

Definition 4.20 Für eine Menge M bezeichne $[M]^2$ die *Menge aller 2–elementigen Teilmengen* von M. Es sei nun

$$[M]^2 = A_1 \dot\cup A_2 \dot\cup \cdots \dot\cup A_n$$

eine Zerlegung von $[M]^2$ in endlich viele disjunkte Mengen. Eine Teilmenge X von M heißt *homogen für die Partition*, falls es ein i gibt mit

$$[X]^2 \subseteq A_i \qquad (1 \leq i \leq n).$$

Beispiel: Sei $[\mathbb{N}]^2 = A \dot\cup B$ mit

$$\begin{aligned} A &= \{\{i,j\} \mid i \neq j \text{ und } i \equiv j \bmod 2\} \quad \text{und} \\ B &= \{\{i,j\} \mid i \neq j \text{ und } i \not\equiv j \bmod 2\}. \end{aligned}$$

Dann sind

$$\begin{aligned} G &= \{i \in \mathbb{N} \mid i \text{ ist gerade}\} \quad \text{und} \\ U &= \{j \in \mathbb{N} \mid j \text{ ist ungerade}\} \end{aligned}$$

beide homogen für $A \dot\cup B$, da $[G]^2 \subseteq A$ und $[U]^2 \subseteq A$.

Satz 4.21 [Ramsey] Es sei $[\mathbb{N}]^2 = A_1 \dot\cup A_2 \dot\cup \cdots \dot\cup A_n$ eine disjunkte Zerlegung von $[\mathbb{N}]^2$. Dann gibt es eine unendliche Teilmenge $X \subseteq \mathbb{N}$, die homogen für diese Zerlegung ist.

Beweis: Siehe z.B. Seite 111 und 112 in [Ros82].

Kommen wir nun zu **Beweis von Satz 4.19**.

1. Es sei $\alpha \in UV^\omega \cap L_\omega(\mathcal{A})$. Das heißt

   (a) $\alpha = uv_1v_2v_3\cdots$ mit $u \in U$ und $v_i \in V \setminus \{\epsilon\}$.

(b) Es gibt einen erfolgreichen Pfad

$$q_0 \xrightarrow{u} q_1 \xrightarrow{v_1} q_2 \xrightarrow{v_2} q_3 \xrightarrow{v_3} \cdots$$

in $\mathcal{A}$ mit Beschriftung $\alpha = uv_1v_2v_3\cdots$ und mit $q_0 \in I$.

Dieser Pfad ist erfolgreich, daher werden unendlich oft Zustände aus F durchlaufen, d.h. es gibt unendlich viele $i \geq 1$, so daß

$$q_i \xrightarrow{v_i,F} q_{i+1}$$

gilt. Es sei nun $\beta \in UV^\omega$ beliebig. Wir müssen zeigen, daß dann auch $\beta \in L_\omega(\mathcal{A})$ gilt. Wir wissen, daß β sich schreiben läßt als

$$\beta = u'v_1'v_2'v_3'\cdots$$

mit $u' \in U$ und $v_i' \in V$. Da U und V $\sim_{\mathcal{A}}$–Klassen sind, folgt $u \sim_{\mathcal{A}} u'$ und $v_i \sim_{\mathcal{A}} v_i{}'$. Darum gibt es einen Pfad

$$q_0 \xrightarrow{u'} q_1 \xrightarrow{v_1'} q_2 \xrightarrow{v_2'} q_3 \xrightarrow{v_3'} \cdots,$$

auf dem dann natürlich auch für unendlich viele $i \geq 1$ gilt: $q_i \xrightarrow{v_i',F} q_{i+1}$. Daher gilt auch $\beta \in L_\omega(\mathcal{A})$. Damit ist 1.(a) von Satz 4.19 gezeigt. Teil 1.(b) folgt dann unmittelbar: Ist $\alpha \in UV^\omega \cap \overline{L_\omega(\mathcal{A})}$, so kann es kein β mit $\beta \in UV^\omega \cap L_\omega(\mathcal{A})$ geben, da sonst mit Teil 1.(a) unmittelbar $\alpha \in L_\omega(\mathcal{A})$ folgen würde.

2. Hier werden wir nun den Satz von Ramsey verwenden. Es sei $\alpha \in \Sigma^\omega$. Zusammen mit der Relation $\sim_{\mathcal{A}}$ definiert α die folgende Zerlegung auf $[\mathbb{N}]^2$.

   Es seien $U_1, U_2, \ldots, U_n$ die (endlich vielen) $\sim_{\mathcal{A}}$–Klassen. Wir definieren

   $$A_\nu = \{\{i,j\} \mid i < j \text{ und } \alpha(i+1,j) \in U_\nu\}.$$

   Da $\sim_{\mathcal{A}}$ eine Äquivalenzrelation auf Σ^* ist, ist jedes endliche Wort in genau einer ihrer Äquivalenzklassen, daher gilt

   $$[\mathbb{N}]^2 = A_1 \dot\cup A_2 \dot\cup \cdots \dot\cup A_n.$$

Nach Ramsey (Satz 4.21) gibt es eine unendliche Teilmenge X von $\mathbb{N}$, die homogen für diese Zerlegung ist. Es gibt also ein $k, 1 \leq k \leq n$, so daß $[X]^2 \subseteq A_k$ gilt, d.h. für alle $i, j \in X$ mit $i < j$ gilt: $\alpha(i+1, j) \in U_k$.

Da X unendlich ist, gibt es eine unendliche Folge $i_1, i_2, i_3, \ldots$ in X mit $i_j + 1 < i_{j+1}$. Damit wissen wir: $\alpha(i_j + 1, i_{j+1}) \in U_k \setminus \{\epsilon\}$. Es sei nun U die $\sim_{\mathcal{A}}$–Klasse von $\alpha(0, i_1)$. Dann läßt sich α schreiben als

$$\alpha = \alpha(0, i_1)\alpha(i_1 + 1, i_2)\alpha(i_2 + 1, i_3) \ldots \in UU_k^{\omega}.$$

■

Korollar 4.22 Zu einem gegebenen Büchi–Automaten $\mathcal{A}$ kann man effektiv einen Büchi–Automaten $\mathcal{B}$ konstruieren mit

$$L_\omega(\mathcal{B}) = \overline{L_\omega(\mathcal{A})}.$$

Beweis:

1. Die $\sim_{\mathcal{A}}$–Klassen können effektiv bestimmt werden: Lemma 4.18 zeigt, wie man sie aus den Sprachen $L_{p,q}$ und $L^F_{p,q}$ erhält.

2. Für ein gegebenes Paar U, V von $\sim_{\mathcal{A}}$–Klassen ist entscheidbar, ob

   $$UV^\omega \cap L_\omega(\mathcal{A}) = \emptyset$$

   gilt (siehe Satz 4.10).

3. Für UV^ω kann man effektiv Büchi–Automaten konstruieren, also auch für endliche Vereinigungen davon. ■

Korollar 4.23 Das Äquivalenzproblem für Büchi–Automaten ist entscheidbar.

Beweis:

$$L_\omega(\mathcal{A}_1) = L_\omega(\mathcal{A}_2) \text{ gdw}$$

$$(L_\omega(\mathcal{A}_1) \setminus L_\omega(\mathcal{A}_2)) \cup (L_\omega(\mathcal{A}_2) \setminus L_\omega(\mathcal{A}_1)) = \emptyset. \tag{4.1}$$

Aus Lemma 4.7 wissen wir, daß die Klasse der Büchi–erkennbaren Sprachen effektiv unter Vereinigung abgeschlossen ist, und Korollar 4.22 liefert die effektive Abgeschlossenheit unter Komplementbildung. Daher kann man auch einen Büchi–Automaten für die in der linken Seite von Gleichung 4.1 genannte Sprache konstruieren. Dieser akzeptiert die leere Sprache, falls kein Endzustand auf einem erreichbaren Zyklus liegt (d.h. falls für alle $i \in I, f \in F$ gilt $L_{i,f} = \emptyset$ oder $L_{f,f} \setminus \{\epsilon\} = \emptyset$). ∎

4.3 Muller–Automaten

Wir haben gesehen, daß deterministische Büchi–Automaten nicht alle ω–regulären Sprachen erkennen können. Wir betrachten nun ein modifiziertes Automatenmodell, bei dem bereits die deterministischen Automaten alle ω–regulären Sprachen erkennen.

Definition 4.24 [Muller–Automat] Ein *Muller–Automat* ist ein *deterministischer* endlicher Automat $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$, wobei

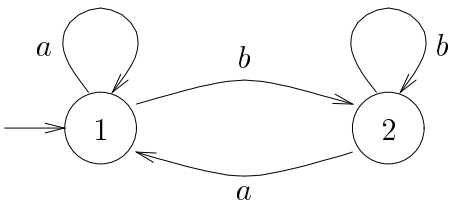
- Q, Σ, q_0, δ wie bei deterministischen Büchi–Automaten definiert sind und
- $\mathcal{F} \subseteq 2^Q$ eine Menge von Mengen von Endzuständen ist.

Ein unendlicher Pfad $p_0 \xrightarrow{a_0} p_1 \xrightarrow{a_0} p_2 \xrightarrow{a_2} \cdots$ in einem Muller–Automaten heißt erfolgreich, falls gilt:

- er beginnt mit dem Anfangszustand, d.h. $p_0 = q_0$ und
- $\{p \in Q \mid \text{Es gibt unendlich viele } i \text{ mit } p = p_i\} \in \mathcal{F}$.

Für einen Muller–Automaten $\mathcal{A}$ ist $L_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \alpha \text{ ist die Beschriftung eines erfolgreichen Pfades}\}$ die von ihm akzeptierte Sprache.

Beispiel 4.25 Betrachten wir wieder $L = (a \cup b)^* b^\omega$. Wie bereits gesehen, wird L nicht von einem deterministischen Büchi–Automaten akzeptiert (Beispiel 4.11). Der folgende (deterministische) Muller–Automat akzeptiert L:



wobei $\mathcal{F} = \{\{2\}\}$ ist. Denn: Ist die Menge aller unendlich oft durchlaufener Zustände eines Pfades $\{2\}$ (d.h. Zustand 1 wird nur endlich oft durchlaufen), so muß es offenbar eine Stelle in der Beschriftung dieses Pfades geben, ab der nur noch b vorkommt.

Beachte: Ein Büchi–Automat mit $F = \{2\}$ akzeptiert z.B. auch $(a \cup b)^\omega$.

Satz 4.26 Für eine ω–Sprache L sind äquivalent:

1. L ist ω–regulär.
2. L wird von einem Muller–Automaten akzeptiert.

Beweis:

"$2 \rightsquigarrow 1$" (relativ einfach) Es sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ ein (deterministischer) Muller–Automat. Dann gilt:

$$\begin{aligned} L &= L_\omega(\mathcal{A}) \\ &= \bigcup_{F \in \mathcal{F}} \left(\bigcap_{q \in F} \lim L_{q_0,q} \setminus \bigcup_{q \in Q \setminus F} \lim L_{q_0,q} \right) \\ &= \bigcup_{F \in \mathcal{F}} \left(\bigcap_{q \in F} \lim L_{q_0,q} \cap \bigcap_{q \in Q \setminus F} \overline{\lim L_{q_0,q}} \right). \end{aligned} \tag{4.2}$$

Begründung: Muller–Automaten akzeptieren ein ω–Wort, falls es einen Pfad beschriftet, der genau die Endzustände *einer* Endzustandsmenge

$F \in \mathcal{F}$ unendlich oft durchläuft. Daher die große Vereinigung über alle diese F. Da $\mathcal{A}$ deterministisch ist, enthält $\lim L_{q_0,q}$ genau die Wörter, die einen unendlichen Pfad beschriften, der q unendlich oft durchläuft. Da $\mathcal{A}$ deterministisch ist, erhält man mit dem Durchschnitt über alle Zustände in F genau die Wörter, deren Pfade jeden dieser Zustände unendlich oft durchlaufen (es kann also nicht vorkommen, daß ein Wort zwei verschiedene Pfade beschriftet, die unterschiedliche Endzustände unendlich oft durchlaufen). Pfade, die mit von $\mathcal{A}$ akzeptierten Wörtern beschriftet sind, durchlaufen Nicht–Endzustände (d.h. Zustände in $Q \setminus F$) jedoch nur endlich oft, daher müssen wir diese Wörter abziehen. Also gilt Gleichung 4.2. Wir wissen, daß $\lim L_{q_0,q}$ ω–regulär ist (Satz 4.13). Außerdem sind die ω–regulären Sprachen nach Lemma 4.7 unter Vereinigung und Durchschnitt und nach Satz 4.15 auch unter Komplement abgeschlossen. Also ist die Sprache in Gleichung 4.2 auch ω–regulär.

"$1 \rightsquigarrow 2$" Dieser Teil des Beweises ist mindestens so aufwendig wie der Beweis von Satz 4.15, da man leicht zeigen kann (siehe Satz 4.27), daß (deterministische) Muller–Automaten unter Komplement abgeschlossen sind. Wir lassen den Beweis daher weg.

Satz 4.27 Wird $L \subseteq \Sigma^\omega$ von einem Muller–Automaten akzeptiert, so auch $\overline{L} = \Sigma^\omega \setminus L$.

Beweis: Sei $L = L_\omega(\mathcal{A})$ für einen (deterministischen) Muller–Automaten $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$. Man sieht leicht, daß $\mathcal{B} = (Q, \Sigma, q_0, \delta, 2^Q \setminus \mathcal{F})$ die ω–Sprache $\overline{L}$ akzeptiert (Ist die Menge aller unendlich oft durchlaufener Zustände eines mit α beschrifteten Pfades nicht in $\mathcal{F}$, d.h. $\alpha \in \overline{L}$, so ist sie in $2^Q \setminus \mathcal{F}$ und $\alpha \in L_\omega(\mathcal{B})$). ∎

Kapitel 5

Unendliche Wörter und logische Formeln

5.1 Die Logik S1S und ω–reguläre Sprachen

Ziel dieses Kapitels ist es, eine Formelklasse zu beschreiben, welche genau die ω–regulären Sprachen akzeptiert. Wie im Fall endlicher Wörter (regulärer Sprachen) werden wir sehen, daß Prädikatenlogik erster Stufe nicht ausreicht. Man muß zusätzlich Quantifizierung über einstellige Prädikate zulassen.

Definition 5.1 (S1S) Formeln der *Monadischen Logik zweiter Stufe eines Nachfolgers* (S1S steht für **S**econd order logic of **1 S**uccessor) sind aufgebaut mit Hilfe von:

- n einstelligen Prädikatssymbolen $P_1, P_2, \ldots, P_n$,
- einem einstelligen Funktionssymbol $\boldsymbol{s}$,
- einem Konstantensymbol $\mathbf{0}$,
- einem 2–stelligen Prädikatssymbol $\dot{<}$,
- den Booleschen Operatoren $\wedge, \vee, \neg$,

- den Quantoren erster Stufe $\exists x_\bullet, \forall x_\bullet$, wobei x für ein Element des Interpretationsbereichs steht,
- den Quantoren zweiter Stufe $\exists X_\bullet, \forall X_\bullet$, wobei X für eine Teilmenge des Interpretationsbereichs steht.

Als Interpretationsbereich betrachten wir die natürlichen Zahlen ω, und wir interpretieren

- $\mathbf{0}$ als 0,
- $\boldsymbol{s}$ als natürliche Nachfolgerfunktion $m \mapsto m+1$,
- $\dot{<}$ als die übliche Ordnung $<$ auf ω.[1]

Wie im endlichen Fall sei $\Sigma = \{0,1\}^n$. Eine gegebene Interpretation $P_1^I, P_2^I, \ldots, P_n^I$ der Symbole $P_1, P_2, \ldots, P_n$ entspricht dann einem unendlichen Wort

$$\alpha = \alpha(0)\alpha(1)\alpha(2)\cdots \in \Sigma^\omega, \text{ wobei}$$

$$\alpha(m) = (b_{m_1}, \ldots, b_{m_n}) \text{ mit } b_{m_i} = \begin{cases} 1 & \text{falls } m \in P_i^I \\ 0 & \text{falls } m \notin P_i^I \end{cases} .$$

Für eine geschlossene S1S–Formel ϕ schreiben wir wieder $\alpha \models \phi$, wenn die Interpretation, die dem ω–Wort α entspricht, ϕ wahr macht. Die von ϕ *akzeptierte ω–Sprache* ist definiert durch:

$$L_\omega(\phi) = \{\alpha \in \Sigma^\omega \mid \alpha \models \phi\}.$$

Beispiel 5.2 Es sei $n = 1$, also ist $\Sigma = \{0,1\}$.

[1] Gleichheit kann daher durch $x \doteq y \Leftrightarrow \neg(x \dot{<} y \vee y \dot{<} x)$ ausgedrückt werden, d.h. S1S mit Gleichheit ist genauso ausdrucksstark ist wie S1S ohne Gleichheit.

1. Betrachten wir zunächst eine Formel ϕ und die von ihr akzeptierte ω–Sprache $L_\omega(\phi)$.

$$\begin{aligned} \phi &= P_1(\mathbf{0}) \wedge \\ &\quad (\forall x. P_1(x) \Rightarrow \neg P_1(\boldsymbol{s}(x))) \wedge \\ &\quad (\forall x. \neg P_1(x) \Rightarrow P_1(\boldsymbol{s}(x))) \\ L_\omega(\phi) &= \{10101010\cdots\} = (10)^\omega \end{aligned}$$

2. Sei $L_1 = \{\alpha \in \Sigma^\omega \mid$ nach jeder 1 in α kommt noch eine $0\}$. Dann wird L_1 offensichtlich durch die Formel

$$\phi = \forall x. P_1(x) \Rightarrow \exists y. x \dot{<} y \wedge \neg P_1(y)$$

beschrieben, das heißt $L_\omega(\phi) = L_1$.

3. Betrachte $L_2 = \{\alpha \in \Sigma^\omega \mid$ zwischen zwei Einsen in α befindet sich eine gerade Anzahl von Nullen$\}$. Um L_2 durch eine Formel zu beschreiben, betrachten wir Positionen x, y im Wort, an denen eine 1 steht und zwischen denen nur Nullen vorkommen. Wir führen die Prädikate Y und X ein, die für "gerade" und "ungerade" stehen, und sorgen dafür, daß nur geradzahlig viele Nullen zwischen zwei Einsen stehen. Damit gilt dann $L_\omega(\phi) = L_2$ für:

$$\begin{aligned} \phi = \forall x. \forall y. (x \dot{<} y \wedge P_1(x) \wedge P_1(y) \wedge \forall z. x \dot{<} z \wedge z \dot{<} y \Rightarrow \neg P_1(z)) \\ \Rightarrow \exists X. \exists Y. \forall z. (x \dot{<} z \wedge z \dot{<} y) \Rightarrow \\ (\neg(X(z) \wedge Y(z)) \wedge \\ (X(z) \Rightarrow Y(\boldsymbol{s}(z))) \wedge (Y(z) \Rightarrow X(\boldsymbol{s}(z))) \wedge \\ X(\boldsymbol{s}(x)) \wedge X(y)). \end{aligned}$$

Wie im endlichen Fall verwenden wir im folgenden wieder $Q_a(x)$ als Abkürzung für die Formel, die ausdrückt, daß an der Position x das Symbol $a \in \Sigma$ steht.

Wir zeigen als nächstes, daß man auf die Symbole $\mathbf{0}$ und $\dot{<}$ verzichten kann, ohne an Ausdrucksstärke zu verlieren.

Lemma 5.3 Sowohl $\mathbf{0}$ als auch $\dot{<}$ kann in S1S mit Hilfe der übrigen Symbole definiert werden.

Beweis: $x \doteq \mathbf{0}$ ist offensichtlich äquivalent zu $\neg\exists y.(y \dot{<} x)$ und $x \dot{<} y$ ist äquivalent zu $\exists X.(\neg X(x) \wedge X(y) \wedge \forall z.(X(z) \Rightarrow X(\boldsymbol{s}(z))))$. ∎

Nun also der Zusammenhang zwischen S1S–Formeln und ω–regulären Sprachen.

Satz 5.4 Für eine ω–Sprache L sind äquivalent:

1. L ist ω–regulär.
2. $L = L_\omega(\phi)$ für eine geschlossene S1S–Formel ϕ.

Beweis:

"$1 \rightsquigarrow 2$" Es sei $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ein Büchi–Automat mit $L = L_\omega(\mathcal{A})$. Wir werden die Existenz eines erfolgreichen Pfades durch eine S1S–Formel beschreiben. Es sei

$$\begin{aligned} Q &= \{q_0, \ldots, q_m\} \text{ und ohne Einschränkung} \\ I &= \{q_0, \ldots, q_k\} \text{ für ein } k \leq m. \end{aligned}$$

Wir verwenden Variablen $Y_0, Y_1, \ldots, Y_m$, wobei $Y_i(x)$ als "der x–te Zustand im Pfad ist q_i" gelesen werden sollte.

$$\exists Y_0 \cdots \exists Y_m. \Big(\forall x. \bigwedge_{0 \le i < j \le m} \neg(Y_i(x) \wedge Y_j(x))\Big) \wedge$$ Die Mengen sind disjunkt,

$$\Big(Y_0(\mathbf{0}) \vee \cdots \vee Y_k(\mathbf{0})\Big) \wedge$$ man beginnt mit einem Anfangszust.,

$$\Big(\forall x. \bigvee_{(q_i,a,q_j)\in\Delta} Y_i(x) \wedge Q_a(x) \wedge Y_j(\boldsymbol{s}(x))\Big) \wedge$$ an Position $\boldsymbol{s}(x)$ steht ein bezügl. Δ erlaubter Folgezust.,

$$\Big(\bigvee_{q_i \in F} \forall x. \exists y. x \dot{<} y \wedge Y_i(y)\Big)$$ einer der Endzust. wird unendlich oft angenommen.

Nach Konstruktion erfüllt $\alpha \in \Sigma^\omega$ diese Formel genau dann, wenn α die Beschriftung eines erfolgreichen Pfades ist.

"$2 \rightsquigarrow 1$" Wir werden zunächst S1S–Formeln in eine passende Normalform bringen:

1. Es dürfen nur noch Variablen X_i zweiter Stufe auftreten (kein x erster Ordnung).
2. Atomare Formeln haben die Gestalt
   - $X_i \dot{\subseteq} X_j$ (als Abkürzung für $\forall x. X_i(x) \Rightarrow X_j(x)$),
   - $\mathrm{Succ}(X_i) \doteq X_j$ (mit Semantik: $X_j^I = \{n_j\}, X_i^I = \{n_i\}$ und $n_i + 1 = n_j$).

   Dabei sind X_i, X_j Variablen zweiter Ordnung oder Prädikatssymbole $P_1, \ldots, P_n$.

Formeln, die aus diesen atomaren Formeln mit Hilfe Boolescher Operatoren und Quantifizierung zweiter Stufe aufgebaut sind, heißen $\mathrm{S1S}_0$–Formeln.

Behauptung: Jede S1S–Formel kann in eine äquivalente $\mathrm{S1S}_0$–Formel überführt werden.

Denn:

1. Wir haben bereits gesehen, daß $\mathbf{0}$ und $\dot{<}$ eliminiert werden können.
2. Geschachtelte Anwendung der Nachfolgerfunktion kann wie folgt eliminiert werden:

   $$y \doteq \underbrace{\boldsymbol{s}(\boldsymbol{s}(\cdots \boldsymbol{s}(x)\cdots))}_{m-\text{mal}}$$

   ist äquivalent zu

   $$\exists y_1 \ldots \exists y_{m-1}.\, y_1 \doteq \boldsymbol{s}(x) \wedge y_2 \doteq \boldsymbol{s}(y_1) \wedge \ldots \wedge \boldsymbol{s}(y_{m-1}) \doteq y.$$

3. Wir haben damit ohne Einschränkung nur noch atomare Formeln der Gestalt

   $$x \doteq y,\ \boldsymbol{s}(x) \doteq y,\ P_i(x),\ X(x).$$

   Im abschließenden Schritt verwenden wir die folgenden Abkürzungen:

   - $X \mathrel{\dot{=}} Y$ für $X \mathrel{\dot{\subseteq}} Y \wedge Y \mathrel{\dot{\subseteq}} X$,
   - $X \not\doteq Y$ für $\neg(X \mathrel{\dot{=}} Y)$,
   - $\text{Einer}(X)$ für "X hat genau eine echte Teilmenge", d.h. X ist Einermenge:

     $$\exists Y.(Y \mathrel{\dot{\subseteq}} X \wedge Y \not\doteq X \wedge \forall Z.(Z \mathrel{\dot{\subseteq}} X \Rightarrow (Z \mathrel{\dot{=}} X \vee Z \mathrel{\dot{=}} Y))).$$

4. Variablen erster Stufe können nun wie im folgenden Beispiel eliminiert werden:

   $$\forall x.\exists y.\, \boldsymbol{s}(x) \doteq y \wedge Z(y)$$
   wird zu
   $$\forall X.\text{Einer}(X) \Rightarrow \exists Y.(\text{Einer}(Y) \wedge \text{Succ}(X) \mathrel{\dot{=}} Y \wedge Y \mathrel{\dot{\subseteq}} Z).$$

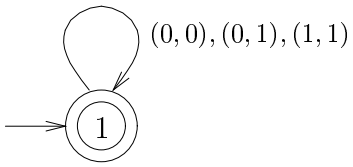
Dies schließt den Beweis der Behauptung ab.

Wir zeigen durch Induktion über den Aufbau von S1S_0–Formeln ϕ, daß $L_\omega(\phi)$ ω–regulär ist. Wir betrachten dabei auch Formeln mit freien Variablen (zweiter Stufe), wobei diese bei der Definition der akzeptierten Sprache wie einstellige Prädikatssymbole behandelt werden. So liefert z.B. $\exists Y.(X \mathrel{\dot{\subseteq}} Y \wedge P_1 \mathrel{\dot{\subseteq}} Y)$ eine ω–Sprache über $\Sigma = \{0,1\}^2$, da P_1 und X frei vorkommen.

Induktionsanfang: Atomare Formeln sind von der Gestalt $X \dot{\subseteq} Y$ oder $\mathrm{Succ}(X) \doteqdot Y$ und werden durch Wörter über $\Sigma = \{0,1\}^2$ interpretiert (Ohne Einschränkung steht dabei die erste Komponente für X und die zweite für Y).

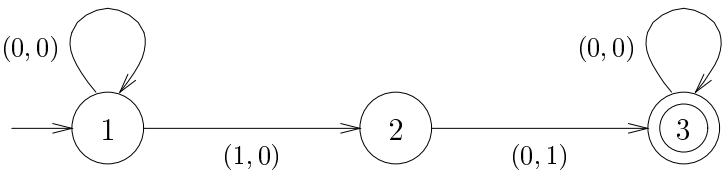
$$L_\omega(X \dot{\subseteq} Y) = \{\alpha = \alpha_0\alpha_1\alpha_2\cdots \mid \text{für } \alpha_i = (b_{i_1}, b_{i_2}) \text{ gilt: } b_{i_1} = 1 \rightsquigarrow b_{i_2} = 1\}$$

Daher gilt, daß $L_\omega(X \dot{\subseteq} Y)$ von folgendem Büchi–Automaten akzeptiert wird:



$$L_\omega(\mathrm{Succ}(X) \doteqdot Y) = \{\alpha = \alpha_0\alpha_1\alpha_2\cdots \mid \text{für } \alpha_i = (b_{i_1}, b_{i_2}) \text{ gilt: es gibt ein } k \text{ mit } b_{k_1} = 1 \text{ und } b_{(k+1)_2} = 1 \text{ und } b_{j_1} = 0 \text{ für } j \neq k \text{ } b_{j_2} = 0 \text{ für } j \neq k+1\}.$$

Daher wird $L_\omega(\mathrm{Succ}(X) \doteqdot Y)$ von folgendem Büchi–Automaten akzeptiert:



Induktionsschritt: Es genügt, $\neg, \vee, \exists X.$ zu betrachten.

1. $L_\omega(\neg\phi) = \Sigma^\omega \setminus L_\omega(\phi)$. Nach Induktionsannahme ist $L_\omega(\phi)$ ω–regulär und damit mit Satz 4.15 auch $\overline{L_\omega(\phi)} = \Sigma^\omega \setminus L_\omega(\phi)$.

2. "$\vee$" entspricht im Prinzip der Vereinigung, allerdings können bei $\phi = \phi_1 \vee \phi_2$ die in ϕ_1 und ϕ_2 vorkommenden Prädikatssymbole und freien Variablen verschieden sein.

   Beispiel:

   $$\phi(\underbrace{X_1, X_2, X_3}_{\text{freie Var.}}) = \phi_1(X_1, X_2) \vee \phi_2(X_2, X_3)$$

   Man erweitert dann ϕ_1 und ϕ_2 um die fehlenden freien Variablen oder Prädikatssymbole, indem man einfach trivial wahre Formeln anfügt, die diese enthalten (z.B. $\hat{\phi}_1(X_1, X_2, X_3) = \phi_1(X_1, X_2) \wedge X_3 \doteq X_3$). Allgemein gilt für $i \in \{1, 2\}$: Ist $\mathcal{A}_i$ ein Büchi–Automat für ϕ_i, so erhält man den Automaten $\hat{\mathcal{A}}_i$ für $\hat{\phi}_i$ wie folgt: $\Sigma = \{0,1\}^{n_i}$ von $\mathcal{A}_i$ wird ersetzt durch $\Sigma = \{0,1\}^{n_i+\ell_i}$ für ℓ_i freie Variablensymbole von ϕ_{3-i}, die nicht in ϕ_i vorkommen. In $\hat{\mathcal{A}}_1$ gibt es den Übergang

   $$q \xrightarrow{(b_1,\ldots,b_{n_i+\ell_i})} q'$$

   genau dann, wenn es n_i Komponenten $(b_{j_1}, \ldots, b_{j_{n_i}})$ in $(b_1, \ldots, b_{n_i+\ell_i})$ gibt, so daß $X_{j_1}, \ldots, X_{j_{n_i}}$ die in ϕ_i frei vorkommenden Variablen sind und

   $$q \xrightarrow{(b_{j_1},\ldots,b_{j_{n_i}})} q'$$

   ein Übergang in $\mathcal{A}_i$ ist. Dann ist

   $$\begin{aligned} & L_\omega(\phi(X_1, \ldots, X_m)) \\ &\quad = L_\omega(\phi_1(X_{j_1}, \ldots, X_{j_{n_1}}) \vee \phi_2(X_{k_1}, \ldots, X_{k_{n_2}})) \\ &\quad = L_\omega(\hat{\phi}_1(X_1, \ldots, X_m) \vee \hat{\phi}_2(X_1, \ldots, X_m)) \\ &\quad = L_\omega(\hat{\phi}_1(X_1, \ldots, X_m)) \cup L_\omega(\hat{\phi}_2(X_1, \ldots, X_m)) \\ &\quad = L_\omega(\hat{\mathcal{A}}_1) \cup L_\omega(\hat{\mathcal{A}}_2) \end{aligned}$$

   nach Induktionsvoraussetzung ω–regulär.

3. Es sei

   $$\phi(X_1, \ldots, X_n) = \exists Y . \rho(Y, X_1, \ldots, X_n).$$

   Ist $\mathcal{A}$ ein Automat für $L_\omega(\rho(Y, X_1, \ldots, X_n))$, so erhält man einen Automaten für $L_\omega(\phi(X_1, \ldots, X_n))$ wie folgt: Ersetze jeden Übergang

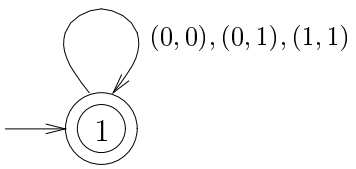
   $$q \xrightarrow{(b_0,b_1,\ldots,b_n)} q' \text{ in } \mathcal{A} \text{ durch } q \xrightarrow{(b_1,\ldots,b_n)} q'.$$ ∎

Das folgende Beispiel veranschaulicht, wie man mit der in diesem Beweis vorgestellten Konstruktion aus einer Formel einen Büchi–Automaten erhält.

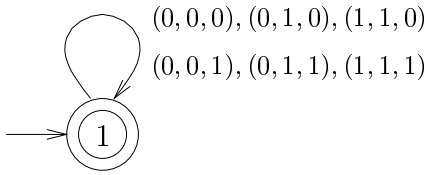
Beispiel 5.5 Es sei

$$\phi = \exists Y.(X \dot{\subseteq} Y \vee \mathrm{Succ}(Y) \doteq Z).$$

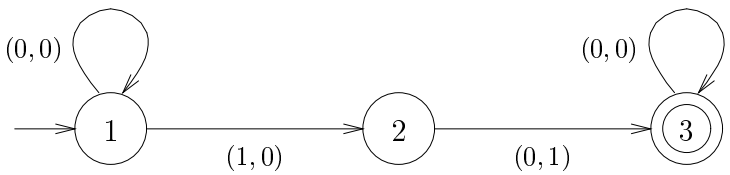
Der Automat $\mathcal{A}_1$ akzeptiert $L_\omega(X \dot{\subseteq} Y)$:



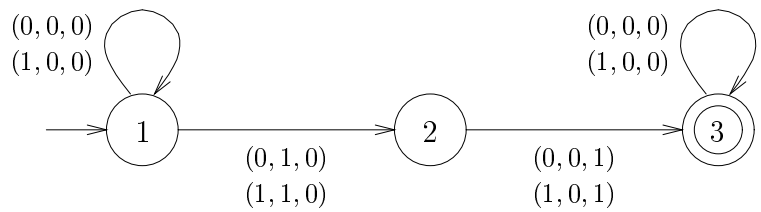
und $\hat{\mathcal{A}}_1$ ist der um Z erweiterte Automat:



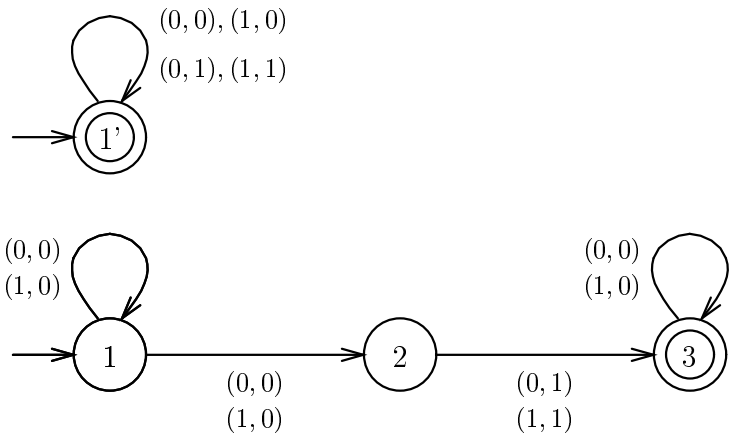
Der Automat $\mathcal{A}_2$ akzeptiert $L_\omega(\mathrm{Succ}(Y) \doteq Z)$:



und $\hat{\mathcal{A}}_2$ ist der um X erweiterte Automat:

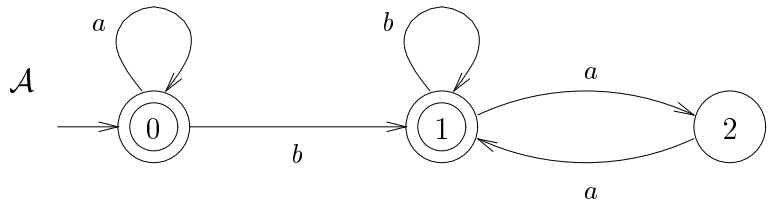


Die disjunkte Vereinigung dieser beiden Automaten, $\hat{\mathcal{A}}_1 \dot{\cup} \hat{\mathcal{A}}_2$, akzeptiert $L_\omega(X \dot{\subseteq} Y \vee \mathrm{Succ}(Y) \doteq Z)$. Der folgende Automat akzeptiert dann $L_\omega(\exists Y.(X \dot{\subseteq} Y \vee \mathrm{Succ}(Y) \doteq Z))$ und entsteht aus $\hat{\mathcal{A}}_1 \dot{\cup} \hat{\mathcal{A}}_2$ durch "Herausstreichen" der 2. Komponente (für Y):



Das nächste Beispiel zeigt, wie die S1S–Formel zu einem gegebenen Büchi–Automaten aussieht.

Beispiel 5.6 Sei $\Sigma = \{a, b\}$. Wir verwenden wie gewohnt die Abkürzungen $Q_a(x)$ für $P_1(x)$ und $Q_b(x)$ für $\neg P_1(x)$.



Es ist offensichtlich, daß $L_\omega(\mathcal{A}) = \{\alpha \in \{a, b\}^\omega \mid$ zwischen zwei b liegt eine gerade Anzahl von $a\}$ gilt. Die Konstruktion aus dem Beweis des Satzes 5.4 liefert folgende Formel, die $L_\omega(\mathcal{A})$ akzeptiert:

$$
\begin{aligned}
\exists Y_0.\exists Y_1.\exists Y_2.\Big(&\forall x.\neg(Y_0(x)\wedge Y_1(x))\wedge\\
&\quad\neg(Y_0(x)\wedge Y_2(x))\wedge\\
&\quad\neg(Y_1(x)\wedge Y_2(x))\Big)\wedge\\
&Y_0(\mathbf{0})\wedge\\
&\Big(\forall x.(Y_0(x)\wedge Q_a(x)\wedge Y_0(\boldsymbol{s}(x)))\vee\\
&\quad(Y_0(x)\wedge Q_b(x)\wedge Y_1(\boldsymbol{s}(x)))\vee\\
&\quad(Y_1(x)\wedge Q_b(x)\wedge Y_1(\boldsymbol{s}(x)))\vee\\
&\quad(Y_1(x)\wedge Q_a(x)\wedge Y_2(\boldsymbol{s}(x)))\vee\\
&\quad(Y_2(x)\wedge Q_a(x)\wedge Y_1(\boldsymbol{s}(x)))\Big)\wedge\\
&\Big((\forall x.\exists y.x\dot{<}y\wedge Y_0(y))\vee\\
&\ (\forall x.\exists y.x\dot{<}y\wedge Y_1(y))\Big)
\end{aligned}
$$

Der Beweis des Satzes 5.4 zeigt, daß man zu jeder S1S–Formel ϕ effektiv einen Büchi–Automaten $\mathcal{A}$ konstruieren kann mit $L_\omega(\mathcal{A}) = L_\omega(\phi)$. Daraus folgt, daß man von einer (geschlossenen) S1S–Formel ϕ entscheiden kann, ob sie in der kanonischen Interpretation $(\omega, <, +1, 0)$ *gültig* ist oder nicht. Offenbar ist eine Formel ϕ genau dann gültig (d.h. sie gilt in allen zulässigen Interpretationen), wenn es keine Interpretation gibt, die $\neg\phi$ wahr macht, also wenn $L_\omega(\neg\phi) = \emptyset$ gilt.

Korollar 5.7 Gültigkeit in S1S ist entscheidbar.

Beweis: Es sei ϕ eine (ohne Einschränkung geschlossene) S1S–Formel. Zu $\neg\phi$ kann man effektiv einen Büchi–Automaten $\mathcal{A}$ konstruieren mit $L_\omega(\mathcal{A}) = L_\omega(\phi)$. Das Leerheitsproblem für ω–reguläre Sprachen ist mit Satz 4.10 entscheidbar. ∎

Der Beweis von Satz 5.4 läßt sich leicht so abändern, daß er auch für reguläre Sprachen endlicher Wörter funktioniert.

Korollar 5.8 Für eine Sprache $L \subseteq \Sigma^*$ sind äquivalent:

1. L ist regulär.

2. $L \setminus \{\epsilon\} = L(\phi)$ für eine geschlossene S1S–Formel ϕ.

Wie im Kapitel 1.3 eingeführt, faßt man hier wieder endliche Wörter als endliche, total geordnete Interpretationen auf. Im **Beweis von "$1 \rightsquigarrow 2$"** muß im wesentlichen nur die Akzeptanzbedingung geändert werden:

$$\bigvee_{q_i \in F} \forall x. \exists y. x \dot{<} y \wedge Y_i(y)$$

wird ersetzt durch

$$\bigvee_{q_i \in F} \forall x. \max(x) \Rightarrow Y_i(x).$$

Im **Beweis von "$2 \rightsquigarrow 1$"** verwendet man die Abschlußeigenschaften regulärer Sprachen. Beim Induktionsanfang muß man die spezielle Semantik von $\boldsymbol{s}$ beachten (auf Maxima und Minima) und mit der Akzeptanzbedingung für endliche Wörter auskommen.

Korollar 5.9 Für eine S1S–Formel ist es entscheidbar, ob sie in allen endlichen S1S–Interpretationen gilt.

Man beachte: es gibt S1S–Formeln, die zwar in allen endlichen S1S–Interpretationen gelten, aber nicht in allen unendlichen.

Beispiel: Die Formel $\neg(\forall x. \exists y. (x \dot{<} y))$ gilt in allen endlichen S1S–Interpretationen, aber in keiner unendlichen.

5.2 Schwächere Logiken und Presburger Arithmetik

Büchi hat auch die schwache Logik WS1S (das W steht für *weak*) untersucht, bei der Quantoren $\exists X., \forall Y.$ zweiter Stufe nur über *endliche* Teilmengen von ω quantifizieren. Offenbar kann man WS1S in S1S ausdrücken:

Die Formel $\exists X.\phi(X)$ in WS1S wird in S1S zu

$$\exists X.(\exists y.\forall x.(X(x) \Rightarrow x \dot{<} y) \wedge \phi(X)).$$

Damit ist gezeigt, daß die Gültigkeit in WS1S ebenfalls entscheidbar ist. Umgekehrt kann man zeigen, daß WS1S genauso ausdrucksstark ist wie S1S, d.h. zu jeder ω–Sprache L gibt es eine WS1S Formel ϕ mit $L_\omega(\phi) = L$. Wir verwenden die Entscheidbarkeit von WS1S nun, um die Entscheidbarkeit der Presburger–Arithmetik zu zeigen.

Definition 5.10 Presburger Arithmetik Man betrachtet hier Formeln der Logik erster Stufe (mit Gleichheit), die als nicht–logische Symbole eine Konstante $\underline{0}$ und ein zweistelliges Funktionssymbol + enthalten dürfen. Als Interpretation verwendet man die natürlichen Zahlen, wobei $\underline{0}$ als 0 und + als Addition interpretiert werden.

Satz 5.11 Die Presburger Arithmetik ist entscheidbar, d.h. für eine Formel der Logik erster Stufe, die als nicht–logische Symbole nur $\underline{0}$ und + enthält, ist es entscheidbar, ob sie in den natürlichen Zahlen gilt.

Beweis: Um in WS1S "rechnen" zu können, werden wir die natürlichen Zahlen in endliche Teilmengen natürlicher Zahlen codieren. Dazu stellt man die natürlichen Zahlen in Binärcodierung dar und interpretiert die resultierende `0-1`-Folge als endliche Teilmenge von ω: Ist $n = 2^{i_1} + 2^{i_2} + \cdots + 2^{i_k}$ mit $i_1 < i_2 < \cdots < i_k$, so wird n durch die Menge $\{i_1, i_2, \ldots, i_k\}$ codiert.

Beispiel:
$13 =$ `1101`, also wird 13 durch $\{0, 2, 3\}$ codiert.

$20 =$ `10100`, also wird 20 durch $\{2, 4\}$ codiert.

Die $\underline{0}$ der Presburger Arithmetik (die als 0 zu interpretieren ist), wird folglich durch $\{\}$ codiert und kann in WS1S durch ein einstelliges Prädikat Z dargestellt werden, das durch $\forall x.\neg Z(x)$ definiert wird. Beachte: Die **0** in S1S und WS1S bezeichnet die nullte Stelle eines Wortes, also den Summanden 2^0 in der oben beschriebenen Codierung. Diese codierten natürlichen Zahlen entsprechen den einstelligen Prädikaten in WS1S[2], mit denen wir auch

[2] Z.B. entspricht die 13 einem Prädikat X mit $\forall x.X(x) \Leftrightarrow x \doteq \mathbf{0} \vee x \doteq s(s(\mathbf{0})) \vee x \doteq s(s(s(\mathbf{0})))$.

"rechnen" können. Wir zeigen daher nun, daß man in WS1S die Addition natürlicher Zahlen ausdrücken kann:

Es gibt eine WS1S–Formel $\phi_+(X,Y,Z)$ mit den folgenden Eigenschaften: Ist K Codierung von k, N Codierung von n und M Codierung von m, so gilt:

$$\phi_+(K,N,M) \text{ gdw } k+n=m.$$

Beispiel: $20 + 13 = 33$

$$\begin{array}{rcll} \texttt{10100} & \widehat{=} & \{2,4\} & X \\ +\ \texttt{01101} & \widehat{=} & \{0,2,3\} & Y \\ \text{Übertrag: } \overline{\texttt{111000}} & \widehat{=} & \{3,4,5\} & R \\ \texttt{100001} & \widehat{=} & \{0,5\} & Z \end{array}$$

Wir benötigen also eine Hilfsvariable R zweiter Stufe, die den Übertrag beschreibt. Es muß nun an jeder Position x gelten:

- x *gehört zu* Z (d.h. an Position x in der Binärdarstellung der Summe Z steht eine `1`) genau dann, wenn eine oder alle drei Mengen X, Y, R das x enthalten.
- $x+1$ *gehört zu* R (d.h. `1` an Position $x+1$ im Übertrag) genau dann, wenn x zu mindestens zwei der Mengen X, Y, R gehört.

Dies wird durch die folgende Formel beschrieben.

$$\begin{array}{lll} \phi_+(X,Y,Z) = \exists R.(& \neg R(\mathbf{0}) \wedge & \text{"An Position 0 kein Übertrag"} \\ & \forall x.(R(s(x)) \Leftrightarrow & (X(x) \wedge Y(x)) \vee \\ & & (X(x) \wedge R(x)) \vee \\ & & (Y(x) \wedge R(x))) \wedge \\ & \forall x.(Z(x) \Leftrightarrow & (X(x) \wedge Y(x) \wedge R(x)) \vee \\ & & (X(x) \wedge \neg Y(x) \wedge \neg R(x)) \vee \\ & & (\neg X(x) \wedge Y(x) \wedge \neg R(x)) \vee \\ & & (\neg X(x) \wedge \neg Y(x) \wedge R(x)))) \end{array}$$

Man kann damit beliebige Presburgerformeln als WS1S–Formel darstellen; z.B. $\forall x.(x + \underline{0} = x)$ durch

$$\exists Z.(\underbrace{\forall x.(\neg Z(x))}_{Z=\{\}\widehat{=}0} \wedge \forall X.\phi_+(X,Z,X)).$$

Dies schließt den Beweis von Satz 5.11 ab. ∎

Man muß im Beweis dieses Satzes WS1S an Stelle von S1S verwenden, da ja die Codierung natürlicher Zahlen *endliche* Mengen sind, also Quantoren $\forall x$ in Presburger–Formeln den Quantoren $\forall X$ über *endliche* Teilmengen von ω entsprechen.

5.3 Sternfreie Sprachen und unendliche Wörter

Bei endlichen Wörtern konnte die Klasse von Sprachen, die von Formeln der Logik erster Stufe akzeptiert werden, sehr schön charakterisiert werden (Satz 3.9).

Für $L \subseteq \Sigma^*$ sind äquivalent:

1. L ist sternfrei.
2. $L \setminus \{\epsilon\} = L(\phi)$ für eine geschlossene Formel ϕ der Logik erster Stufe über den nicht–logischen Symbolen $\dot{<}$, $Q_a(a \in \Sigma)$.

Bei ω–Sprachen gibt es einen entsprechenden Zusammenhang.

Definition 5.12 Eine ω–Sprache $L \subseteq \Sigma^\omega$ heißt *sternfrei* genau dann, wenn $L = \bigcup_{i=1}^n U_i V_i^\omega$ für sternfreie Sprachen $U_i, V_i \subseteq \Sigma^*$.

Satz 5.13 Für eine ω–Sprache $L \subseteq \Sigma^\omega$ sind äquivalent:

1. L ist sternfrei.
2. $L = L_\omega(\phi)$ für eine geschlossene S1S–Formel ϕ, welche keine Quantifizierung über einstellige Prädikate enthält, d.h. ϕ ist eine Formel der Logik erster Stufe.

Wir erwähnen dieses und die folgenden Resultate ohne Beweis (Literaturhinweise finden sich bei [Tho90b]).

Man kann sternfreie ω–Sprachen auch wieder mit Hilfe endlicher Monoide charakterisieren.

Definition 5.14 Es sei $L \subseteq \Sigma^\omega$ eine ω–Sprache. Die Kongruenzrelation $\approx_L$ auf Σ^* ist definiert durch

$$\begin{aligned} u \approx_L v \quad \text{gdw} \quad & \text{Für alle } x, y, z \in \Sigma^* \text{ gilt} \\ & (xuyz^\omega \in L \quad \text{gdw} \quad xvyz^\omega \in L) \text{ und} \\ & (x(yuz)^\omega \in L \quad \text{gdw} \quad x(yvz)^\omega \in L). \end{aligned}$$

Es ist leicht zu überprüfen, daß $\approx_L$ tatsächlich eine Kongruenzrelation ist. Für ω–reguläre Sprachen L hat $\approx_L$ stets endlichen Index. Im Gegensatz zum Fall endlicher Wörter gilt aber die Umkehrung nicht, d.h. es gibt nicht–reguläre ω–Sprachen, für die $\approx_L$ auch endlichen Index hat. Für eine Charakterisierung ω–regulärer Sprachen benötigt man noch eine zusätzliche Bedingung.

Definition 5.15 Es sei $L \subseteq \Sigma^\omega$ und $\sim$ eine Kongruenzrelation auf Σ^*. Die Relation $\sim$ *saturiert*[3] L genau dann, wenn es endlich viele $\sim$–Klassen $U_1, \ldots, U_n, V_1, \ldots, V_n$ gibt mit $L = \bigcup_{i=1}^n U_i V_i^\omega$.

Satz 5.16 Für eine ω–Sprache $L \subseteq \Sigma^\omega$ sind äquivalent:

1. L ist ω–regulär.
2. $\approx_L$ hat endlichen Index und saturiert L.

[3] Deutsch: sättigt

Die sternfreien ω–Sprachen entsprechen nun wieder den aperiodischen Monoiden.

Satz 5.17 Für eine ω–reguläre Sprache L sind äquivalent:

1. L ist sternfrei.
2. $\Sigma^*/_{\approx_L}$ ist aperiodisch.

Satz 5.13 und Satz 5.17 ermöglichen es nun wieder, für eine gegebene ω–reguläre Sprache zu überprüfen, ob sie von einer Formel der Logik erster Stufe akzeptiert wird.

Beispiel 5.18 (Vergleiche Beispiel 4.5) Sei $\Sigma = \{a, b, c\}$ und $L = \{\alpha \in \Sigma^\omega \mid$ zwischen zwei a befindet sich eine gerade Anzahl von b und $c\}$.

Die M–Varietät aller aperiodischen Monoide ist schließlich definiert durch $(x^{n+1} = x^n)_{n\geq 1}$. Wir zeigen, daß $x := [b]_{\approx_L} \in \Sigma^*/_{\approx_L}$ die Identität $x^{n+1} = x^n$ für kein n erfüllt. Ohne Einschränkung sei dazu n gerade und $n+1$ ungerade. Aus $x^{n+1} = x^n$ würde folgen, daß $b^n \approx_L b^{n+1}$ gilt. Es ist aber $ab^naa^\omega \in L$ (n gerade) und $ab^{n+1}aa^\omega \notin L$ ($n+1$ ungerade), also $b^n \not\approx_L b^{n+1}$.

Damit ist $\Sigma^*/_{\approx_L}$ nicht aperiodisch, also kann L nicht durch eine Formel der Logik erster Stufe definiert werden.

5.4 Dynamische Logik

Zum Abschluß dieses Kapitels betrachten wir eine weitere Logik, deren Entscheidbarkeit man mit Hilfe von Büchi–Automaten zeigen kann. Diese Logik soll das *Eingabe/Ausgabeverhalten eines deterministischen Programms* p beschreiben. Dieses Programm überführt (deterministisch) eine gegebene Konfiguration k in höchstens eine Folgekonfiguration k'. Gibt es ausgehend von k keine Folgekonfiguration, so terminiert p nicht. Das Programm kann mehrmals hintereinander ausgeführt werden und beliebig oft iteriert werden.

Wir gehen davon aus, daß endlich viele (atomare) Eigenschaften $A_1, \ldots, A_n$ von Konfigurationen gegeben sind. Man sucht nun nach einer Logik, mit der man Aussagen wie die folgenden ausdrücken kann:

a1: "Gilt A_1 in k, so gilt A_2 in der Folgekonfiguration."

a2: "Gilt A_1 in k, so existiert keine Folgekonfiguration (d.h. p terminiert nicht)."

a3: "Beginnend mit der Konfiguration k, in der A_1 gilt, kann man p endlich oft hintereinander ausführen und erreicht dadurch eine Konfiguration, in der A_2 gilt."

Definition 5.19 Formeln der Logik 1DPDL (**p**ropositionale **d**ynamische **L**ogik eines **d**eterministischen Programms) sind wie folgt aufgebaut:

Programmformeln

- p und ϵ sind Programmformeln.
- Sind π_1 und π_2 Programmformeln, so auch $\pi_1; \pi_2$ (Hintereinanderausführung), $\pi_1 \cup \pi_2$ (Alternative), π_1^* (Iteration).

Konfigurationsformeln

- $A_1, \ldots, A_n$ sind Konfigurationsformeln.
- Sind ϕ_1, ϕ_2 Konfigurationsformeln und ist π eine Programmformel, so sind auch $\phi_1 \vee \phi_2$, $\phi_1 \wedge \phi_2$, $\neg\phi_1$, $\langle\pi\rangle\phi_1$, $[\pi]\phi_1$ Konfigurationsformeln.

Programmformeln beschreiben Programme, die man aus dem atomaren Programm p aufbauen kann. (In 1DPDL gibt es genau ein atomares Programm p.) Zum Beispiel ist $p \cup (p; p)$ ein (nicht–deterministisches) Programm, bei dem entweder p einmal oder p zweimal ausgeführt wird.

$\langle\pi\rangle\phi_1$ heißt: es gibt eine π–Folgekonfiguration, in der ϕ_1 gilt, d.h. durch Ausführen von π kann eine Konfiguration erreicht werden, in der ϕ_1 gilt.

$[\pi]\phi_1$ heißt: in allen π–Folgekonfigurationen gilt ϕ_1.

Beachte: Da ein zusammengesetztes Programm nicht–deterministisch sein kann (im Gegensatz zu dem atomaren Programm p), kann es mehrere π-Folgekonfigurationen geben.

Beispiel 5.20 Die Aussagen a1, a2, a3 von oben können nun wie folgt beschrieben werden ($\phi \Rightarrow \psi$ sei wie üblich eine Abkürzung für $\neg\phi \vee \psi$):

a1: $A_1 \Rightarrow \langle p \rangle A_2$

a2: $A_1 \Rightarrow [p](A_1 \wedge \neg A_1)$ (keine Konfiguration kann $A_1 \wedge \neg A_1$ erfüllen)

a3: $A_1 \Rightarrow \langle p^* \rangle A_2$

Formal wird die Semantik von 1DPDL wie folgt definiert.

Definition 5.21 Eine *Interpretation* I besteht aus einer nicht–leeren Menge Δ^I von Konfigurationen und einer Interpretationsfunktion $\cdot^I$, die

1. dem atomaren Programm p eine funktionale Relation $p^I \subseteq \Delta^I \times \Delta^I$ zuordnet. Funktional heißt dabei, daß für alle k, k_1, k_2 gilt: Aus $(k, k_1) \in p^I$ und $(k, k_2) \in p^I$ folgt $k_1 = k_2$.
2. jeder atomaren Eigenschaft A_i eine Menge ${A_i}^I \subseteq \Delta^I$ zuordnet (die Konfigurationen, in denen A_i gilt).

Die Interpretationsfunktion kann auf Programmformeln und Konfigurationsformeln erweitert werden. Dies geschieht induktiv über den Aufbau der Formeln. Die Interpretation der atomaren Formeln $p, A_1, \ldots, A_n$ ist bereits definiert.

Jeder **Programmformel** π wird eine Relation $\pi^I \subseteq \Delta^I \times \Delta^I$ zugeordnet:

$$
\begin{aligned}
\epsilon^I &:= \{(k,k) \mid k \in \Delta^I\} \\
(\pi_1;\pi_2)^I &:= {\pi_1}^I \circ {\pi_2}^I = \\
&\quad \{(k_1,k_2) \mid \text{Es gibt } k \text{ mit } (k_1,k) \in {\pi_1}^I \text{ und } (k,k_2) \in {\pi_2}^I\} \\
(\pi_1 \cup \pi_2)^I &:= {\pi_1}^I \cup {\pi_2}^I \\
({\pi_1}^*)^I &:= \textstyle\bigcup_{i\geq 0} ({\pi_1}^I)^i, \qquad \text{wobei } ({\pi_1}^I)^i = \underbrace{{\pi_1}^I \circ \ldots \circ {\pi_1}^I}_{i\text{-mal}} \text{ ist.}
\end{aligned}
$$

Jeder **Konfigurationsformel** ϕ wird eine Menge $\phi^I \subseteq \Delta^I$ zugeordnet:

$$\begin{aligned}
(\phi_1 \wedge \phi_2)^I &:= {\phi_1}^I \cap {\phi_2}^I \\
(\phi_1 \vee \phi_2)^I &:= {\phi_1}^I \cup {\phi_2}^I \\
(\neg\phi_1)^I &:= \Delta^I \setminus {\phi_1}^I \\
(\langle\pi\rangle\phi_1)^I &:= \{k \in \Delta^I \mid \text{Es gibt } k' \text{ mit } (k,k') \in \pi^I \text{ und } k' \in {\phi_1}^I\} \\
([\pi]\phi_1)^I &:= \{k \in \Delta^I \mid \text{Für alle } k' \text{ mit } (k,k') \in \pi^I \text{ gilt } k' \in {\phi_1}^I\}
\end{aligned}$$

Beispiel 5.22 Wir betrachten die Interpretation I mit $\Delta^I = \{k_i \mid i \in \mathbb{N}\}$ und $p^I = \{(k_i, k_{i+1}) \mid i \in \mathbb{N}\}$, $A_1^I = \{k_i \mid i \in \mathbb{N} \text{ und } i \text{ ist ungerade}\}$, die wir wie folgt darstellen

$$I: \quad \underset{\neg {A_1}^I}{k_0} \xrightarrow{p^I} \underset{{A_1}^I}{k_1} \xrightarrow{p^I} \underset{\neg {A_1}^I}{k_2} \xrightarrow{p^I} \underset{{A_1}^I}{k_3} \xrightarrow{p^I} \underset{\neg {A_1}^I}{k_4} \xrightarrow{p^I} \ldots$$

und folgende Formel

$$\phi: \quad \neg A_1 \wedge [p^*]((A_1 \Rightarrow \langle p\rangle\neg A_1) \wedge (\neg A_1 \Rightarrow \langle p\rangle A_1)).$$

In dieser Interpretation I ist $\phi^I = \{k_0, k_2, k_4, \ldots\}$.

Definition 5.23

1. Die Konfigurationsformel ϕ heißt *erfüllbar*, falls es eine Interpretation I und ein $k \in \Delta^I$ gibt mit $k \in \phi^I$. I heißt dann *Modell* von ϕ.
2. Die Konfigurationsformel ϕ heißt *gültig*, falls für alle Interpretationen I gilt $\phi^I = \Delta^I$.
3. ϕ und ψ sind *äquivalent* genau dann, wenn $\phi \Leftrightarrow \psi$ gültig ist, d.h. für alle Interpretationen I gilt $\phi^I = \psi^I$.

Offenbar ist ϕ gültig genau dann, wenn $\neg\phi$ unerfüllbar ist.

Um Erfüllbarkeit (und damit auch Gültigkeit und Äquivalenz) zu entscheiden, ordnen wir jeder Konfigurationsformel ϕ einen Büchi–Automaten $\mathcal{A}_\phi$ zu, für den gilt:

$$\phi \text{ ist erfüllbar gdw } L_\omega(\mathcal{A}_\phi) \neq \emptyset.$$

Bei S1S wurde das Alphabet des Automaten durch die vorhandenen einstelligen Prädikatssymbole bestimmt (diese entsprechen den atomaren Eigenschaften). Für 1DPDL genügt dies nicht: Wir müssen alle "Unterformeln" der gegebenen Formel betrachten.

Zunächst stellen wir dazu jede Programmformel als eine reguläre Sprache endlicher Wörter (über dem Alphabet $\Sigma = \{p\}$) dar:

- $L(\epsilon) := \{\epsilon\}$,
- $L(p) := \{p\}$,
- $L(\pi_1; \pi_2) := L(\pi_1) \cdot L(\pi_2)$,
- $L(\pi_1 \cup \pi_2) := L(\pi_1) \cup L(\pi_2)$,
- $L(\pi_1{}^*) := L(\pi_1)^*$.

Beispiel: $L(p \cup (p;p)) = \{p, pp\}$, $L((p;p)^*) = \{\epsilon, pp, pppp, pppppp, \dots\}$

Wir definieren für eine Sprache $L(\pi) \subseteq p^*$ und eine Interpretation I:

$$L(\pi)^I := \{(k,k') \mid \begin{array}{l}\text{es gibt } m \geq 0,\ k_0, \dots, k_m \in \Delta^I \text{ mit} \\ p^m \in L(\pi), k_0 = k, k_m = k' \text{ und} \\ (k_i, k_{i+1}) \in p^I \text{ für } 0 \leq i < m\}.\end{array}$$

Offensichtlich gilt $(k,k') \in L(\pi)^I$ gdw $(k,k') \in \pi^I$, also $L(\pi)^I = \pi^I$.

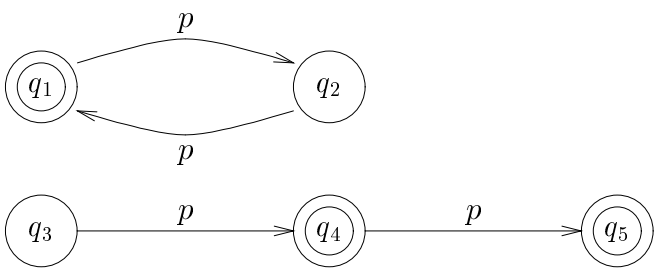
Es sei $\mathcal{B} = (Q, \{p\}, \cdot, \Delta, F)$ ein endlicher Automat, bei dem die Anfangszustandsmenge noch nicht festgelegt ist. Für jedes $q \in Q$ bezeichne $\mathcal{B}_q$ den Automaten, der aus $\mathcal{B}$ entsteht, wenn man $\{q\}$ als Menge von Anfangszuständen verwendet.

Lemma 5.24 Es sei ϕ_0 eine Konfigurationsformel, die (direkt zwischen $\langle\rangle$ oder $[]$) die Programmformeln $\pi_1, \dots, \pi_r$ enthält (d.h. Teilformeln dieser Programmformeln werden nicht betrachtet). Dann gibt es einen endlichen Automaten $\mathcal{B} = (Q, \{p\}, \cdot, \Delta, F)$, dessen Größe linear in der Größe von ϕ_0 ist, so daß gilt: Für alle i, $1 \leq i \leq r$, gibt es $q_i \in Q$ mit $L(\pi_i) = L(\mathcal{B}_{q_i})$.

Die Programmformel π_i wird also eindeutig durch den Zustand q_i beschrieben. Wir gehen daher im folgenden davon aus, daß in ϕ_0 anstelle der π_i die Zustände q_i des entsprechenden Automaten stehen.

Beispiel 5.25 $\phi_0 = [(p;p)^*](\langle p \cup (p;p)\rangle A_1)$

$\mathcal{B}$:



$L((p;p)^*) = L(\mathcal{B}_{q_1})$, $L(p \cup (p;p)) = L(\mathcal{B}_{q_3})$

ϕ_0 kann daher als $\widehat{\phi_0} = [q_1](\langle q_3\rangle A_1)$ geschrieben werden.

Es sei nun ϕ_0 eine Konfigurationsformel, $\mathcal{B}$ der zugehörige Automat und $\widehat{\phi_0}$ die Formel, die aus ϕ_0 entsteht, indem man jede Programmformel π_i in ϕ_0 durch den zugehörigen Zustand q_i ersetzt.

Interpretationen können auch auf Formeln mit Zuständen angewendet werden: Dazu definieren wir $q^I := L(\mathcal{B}_q)^I$ für alle Zustände q von $\mathcal{B}$. Offensichtlich ist damit ${\phi_0}^I = \widehat{\phi_0}^I$.

Wir können ohne Einschränkung davon ausgehen, daß ϕ_0 und $\widehat{\phi_0}$ in Negationsnormalform (NNF) sind, d.h. Negation kommt nur unmittelbar vor atomaren Eigenschaften vor. Man schiebt dazu mit Hilfe der folgenden äquivalenzerhaltenden Regeln Negation soweit wie möglich nach innen:

$$\neg(\phi \vee \psi) \to \neg\phi \wedge \neg\psi, \quad \neg(\phi \wedge \psi) \to \neg\phi \vee \neg\psi, \quad \neg\neg\phi \to \phi,$$
$$\neg\langle\pi\rangle\phi \to [\pi]\neg\phi, \qquad \neg[\pi]\phi \to \langle\pi\rangle\neg\phi.$$

Definition 5.26 Sei ϕ_0 eine Konfigurationsformel und Δ die Übergangsrelation des zugehörigen Automaten. Der *Fischer–Ladner–Abschluß* von ϕ_0 ist die kleinste Menge $FL(\phi_0)$ von Konfigurationsformeln, für die gilt:

- $\widehat{\phi_0} \in FL(\phi_0)$,
- $\phi_1 \wedge \phi_2 \in FL(\phi_0) \rightsquigarrow \phi_1, \phi_2 \in FL(\phi_0)$,
- $\phi_1 \vee \phi_2 \in FL(\phi_0) \rightsquigarrow \phi_1, \phi_2 \in FL(\phi_0)$,
- $\neg A_i \in FL(\phi_0)$ gdw $A_i \in FL(\phi_0)$,
- $[q]\phi \in FL(\phi_0) \rightsquigarrow$ $\phi \in FL(\phi_0)$ und $[q']\phi \in FL(\phi_0)$ für alle $q' \in Q$ mit $(q, p, q') \in \Delta$,
- $\langle q\rangle\phi \in FL(\phi_0) \rightsquigarrow$ $\phi \in FL(\phi_0)$ und $\langle q'\rangle\phi \in FL(\phi_0)$ für alle $q' \in Q$ mit $(q, p, q') \in \Delta$.

Beachte: $|FL(\phi_0)|$ ist linear in der Größe von ϕ_0.

Beispiel 5.25 (Fortsetzung) $\widehat{\phi_0} = [q_1](\langle q_3\rangle A_1)$ ist in NNF.

$$FL(\phi_0) = \{\widehat{\phi_0}, [q_2](\langle q_3\rangle A_1), \langle q_3\rangle A_1, \langle q_4\rangle A_1, \langle q_5\rangle A_1, A_1, \neg A_1\}.$$

Es sei nun ϕ_0 eine Konfigurationsformel in NNF und I ein Modell von ϕ_0 mit $k_0 \in {\phi_0}^I$. Beginnend mit k_0 wenden wir nun das Programm p iteriert an. Es können zwei Fälle eintreten.

1. p terminiert stets, d.h. es gibt eine unendliche Folge $k_0, k_1, k_2, \ldots \in \Delta^I$ mit $(k_i, k_{i+1}) \in p^I$ für $i \geq 0$.
2. p terminiert irgendwann nicht, d.h. es gibt eine endliche Folge $k_0, k_1, \ldots, k_m \in \Delta^I$ mit $(k_i, k_{i+1}) \in p^I$ für $0 \leq i < m$ und $(k_m, k) \notin p^I$ für alle $k \in \Delta^I$.

Für $k \in \Delta^I$ sei nun

$$S(k) := \{\phi \in FL(\phi_0) \mid k \in \phi^I\}.$$

Beachte: $S(k) \neq \emptyset$, da für alle atomaren Eigenschaften A_i gilt: $A_i \in S(k)$ oder $\neg A_i \in S(k)$.

Im ersten Fall definiert S ein unendliches Wort $\alpha(k_0) = S(k_0)S(k_1)S(k_2)\cdots$ über dem Alphabet $\Sigma := 2^{FL(\phi_0)}$. Im zweiten Fall verlängern wir das endliche Wort $S(k_0)S(k_1)\cdots S(k_m)$ zu einem unendlichen Wort $\alpha(k_0) \in \Sigma^\omega$ durch Anfügen von $\emptyset\emptyset\emptyset\emptyset\cdots$. Man erhält so ein Wort aus Σ^ω, das Eigenschaften hat, die sehr stark von der Struktur von $\widehat{\phi_0}$ abhängen.

Definition 5.27 Das Wort $\alpha = \alpha_0\alpha_1\alpha_2\cdots \in \Sigma^\omega$ ist ein *Hintikka–Wort* für ϕ_0, falls für alle $m \geq 0$ und für alle Formeln $\phi, \psi, \langle q\rangle\phi, [q]\phi, \phi \vee \psi, \phi \wedge \psi \in FL(\phi_0)$ gilt:

1. $\widehat{\phi_0} \in \alpha_0$.
2. $\alpha_m = \emptyset$ oder $A_i \in \alpha_m$ gdw $\neg A_i \notin \alpha_m$ für alle atomaren Eigenschaften $A_i \in FL(\phi_0)$.
3. $\phi \wedge \psi \in \alpha_m$ gdw $\phi \in \alpha_m$ und $\psi \in \alpha_m$.
4. $\phi \vee \psi \in \alpha_m$ gdw $\phi \in \alpha_m$ oder $\psi \in \alpha_m$.
5. Ist $[q]\phi \in \alpha_m$, dann gilt
   (a) falls $\epsilon \in L(\mathcal{B}_q)$, so $\phi \in \alpha_m$ und
   (b) $\alpha_{m+1} = \emptyset$ oder für alle q' mit $(q, p, q') \in \Delta$ gilt $[q']\phi \in \alpha_{m+1}$.
6. Ist $\langle q\rangle\phi \in \alpha_m$, dann gibt es ein $r \geq 0$ und Zustände $q_1, \ldots, q_r$ in $\mathcal{B}$, wobei q_r ein Endzustand ist, mit $q = q_0 \xrightarrow{p}_{\mathcal{B}} q_1 \xrightarrow{p}_{\mathcal{B}} \cdots \xrightarrow{p}_{\mathcal{B}} q_r$ (also $p^r \in L(\mathcal{B}_q)$), $\phi \in \alpha_{m+r}$ und $\langle q_i\rangle\phi \in \alpha_{m+i}$ für $0 \leq i < r$.
7. Ist $\alpha_m = \emptyset$, so auch $\alpha_{m+1} = \emptyset$.

Satz 5.28 Die Formel ϕ_0 ist erfüllbar genau dann, wenn es ein Hintikka–Wort für ϕ_0 gibt.

Beweis:

"$\rightsquigarrow$" Es sei I ein Modell für ϕ_0 und k_0 so, daß $k_0 \in {\phi_0}^I$. Dann ist $\alpha(k_0) = S(k_0)S(k_1)S(k_2)\ldots$ ein Hintikka–Wort für ϕ_0:

1. Wegen $k_0 \in {\phi_0}^I = \widehat{\phi_0}^I$ folgt $\widehat{\phi_0} \in S(k_0) = \{\phi \in FL(\phi_0) \mid k_0 \in \phi^I\}$.

2.–4. sind trivial.

5. Sei $[q]\phi \in S(k_m)$, d.h. $k_m \in ([q]\phi)^I$. Da I ein Modell ist, gilt für alle $p^r \in L(\mathcal{B}_q)$: ist $(k_m, k) \in (p^r)^I$, so $\phi \in S(k)$.

   5.(a) $\epsilon = p^0 \in L(\mathcal{B}_q) \rightsquigarrow \phi \in S(k_m)$, da $(k_m, k_m) \in \epsilon^I$.

   5.(b) Es sei $S(k_{m+1}) \neq \emptyset$ und $(q, p, q') \in \Delta$. Wir müssen $[q']\phi \in S(k_{m+1})$ zeigen, d.h. $k_{m+1} \in ([q']\phi)^I$, also für alle k' mit $(k_{m+1}, k') \in (p^{r'})^I$ und $p^{r'} \in L(\mathcal{B}_{q'})$ gilt $\phi \in S(k')$. Aus $(k_m, k_{m+1}) \in p^I$ und $(q, p, q') \in \Delta$ folgt $(k_m, k') \in (p^{r'+1})^I$ und $p^{r'+1} \in L(\mathcal{B}_q)$. Daraus folgt $k' \in \phi^I$, d.h. $\phi \in S(k')$.

6. Sei $\langle q\rangle\phi \in S(k_m)$, d.h. $k_m \in (\langle q\rangle\phi)^I$. Es gibt daher ein $p^r \in L(\mathcal{B}_q)$ mit $(k_m, k_{m+r}) \in (p^r)^I$ und $\phi \in S(k_{m+r})$. Wegen $p^r \in L(\mathcal{B}_q)$ gibt es $q_1, \ldots, q_r$ (wobei q_r ein Endzustand ist) mit $q = q_0 \xrightarrow{p}_{\mathcal{B}} q_1 \xrightarrow{p}_{\mathcal{B}} \cdots \xrightarrow{p}_{\mathcal{B}} q_r$. Daraus folgt $\langle q_i\rangle\phi \in S(k_{m+i})$ für $0 \leq i < r$.

7. ist trivial

"$\rightsquigarrow$" Es sei $\alpha = \alpha_0\alpha_1\alpha_2\cdots$ ein Hintikka–Wort für ϕ_0. Wir definieren das Modell I wie folgt:

$$\begin{aligned} \Delta^I &= \{k_0, k_1, k_2, \ldots\}, \\ p^I &= \{(k_i, k_{i+1}) \mid \alpha_{i+1} \neq \emptyset\}, \\ {A_j}^I &= \{k_i \mid A_j \in \alpha_i\}. \end{aligned}$$

Man zeigt nun:

$(*)$ Für alle $i \geq 0$ und alle $\phi \in FL(\phi_0)$ gilt: $\phi \in \alpha_i \rightsquigarrow k_i \in \phi^I$.

Wegen $\widehat{\phi_0} \in \alpha_0$ folgt daraus dann unmittelbar $k_0 \in \widehat{\phi_0}^I$, d.h. $k_0 \in {\phi_0}^I$.

Beweis von $(*)$: Induktion über den Formelaufbau

i) $\phi = A_j$: $(*)$ gilt nach Definition von ${A_j}^I$.

ii) $\phi = \neg A_j$: wegen 2. in Definition 5.27 ist daher $A_j \notin \alpha_i$ und somit nach Definition von ${A_j}^I$ auch $k_i \notin {A_j}^I$, d.h. $k_i \in (\neg A_j)^I$.

iii) $\phi = \phi_1 \wedge \phi_2$: wegen 3. gilt $\phi_1 \in \alpha_i$ und $\phi_2 \in \alpha_i$. Die Induktionsannahme liefert $k_i \in {\phi_1}^I$ und $k_i \in {\phi_2}^I$, was $k_i \in (\phi_1 \wedge \phi_2)^I$ liefert.

iv) $\phi = \phi_1 \vee \phi_2$ entsprechend.

v) $\phi = [q]\phi_1$: Es sei $p^r \in L(\mathcal{B}_q)$ und $(k_i, k) \in (p^r)^I$. Wir zeigen $k \in {\phi_1}^I$ durch Induktion über r:

$r = 0$: $p^r = \epsilon \in L(\mathcal{B}_q)$ liefert mit 5.(a), daß $\phi_1 \in \alpha_i$. Die äußere Induktionsannahme (über den Formelaufbau von ϕ) liefert daher $k = k_i \in {\phi_1}^I$.

$r \to r+1$: Ist $p^{r+1} \in L(\mathcal{B}_q)$, so gibt es ein q' mit $(q, p, q') \in \Delta$ und $p^r \in L(\mathcal{B}_{q'})$. Wegen $(k_i, k) \in (p^{r+1})^I$ folgt $(k_i, k_{i+1}) \in p^I$ und damit $\alpha_{i+1} \neq \emptyset$. Mit 5.(b) ist daher $[q']\phi_1 \in \alpha_{i+1}$. Außerdem gilt $(k_{i+1}, k) \in (p^r)^I$ und $p^r \in L(\mathcal{B}_{q'})$. Mit Induktionsannahme für r folgt $k \in {\phi_1}^I$.

vi) $\phi = \langle q \rangle \phi_1$: mit 6. existiert $p^r \in L(\mathcal{B}_q)$ und $\phi \in \alpha_{m+r}$. Wegen 7. gilt für $0 \leq j < r$: $\alpha_{m+j} \neq \emptyset$ und somit $(k_m, k_{m+r}) \in (p^r)^I$. Induktion liefert $k_{m+r} \in {\phi_1}^I$. ∎

Gesucht ist nun ein Büchi–Automat, der genau die Hintikka–Wörter für ϕ_0 akzeptiert. Zuerst definieren wir den *lokalen Automaten* $\mathcal{A}_L(\phi_0)$, der eine etwas größere Menge von Wörtern akzeptiert. Bei diesem ist die globale Bedingung 6. in Definition 5.27 ersetzt durch die lokale Bedingung

6'. $\langle q \rangle \phi \in \alpha_m$, dann gilt

(a) $\epsilon \in L(\mathcal{B}_q)$ und $\phi \in \alpha_m$ oder

(b) es gibt ein q' mit $(q, p, q') \in \Delta$ und $\langle q' \rangle \phi \in \alpha_{m+1}$.

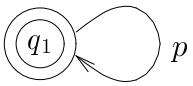
Definition 5.29 Der (nicht–deterministische) *lokale Automat* zu ϕ_0 ist definiert durch $\mathcal{A}_L(\phi_0) = (Q_L, \Sigma, I_L, \Delta_L, F_L)$ mit

$$
\begin{aligned}
Q_L &= \{\alpha_i \in 2^{FL(\phi_0)} \mid \alpha_i \text{ erfüllt 2., 3. und 4. von Definition 5.27}\},\\
\Sigma &= 2^{FL(\phi_0)},\\
I_L &= \{\alpha_i \in Q_L \mid \widehat{\phi_0} \in \alpha_i\},\\
\Delta_L &= \{(\alpha_m, \sigma, \alpha_{m+1}) \mid \bullet\ \alpha_m = \sigma \text{ und}\\
&\qquad\qquad \bullet\ \alpha_m \text{ und } \alpha_{m+1} \text{ erfüllen 5., 6'. und 7.}\},\\
F_L &= Q_L.
\end{aligned}
$$

Offenbar akzeptiert $\mathcal{A}_L(\phi_0)$ genau die Wörter $\alpha_0\alpha_1\alpha_2\cdots \in \Sigma^\omega$, die 1.–5. und 7. von Definition 5.27 sowie 6'. erfüllen.

Beispiel 5.30 Aus $\alpha \in L_\omega(\mathcal{A}_L(\phi_0))$ folgt noch nicht, daß α ein Hintikka–Wort für ϕ_0 ist: $\phi_0 = \langle p^* \rangle (A_1 \wedge \neg A_1)$

$\mathcal{B}$:



$\widehat{\phi_0} = \langle q_1 \rangle (A_1 \wedge \neg A_1)$. Es gilt für $\alpha_0 = \{\widehat{\phi_0}, A_1\}$, daß $\alpha_0^\omega \in L_\omega(\mathcal{A}_L(\phi_0))$. Das Wort α_0^ω erfüllt zwar 6'., aber nicht 6., d.h. es ist kein Hintikka–Wort für ϕ_0.

Das Problem ist hier, daß bei 6'. stets die Alternative 6'.(b) gewählt wird. Damit 6. erfüllt ist, muß aber irgendwann einmal 6'.(a) gewählt werden. Um dies zu erzwingen, wird der lokale Automat erweitert zum *Hintikka–Automaten*. In diesem sind die Zustände Paare, die in der ersten Komponente den gewohnten Zustand des lokalen Automaten enthalten und die in der zweiten Komponente aus der Menge der noch zu erfüllenden $\langle\rangle$–Formeln bestehen.

Definition 5.31 Der *Hintikka–Automat* zu ϕ_0 ist definiert durch $\mathcal{A}_H(\phi_0) = (Q_H, \Sigma, I_H, \Delta_H, F_H)$ mit

$$\begin{aligned} Q_H &= Q_L \times 2^{FL_{\langle\rangle}(\phi_0)}, \text{ wobei } FL_{\langle\rangle}(\phi_0) = \{\langle q\rangle\phi \mid \langle q\rangle\phi \in FL(\phi_0)\}, \\ \Sigma &= 2^{FL(\phi_0)}, \\ I_H &= I_L \times \{\emptyset\}, \\ F_H &= Q_L \times \{\emptyset\} \end{aligned}$$

und $((\alpha_m, s), \sigma, (\alpha_{m+1}, t)) \in \Delta_H$ genau dann, wenn

entweder

- $s = \emptyset$ und
- $(\alpha_m, \sigma, \alpha_{m+1}) \in \Delta_L$ und
- t enthält für jedes $\langle q\rangle\phi \in \sigma$ mit ($\epsilon \notin L(\mathcal{B}_q)$ oder $\phi \notin \sigma$) ein $\langle q'\rangle\phi \in \alpha_{m+1}$ mit $(q, p, q') \in \Delta$

oder

- $s \neq \emptyset$ und
- $(\alpha_m, \sigma, \alpha_{m+1}) \in \Delta_L$ und
- t enthält für jedes $\langle q \rangle \phi \in s$ mit ($\epsilon \notin L(\mathcal{B}_q)$ oder $\phi \notin \sigma$) ein $\langle q' \rangle \phi \in \alpha_{m+1}$ mit $(q, p, q') \in \Delta$.

Beachte: Nach Definition von Δ_L enthält α_{m+1} für $\epsilon \notin L(\mathcal{B}_q)$ oder $\phi \notin \sigma$ auch stets ein geeignetes $\langle q' \rangle \phi$, da 6'., durch Δ_L erfüllt wird.

Ein Wort wird von $\mathcal{A}_H$ nur dann akzeptiert, wenn die zweite Komponente der Zustände immer wieder leer wird, d.h. alle auftretenden $\langle\rangle$–Formeln werden irgendwann abgearbeitet. Immer dann, wenn die zweite Komponente leer ist, werden beim nächsten Übergang alle $\langle\rangle$–Formeln aus dem aktuellen Buchstaben σ in die zweite Komponente übernommen. Danach werden diese Formeln dann nach und nach nur noch entfernt, bis die Menge wieder leer ist.

Satz 5.32 Der Automat $\mathcal{A}_H(\phi_0)$ akzeptiert genau die Hintikka–Wörter für ϕ_0.

Beweis: Da der Hintikka–Automat auf dem lokalen Automat aufbaut und somit 1. bis 5. und 7. von Definition 5.27 erfüllt sind, genügt es, für alle $\alpha \in L_\omega(\mathcal{A}_L(\phi_0))$ zu zeigen:

$$\alpha \in L_\omega(\mathcal{A}_L(\phi_0)) \text{ gdw } \alpha \text{ erfüllt 6.}$$

Beachte, daß das Eingabewort $\alpha = \alpha_0\alpha_1\alpha_2\ldots$ auch die Folge der ersten Komponenten der Zustände beschreibt.

"$\rightsquigarrow$" Wir wissen aus der Definition, daß immer, wenn $s_m = \emptyset$ in einem Zustand (α_m, s_m) gilt, alle Formeln der Form $\langle q \rangle \phi \in \alpha_m$ in s_{m+1} übernommen werden. Da es wegen der Akzeptanzbedingung des Automaten einen Folgezustand (α_{m+k}, s_{m+k}) mit $s_{m+k} = \emptyset$ gibt, gibt es dazwischen für jede Formel $\langle q \rangle \phi \in s_m$ einen Zustand, in dem die Formel mit 6.'(a) abschließend abgearbeitet wird, d.h. 6. wird erfüllt. Wir müssen nun

aber noch darauf achten, daß auch die $\langle\rangle$–Formeln, die zwischendurch in einem α_{m+i} auftreten, erfüllt werden. Diese werden wegen der Definition des lokalen Automaten entweder mit 6'.(a) auch vor dem Zustand (α_{m+k}, s_{m+k}) abschließend abgearbeitet, oder sie treten in der Form $\langle q'\rangle\phi$ in α_{m+k} auf und werden dann in s_{m+k+1} übernommen. Auch diese Formeln erfüllen also 6.

Formal beweisen wir dies wie folgt. Es sei $\langle q\rangle\phi \in \alpha_m$. Da α von $\mathcal{A}_H(\phi_0)$ akzeptiert wird, gibt es ein $k \geq 1$, sodaß beim Übergang $(\alpha_{m+k}, s_{m+k}) \xrightarrow{\alpha_{m+k}} (\alpha_{m+k+1}, s_{m+k+1})$ gilt: $s_{m+k} = \emptyset$.

1. Fall Es gibt $r \leq k$ mit $p^r \in L(\mathcal{B}_q)$ und $\phi \in \alpha_{m+r}$. Dann ist 6. erfüllt.

2. Fall Für alle $r \leq k$ ist $p^r \notin L(\mathcal{B}_q)$ oder $\phi \notin \alpha_{m+r}$. Dann muß bei den Δ_L–Übergängen für $\langle q\rangle\phi$ und die daraus abgeleiteten Formeln stets 6'.(b) gelten. Es gibt daher ein q' mit $q \xrightarrow{p^k}_{\mathcal{B}} q'$ und $\langle q'\rangle\phi \in \alpha_{m+k}$. Da $s_{m+k} = \emptyset$ und laut Voraussetzung $\epsilon \notin L(\mathcal{B}_{q'})$ oder $\phi \notin \alpha_{m+k}$, gibt es wegen der Definition von Hintikka–Automaten ein q_1 mit $q' \xrightarrow{p}_{\mathcal{B}} q_1$, $\langle q_1\rangle\phi \in \alpha_{m+k+1}$ und $\langle q_1\rangle\phi \in s_{m+k+1}$.

Nun bleibt noch zu zeigen, daß alle Formeln in s_{m+k+1} irgendwann mit 6'.(a) abschließend abgearbeitet werden. Angenommen, es gibt kein $r > k$ mit $p^r \in L(\mathcal{B}_q)$ und $\phi \in \alpha_{m+r}$. Dann muß weiterhin stets Fall 6'.(b) auftreten. Es gibt daher einen Pfad $q_1 \xrightarrow{p}_{\mathcal{B}} q_2 \xrightarrow{p}_{\mathcal{B}} q_3 \xrightarrow{p}_{\mathcal{B}} \ldots$, so daß $\langle q_i\rangle\phi \in \alpha_{m+k+i}, \langle q_i\rangle\phi \in s_{m+k+i}$ für alle $i \geq 1$. Damit kann aber s_{m+i} nie leer werden, im Widerspruch dazu, daß der Pfad α akzeptiert wird.

"$\rightsquigarrow$" Kommt $\langle q'\rangle\phi$ nach dem Lesen von α_m in die zweite Komponente eines Zustands von $\mathcal{A}_H(\phi_0)$, so ist $\langle q'\rangle\phi \in \alpha_{m+1}$. Wegen 6. gibt es $p^r \in L(\mathcal{B}_{q'})$ mit $\phi \in \alpha_{m+1+r}$. Daher kann man den Pfad in $\mathcal{A}_H$ so wählen, daß nach spätestens r Schritten keine aus $\langle q'\rangle\phi$ entstandene $\langle\rangle$–Formel mehr in der zweiten Komponente liegt.

Die endlich vielen $\langle\rangle$–Formeln, die in einem Schritt in die zweite Komponente gekommen sind, werden also nach endlicher Zeit alle entfernt (erst danach wird die zweite Komponente wieder gefüllt), d.h. die zweite Komponente wird stets nach endlich vielen Schritten wieder leer.

■

Satz 5.33 Erfüllbarkeit in 1DPDL ist in exponentieller Zeit entscheidbar.

Beweis: Der Automat $\mathcal{A}_H(\phi_0)$ ist exponentiell in der Größe von ϕ_0 (weil bei der Konstruktion Teilmengen von $FL(\phi_0)$ benutzt werden). Das Leerheitsproblem für Büchi–Automaten ist in linearer Zeit entscheidbar. ∎

Anders als bei S1S wurde bei 1DPDL der für Büchi–Automaten sehr aufwendige Abschluß unter Komplement nicht verwendet (sehr aufwendig heißt hier schlimmer als exponentiell).

Häufig möchte man nicht nur *ein* atomares Programm zur Verfügung haben, sondern endlich viele Programme $p_1, \ldots, p_n$. Dann reicht die Betrachtung von Wörtern nicht aus. Stattdessen entsprechen die Modelle dann Hintikka–Bäumen.

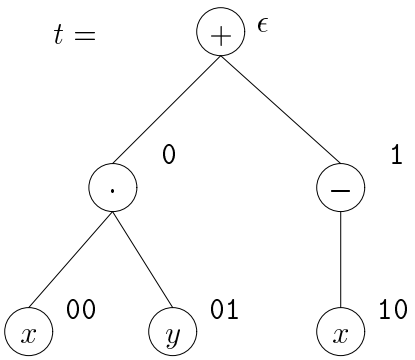
Kapitel 6

Automaten auf endlichen Bäumen

Wie wir im letzten Kapitel gesehen haben, gibt es Entscheidbarkeitsfragen, zu deren Beantwortung eine Reduktion auf Sprachen oder ω–Sprachen nicht ausreicht. Das sind z.B. Fragen nach der Erfüllbarkeit von Formeln der propositionalen dynamischen Logik, wenn wir mehr als ein atomares Programm zur Bildung der Programmformeln zulassen. Daher betrachten wir nun Bäume mit beschrifteten Knoten, bei denen die Anzahl der Nachfolgerknoten durch die Stelligkeit der Knotenbeschriftung gegeben ist. Automaten werden nun statt Wörtern oder ω–Wörtern endliche oder unendliche Bäume akzeptieren.

6.1 Endliche Bäume

Beispiel 6.1 $\Sigma = \{+, \cdot, -, x, y\}$ sei das Alphabet der Knotenbeschriftung, wobei $+$ und $\cdot$ zweistellig, $-$ einstellig und x, y nullstellig sind.



Der nebenstehende Baum t ist ein endlicher, Σ–beschrifteter Baum. Man kann die Knoten des Baums eindeutig durch Wörter über dem Alphabet $\{0, 1\}$ ansprechen, da die maximale Stelligkeit zwei ist. Dabei steht 0 für links und 1 für rechts. Der Baum t entspricht daher einer partiellen Funktion

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

mit Definitionsbereich $\mathrm{dom}(t) = \{\epsilon, 0, 1, 00, 01, 10\}$, und z.B. ist $t(0) = \cdot$.

Definition 6.2 Es sei Σ ein Alphabet und $\nu : \Sigma \to \omega$ eine Funktion, die jedem $a \in \Sigma$ eine *Stelligkeit* $\nu(a)$ zuordnet. Für $n \in \omega$ sei $\Sigma_n = \{a \in \Sigma \mid \nu(a) = n\}$. Ein Σ–*Baum* ist eine partielle Funktion $t : \omega^* \to \Sigma$, deren Definitionsbereich $\mathrm{dom}(t)$ erfüllt:

1. $\epsilon \in \mathrm{dom}(t)$,
2. für alle $u \in \omega^*$ und $i \in \omega$ gilt:

$$ui \in \mathrm{dom}(t) \text{ gdw } u \in \mathrm{dom}(t) \text{ und } i < \nu(t(u)).$$

Erklärung: Die erste Bedingung bedeutet, daß jeder Baum zumindest einen Knoten hat. Die zweite fordert, daß zu jedem Knoten $\neq \epsilon$ in t ein Vorgängerknoten existieren muß, und daß der Knoten u mit $\nu(t(u)) = n$ die Nachfolgerknoten $u0, u1, \ldots, u(n-1)$ hat.

Ein *Blatt* von t ist ein Knoten $u \in \mathrm{dom}(t)$ mit $\nu(t(u)) = 0$, d.h. u hat keine Nachfolgerknoten. Der Baum ist *endlich*, falls $\mathrm{dom}(t)$ endlich ist. Mit T_Σ bezeichnen wir die Menge aller endlichen Σ–Bäume. Teilmengen von T_Σ heißen dann *Baumsprachen*. Es sei $\prec$ die *Präfixrelation* auf ω^*, d.h.

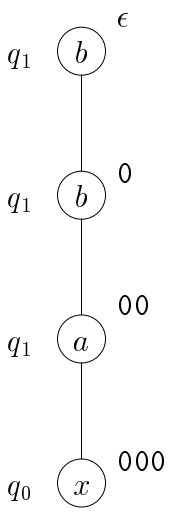
$$u \prec v \quad \text{gdw} \quad \text{es gibt } u' \in \omega^+ \text{ mit } uu' = v.$$

Wegen 2. in Definition 6.2 gilt für jeden Baum t, daß $\mathrm{dom}(t)$ unter Präfixbildung abgeschlossen ist, d.h. ist $v \in \mathrm{dom}(t)$ und $u \prec v$, so ist auch $u \in \mathrm{dom}(t)$.

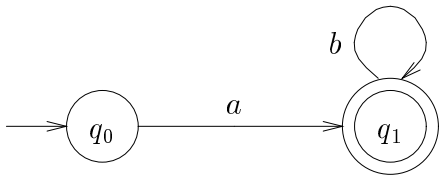
Definition 6.3 Es sei $t \in T_\Sigma$.

1. Ein *Pfad durch* t ist eine (bezüglich $\prec$) totalgeordnete, maximale Teilmenge von $\mathrm{dom}(t)$.
2. Der *Unterbaum* von t an der Stelle $u \in \mathrm{dom}(t)$ ist der Baum t_u mit
   - $\mathrm{dom}(t_u) = \{v \mid uv \in \mathrm{dom}(t)\}$,
   - $t_u(v) = t(uv)$.

Beispiel: In Beispiel 6.1 sind $\{\epsilon, \mathtt{0}, \mathtt{00}\}, \{\epsilon, \mathtt{0}, \mathtt{01}\}, \{\epsilon, \mathtt{1}, \mathtt{10}\}$ alle Pfade. $\{\epsilon, \mathtt{00}\}$ ist nicht maximal, und $\{\epsilon, \mathtt{0}, \mathtt{00}, \mathtt{01}\}$ ist nicht total geordnet.



Ein Wort $w \in \Sigma^*$ für ein endliches Alphabet Σ kann als endlicher Baum über $\widehat{\Sigma} = \Sigma \cup \{x\}$ angesehen werden, falls man $\nu(a) = 1$ für $a \in \Sigma$ und $\nu(x) = 0$ definiert. Zum Beispiel entspricht $abb \in \{a, b\}^*$ dann dem nebenstehenden Baum. Das Zusatzsymbol x wird benötigt, damit der Baum mit einem Blatt abgeschlossen werden kann. Das Wort abb wird z.B. von folgendem Automaten erkannt:



Die Verarbeitung eines mit $u \in \Sigma^*$ beschrifteten Pfades durch einen endlichen Automaten kann durch eine zusätzliche Markierung der Knoten mit Zuständen des Automaten (hier q_0 und q_1) in dem mit u assoziierten Baum beschrieben werden. Die Markierung in dem mit abb assoziierten Baum beschreibt einen mit abb beschrifteten Pfad in dem oben gegebenen Automaten.

Dies kann auch auf Bäume mit Verzweigungsgrad größer als eins erweitert werden.

Definition 6.4 Ein *BW–Baumautomat*[1] $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ besteht aus

- einer endlichen Zustandsmenge Q,
- einem endlichen Alphabet Σ mit Stelligkeitsfunktion ν (d.h. $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \ldots \cup \Sigma_m$),
- einer Anfangszuweisung $I : \Sigma_0 \to 2^Q$,
- einer Übergangszuweisung Δ, die für $n > 0$ jedem $a \in \Sigma_n$ eine Funktion $\Delta_a : Q^n \to 2^Q$ zuweist,
- einer Endzustandsmenge $F \subseteq Q$.

Ein *Lauf* dieses Baumautomaten auf einem Baum $t \in T_\Sigma$ ist eine Abbildung $\ell : \mathrm{dom}(t) \to Q$, d.h. eine Abbildung, die den Knoten des Baumes Zustände des Automaten zuweist, mit

$$\ell(u) \in \Delta_a(\ell(u0), \ldots, \ell(u(n-1))) \text{ für } t(u) = a \in \Sigma_n.$$

Der Lauf ist *erfolgreich*, falls

- $\ell(u) \in I(t(u))$ für alle Blätter $u \in \mathrm{dom}(t)$,
- $\ell(\epsilon) \in F$.

Die *von $\mathcal{A}$ akzeptierte Baumsprache* ist

$$L(\mathcal{A}) = \{t \in T_\Sigma \mid \text{ Es gibt einen erfolgreichen Lauf von } \mathcal{A} \text{ auf } t\}.$$

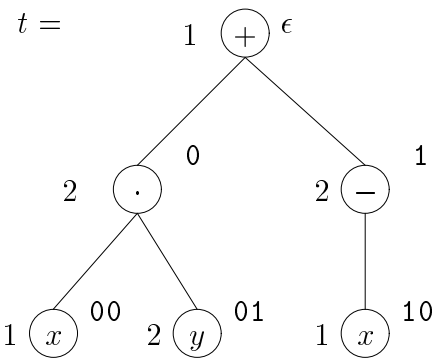
$\mathcal{A}$ heißt *deterministisch*, wenn für alle $a \in \Sigma_0$ gilt $|I(a)| = 1$ und für alle $n > 0, a \in \Sigma_n$ und $q_1, q_2, \ldots, q_n \in Q$ gilt $|\Delta_a(q_1, q_2, \ldots, q_n)| = 1$[2]. Wir schreiben I und Δ_a dann als Funktionen $I : \Sigma_0 \to Q$ und $\Delta_a : Q^{\nu(a)} \to Q$.

[1] "BW" steht für "von Blatt zu Wurzel" und wird weggelassen, falls der Zusammenhang klar ist.

[2] In diesem Skript sind deterministische Automaten auch immer vollständig. Bei nicht–vollständigen Automaten gilt hier $\ldots \leq 1$ statt $\ldots = 1$.

Beispiel 6.5 Es sei $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ mit $\Sigma_0 = \{x, y\}, \Sigma_1 = \{-\}, \Sigma_2 = \{+, \cdot\}$ und $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ mit

- $Q = \{0, 1, 2\}$,
- $I(x) = 1, I(y) = 2$,
- $\Delta_-(q) = (3 - q) \bmod 3$,
  $\Delta_+(q, q') = (q + q') \bmod 3$,
  $\Delta_\cdot(q, q') = (q \cdot q') \bmod 3$,
- $F = \{1\}$.



Nebenstehend ist ein erfolgreicher Lauf von $\mathcal{A}$ auf dem Baum t durch die Zustandsmarkierung aus $Q = \{0, 1, 2\}$ links an den Knoten abgebildet. Es gilt $t \in L(\mathcal{A})$. $L(\mathcal{A})$ besteht aus den arithmetischen Ausdrücken über $-, +, \cdot, x, y$, deren Auswertung modulo 3 mit $x = 1$ und $y = 2$ als Ergebnis 1 liefert.

Wie bei Automaten für endliche Wörter kann man zu jedem nicht–deterministischen BW–Automaten $\mathcal{A}$ durch Potenzmengenkonstruktion einen äquivalenten deterministischen BW–Automaten $\mathcal{A}'$ konstruieren.

Satz 6.6 Für eine Baumsprache $L \subseteq T_\Sigma$ sind äquivalent:

1. L wird von einem deterministischen BW–Automaten akzeptiert.
2. L wird von einem BW–Automaten akzeptiert.

Anstatt Bäume von den Blättern zur Wurzel zu "verarbeiten" kann man auch von der Wurzel zu den Blätter vorgehen.

Definition 6.7 Ein *WB–Baumautomat* $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ besteht aus

- einer endlichen Zustandsmenge Q,
- einem endlichen Alphabet Σ mit Stelligkeitsfunktion ν,
- einer Menge I von Anfangszuständen,
- einer Übergangszuweisung Δ, die für $n > 0$ jedem $a \in \Sigma_n$ eine Funktion $\Delta_a : Q \to 2^{Q^n}$ zuweist,
- einer Endzuweisung $F : \Sigma_0 \to 2^Q$.

Ein *Lauf* eines WB–Baumautomaten $\mathcal{A}$ auf einem Baum $t \in T_\Sigma$ ist eine Abbildung $\ell : \mathrm{dom}(t) \to Q$ mit

$$(\ell(u0), \ldots, \ell(u(n-1))) \in \Delta_a(\ell(u)) \text{ für } t(u) = a \in \Sigma_n.$$

Der Lauf ist *erfolgreich*, falls

- $\ell(\epsilon) \in I$,
- $\ell(u) \in F(t(u))$ für alle Blätter $u \in \mathrm{dom}(t)$.

Die *von* $\mathcal{A}$ *akzeptierte Baumsprache* ist

$$L(\mathcal{A}) = \{t \in T_\Sigma \mid \text{ Es gibt einen erfolgreichen Lauf von } \mathcal{A} \text{ auf } t\}.$$

$\mathcal{A}$ heißt *deterministisch*, wenn $|I| = 1$ gilt und für alle $n > 0, a \in \Sigma_n, q \in Q$ gilt: $|\Delta_a(q)| = 1$.

Satz 6.8 Für eine Baumsprache $L \subseteq T_\Sigma$ sind äquivalent:

1. L wird von einem BW–Automaten akzeptiert.
2. L wird von einem WB–Automaten akzeptiert.

Beweis:

"$1 \rightsquigarrow 2$" Es sei $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ein BW–Automat. Wir betrachten den WB–Automaten $\mathcal{B} = (Q, \Sigma, F, \Delta', I)$, wobei

$$\Delta'_a : q \mapsto \{(q_1, \ldots, q_n) \mid q \in \Delta_a(q_1, \ldots, q_n)\}.$$

Man sieht leicht, daß jeder Lauf (bzw. erfolgreiche Lauf) von $\mathcal{A}$ auf t auch ein Lauf (bzw. erfolgreicher Lauf) von $\mathcal{B}$ auf t ist und umgekehrt.

"$2 \rightsquigarrow 1$" entsprechend. ∎

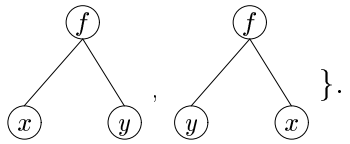
Beispiel 6.9 Es sei $\mathcal{A}$ der BW–Automat aus Beispiel 6.5. Der zugehörige WB–Automat ist dann $\mathcal{B} = (Q, \Sigma, I', \Delta', F')$ mit

- $Q = \{0, 1, 2\},\ \Sigma = \{x, y, -, +, \cdot\}$,
- $I' = \{1\}$ (die Endzustandsmenge von $\mathcal{A}$),
- $$\begin{aligned} \Delta_-(q) &= \{q' \in Q \mid \Delta_-(q') = q\} \\ &= \{q' \in Q \mid (3 - q') \bmod 3 = q\} = \{q' \in Q \mid (3 - q) \bmod 3 = q'\} \\ &= \{(3 - q) \bmod 3\}, \end{aligned}$$
- $\Delta_+(q) = \{(q', q'') \in Q^2 \mid q = (q' + q'') \bmod 3\}$,
- $\Delta_\cdot(q) = \{(q', q'') \in Q^2 \mid q = (q' \cdot q'') \bmod 3\}$,
- $F'(x) = 1,\ F'(y) = 2$ (die Anfangszuweisung von $\mathcal{A}$).

Der Lauf von $\mathcal{A}$ auf t in Beispiel 6.5 kann ebenso als erfolgreicher Lauf von $\mathcal{B}$ betrachtet werden.

Beachte: Obwohl $\mathcal{A}$ deterministisch war, ist $\mathcal{B}$ nicht–deterministisch. So ist z.B. $\Delta_+(1) = \{(0, 1), (1, 0), (2, 2)\}$. Das nächste Beispiel zeigt: Nicht jede von einem nicht–deterministischen WB–Automaten akzeptierte Sprache wird von einem deterministischen WB–Automaten akzeptiert.

Beispiel 6.10 Sei $\Sigma = \{x, y, f\}$ und

$L = \{$  $\}$.

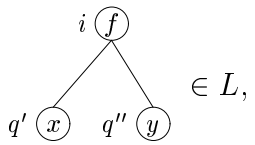
L wird von dem WB–Automaten

$$\mathcal{A} = (\underbrace{\{q_0, q_1, q_x, q_y\}}_{Q}, \Sigma, \underbrace{\{q_0, q_1\}}_{I}, \Delta, F)$$

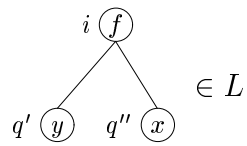
akzeptiert mit

$$\begin{aligned} \Delta_f(q_0) &= \{(q_x, q_y)\}, \\ \Delta_f(q_1) &= \{(q_y, q_x)\}, \\ \Delta_f(q_x) &= \Delta_f(q_y) = \emptyset, \\ F(x) &= \{q_x\}, \\ F(y) &= \{q_y\}. \end{aligned}$$

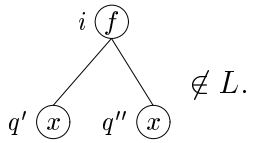
Angenommen, L wird von einem deterministischen WB–Automaten $\mathcal{B} = (Q', \Sigma, \{i\}, \Delta', F')$ akzeptiert. Dann gilt für $(q', q'') = \Delta'_f(i)$ wegen

 $\in L$,

daß $q' \in F(x)$ ist. Wegen

 $\in L$

gilt aber auch $q'' \in F(x)$. Daher akzeptiert $\mathcal{B}$ auch

 $\notin L$.

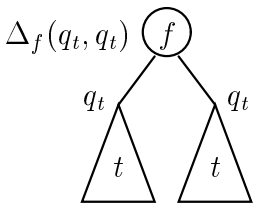
Deterministische und nicht–deterministische BW–Automaten sowie nicht–deterministische WB–Automaten akzeptieren also dieselbe Sprachklasse, während deterministische WB–Automaten eine kleinere Klasse von Sprachen akzeptieren.

Definition 6.11 Eine Baumsprache $L \subseteq T_\Sigma$ heißt *erkennbar*, wenn sie von einem BW–Automaten akzeptiert wird.

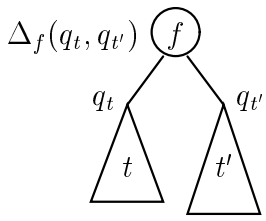
Beispiel 6.12 Nicht jede Baumsprache ist erkennbar. Es sei dazu Σ ein Alphabet mit Stelligkeitsfunktion, für das $|\Sigma_0| > 0$ und $f \in \Sigma_2$. Offensichtlich folgt daraus, daß T_Σ unendlich ist. Wir zeigen, $L = \{f(t,t) \mid t \in T_\Sigma\} = \{t' \in T_\Sigma \mid t'(\epsilon) = f \text{ und } t'_0 = t'_1\}$ ist nicht erkennbar.

Angenommen, $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ist ein (ohne Einschränkung) deterministischer BW–Automat für L. Für jeden Baum $t \in T_\Sigma$ sei q_t der (eindeutig bestimmte) Zustand aus Q für den gilt: Bei dem Lauf r auf t mit $I(t(u)) = r(u)$ für alle Blätter u wird die Wurzel mit q_t beschriftet (d.h. $r(\epsilon) = q_t$). Da Q endlich und T_Σ unendlich ist, gibt es $t, t' \in Q$ mit $t \neq t'$ und $q_t = q_{t'}$. Betrachte den Lauf von $\mathcal{A}$ auf

(a)



(b)



der mit $I(\cdot)$ an den Blättern beginnt. Wegen (a) ist $\Delta_f(q_t, q_t) \in F$, wegen $q_t = q_{t'}$ ist $\Delta_f(q_t, q_{t'}) = \Delta_f(q_t, q_t) \in F$, also wird auch $f(t,t')$ von $\mathcal{A}$ akzeptiert. Da aber $f(t,t') \notin L$ ist, wird L von keinem deterministischen BW–Automaten akzeptiert und ist deshalb nicht erkennbar.

6.2 Reguläre Baumsprachen

Bei Wörtern können erkennbare Sprachen durch reguläre Ausdrücke beschrieben werden. Bei erkennbaren Baumsprachen ist eine ähnliche Charakterisierung möglich. Dazu müssen wir geeignete Operationen auf Baumsprachen

einführen und zeigen, daß die erkennbaren Baumsprachen unter diesen Operatoren abgeschlossen sind.

Zur Erinnerung: Bei Wörtern gilt: $\emptyset \in Reg_\Sigma$, $\{a\} \in Reg_\Sigma$ für alle $a \in \Sigma$, $L_1, L_2 \in Reg_\Sigma \rightsquigarrow L_1 \cup L_2, L_1 \cdot L_2, {L_1}^* \in Reg_\Sigma$.

Satz 6.13

1. Die leere Baumsprache ist erkennbar.
2. Für $a \in \Sigma_0$ ist der Baum $\textcircled{a}$ erkennbar.

Beweis:

1. Verwende BW–Automat mit $F = \emptyset$.
2. Der folgende BW–Automat leistet das Gewünschte:
   - $Q = \{0, 1\}$
   - $I(a) = 1$, $I(b) = 0$ für alle $b \in \Sigma_0 \setminus \{a\}$
   - $\Delta_f(q_1, \ldots, q_n) = \emptyset$ für alle $f \in \Sigma_n, n > 0$
   - $F = \{1\}$

Satz 6.14 Die Klasse der erkennbaren Baumsprachen ist abgeschlossen unter Vereinigung, Durchschnitt und Komplement.

Beweis: (Wie bei Wörtern)

1. Vereinigung: vereinige zwei BW–Automaten (mit disjunkten Zustandsmengen) zu einem nicht–deterministischen BW–Automaten. (Übung)
2. Komplement: verwende deterministische BW–Automaten und mache Endzustände zu Nicht–Endzuständen und umgekehrt. ∎

Anstelle der Konkatenation von Wörtern verwenden wir *Baumkonkatenation*. Einen Baum mit Wurzelbeschriftung $f \in \Sigma_n$ und direkten Unterbäumen $t_1, \ldots, t_n$ schreiben wir auch $f(t_1, \ldots, t_n)$.

Definition 6.15 Es sei Σ ein Alphabet mit Stelligkeitsfunktion und $\overline{x} = (x_1, \ldots, x_k)$ ein k–Tupel verschiedener Elemente von Σ_0.

1. Für $t \in T_\Sigma$ und $L_1, \ldots, L_k \subseteq T_\Sigma$ wird $t \cdot^{\overline{x}} (L_1, \ldots, L_k) \subseteq T_\Sigma$ induktiv definiert durch:
   - $t \in \Sigma_0$: falls $t = \textcircled{$x_i$}$: $t \cdot^{\overline{x}} (L_1, \ldots, L_k) := L_i$
     sonst ($t \neq \textcircled{$x_i$}$ für alle i) : $t \cdot^{\overline{x}} (L_1, \ldots, L_k) := \{t\}$
   - $t = f(t_1, \ldots, t_n)$ für $f \in \Sigma_n, n > 0$:
     $t \cdot^{\overline{x}} (L_1, \ldots, L_k) := \{f(t'_1, \ldots, t'_n) \mid t'_i \in t_i \cdot^{\overline{x}} (L_1, \ldots, L_k)\}$.
2. Für $L, L_1, \ldots, L_k \subseteq T_\Sigma$ ist
$$L \cdot^{\overline{x}} (L_1, \ldots, L_k) := \bigcup_{t \in L} t \cdot^{\overline{x}} (L_1, \ldots, L_k).$$

Ein Baum in $L \cdot^{\overline{x}} (L_1, \ldots, L_k)$ entsteht aus einem Baum $t \in L$ also dadurch, daß man jedes Blatt mit Beschriftung x_i durch einen Baum aus L_i ersetzt.

Beispiel: Seien $a, b \in \Sigma_0$ und $L = \{a, b\}$. Dann gilt

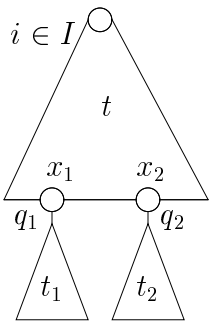
$$\{f(x, x)\} \cdot^x L = \{f(a, a), f(a, b), f(b, a), f(b, b)\}.$$

Beachte:

1. Verschiedene Vorkommen von x_i können durch verschiedene Elemente L_i ersetzt werden. Insbesondere ist daher:
$$\begin{aligned} \{f(x, x)\} \cdot^x T_\Sigma &= \{f(t, t') \mid t, t' \in T_\Sigma\} \\ &\neq \{f(t, t) \mid t \in T_\Sigma\}. \end{aligned}$$
2. Die Reihenfolge, mit der diese Konkatenationen ausgeführt werden, ist wichtig! Dies wird an einem Beispiel $(L_1 \cdot^x L_2) \cdot^y L_3 \neq L_1 \cdot^x (L_2 \cdot^y L_3)$ klar, wenn in L_1 auch Blätter y vorkommen.

Satz 6.16 Sind $L, L_1, L_2, \ldots, L_k$ erkennbare Baumsprachen und $\overline{x} = (x_1, \ldots, x_k)$, so ist auch $L \cdot^{\overline{x}} (L_1, \ldots, L_k)$ erkennbar.

Beweis:
Die Idee des Beweises ist, $t \in L \cdot^{\overline{x}} (L_1, \ldots, L_k)$ mit dem WB–Automaten für L solange zu verarbeiten, bis wir an Blätter x_i von $t' \in L$ gelangen. Die Entscheidung, wann dies der Fall ist, wird nicht-deterministisch getroffen. Notwendige Bedingung ist, daß ein $q_i \in F(x_i)$ erreicht ist. Die daran hängenden Teilbäume werden dann mit den Automaten für L_i weiterverarbeitet.



Es sei $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ein WB–Automat für L und $\mathcal{A}_i = (Q^{(i)}, \Sigma, I^{(i)}, \Delta^{(i)}, F^{(i)})$ seien WB–Automaten für L_i $(1 \leq i \leq k)$. Ohne Einschränkung sind die Zustandsmengen $Q, Q^{(1)}, \ldots, Q^{(k)}$ paarweise disjunkt. Betrachten wir nun

$$\mathcal{B} = (Q \cup Q^{(1)} \cup \ldots \cup Q^{(k)}, \Sigma, I, \Delta', F')$$

mit

- für $a \in \Sigma_n$ mit $n > 0$:
  - für $q \in Q^{(j)}$ $(1 \leq j \leq k)$: $\Delta'_a(q) = \Delta^{(j)}_a(q)$,
  - für $q \in Q$: $\Delta'_a(q) = \Delta_a(q) \cup \bigcup\limits_{\substack{1 \leq j \leq k \\ q \in F(x_j)}} \{\Delta^{(j)}_a(i) \mid i \in I^{(j)}\}$,
- für $a \in \Sigma_0$:
  - für $a \notin \{x_1, \ldots, x_k\}$:

$$\begin{aligned} F'(a) \;=\; & F(a) \cup F^{(1)}(a) \cup F^{(2)}(a) \cup \ldots \cup F^{(k)}(a) \cup \\ & \{q \in Q \mid q \in F(x_j) \text{ mit } 1 \leq j \leq k \text{ und} \\ & \qquad F^{(j)}(a) \cap I^{(j)} \neq \emptyset\}, \end{aligned}$$

    (Die letzte Teilmenge ist für die aus L_j angehängten Teilbäume, die nur aus einer Wurzel bestehen.)

– für $a \in \{x_1, \ldots, x_k\}$:

$$\begin{aligned} F'(a) &= F^{(1)}(a) \cup F^{(2)}(a) \cup \ldots \cup F^{(k)}(a) \cup \\ &\quad \{q \in Q \mid q \in F(x_j) \text{ mit } 1 \leq j \leq k \text{ und} \\ &\qquad F^{(j)}(a) \cap I^{(j)} \neq \emptyset\}. \end{aligned}$$

Man sieht leicht, daß $\mathcal{B}$ ein WB–Automat für $L \cdot^{\overline{x}} (L_1, \ldots, L_k)$ ist. ∎

Bei der Definition der regulären Baumsprachen verwenden wir zwei Spezialfälle der allgemeinen Baumkonkatenation:

1. Das Anwenden eines $f \in \Sigma_k$, $k > 0$ auf Baumsprachen $L_1, \ldots, L_k$ ist definiert durch
$$f(L_1, \ldots, L_k) = f(x_1, \ldots, x_k) \cdot^{(x_1, \ldots, x_k)} (L_1, \ldots, L_k).$$
2. $k = 1$, also $L \cdot^x L'$, d.h. wir ersetzen die Blätter x in $t \in L$ durch Bäume aus L'.

Auch der Sternoperator kann auf Baumsprachen erweitert werden: Intuitiv ist dann ein Baum in $L^{*,x}$, wenn er sich in eine endliche Kette von Bäumen aus L zerlegen läßt (mit Blatt–Wurzelbindeglied x).

Definition 6.17 Es sei $L \subseteq T_\Sigma$ und $x \in \Sigma_0$. Wir definieren den *Sternoperator auf Baumsprachen*:

$$\begin{aligned} L^{0,x} &= \{x\}, \\ L^{n+1,x} &= L^{n,x} \cup L \cdot^x L^{n,x}, \\ L^{*,x} &= \bigcup_{n \geq 0} L^{n,x}. \end{aligned} \tag{6.1}$$

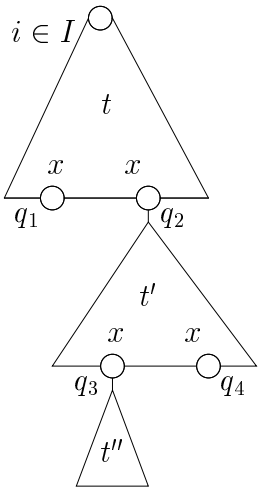
Man macht sich am Beispiel $L = \{f(x, x)\}$ leicht klar, daß durch Weglassen von "$L^{n,x} \cup$" in Gleichung 6.1 eine andere Menge $L^{*,x}$ definiert wird: Die so definierte Baumsprache wäre die Menge aller Bäume, bei denen auf allen Pfaden von der Wurzel zu einem Blatt x dieselbe Anzahl von Bäumen aus L durchlaufen würde.

Der nächste Satz vervollständigt die Sätze über Abschlußeigenschaften erkennbarer Baumsprachen unter regulären Operatoren.

Satz 6.18 Ist L eine erkennbare Baumsprache, so auch $L^{*,x}$.

Beweis:

Es sei $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ein WB–Automat für L. Die Idee des Beweises ist, daß man einen Automaten konstruiert, der bei $q_i \in F(x)$ entweder aufhören kann oder aber mit einem Anfangszustand weitermacht.



Es sei $\mathcal{B} = (Q \cup \{\hat{\imath}\}, \Sigma, I', \Delta', F')$ ein (nicht–deterministischer) WB–Automat für $L^{*,x}$ mit $\{\hat{\imath}\} \notin Q$ und

- $I' = I \cup \{\hat{\imath}\}$,
- für $a \in \Sigma_n$ mit $n > 0$:

$$\Delta'_a(q) = \Delta_a(q) \cup \begin{cases} \emptyset & q \notin F(x) \\ \bigcup_{i \in I} \Delta_a(i) & q \in F(x), \end{cases}$$

- für $a \in \Sigma_0$, $a \neq x$:

$$F'(a) = F(a) \cup \{q \in Q \mid q \in F(x) \text{ und } F(a) \cap I \neq \emptyset\},$$

- für $a = x$: $F'(x) = F(x) \cup \{\hat{\imath}\}$.

Der zusätzliche Anfangs– und Endzustand $\hat{\imath}$ sorgt dafür, daß auch $\textcircled{x}$ akzeptiert wird. Nach Konstruktion wird $L^{*,x}$ von $\mathcal{B}$ akzeptiert. ∎

Definition 6.19 Es sei Σ ein Alphabet mit Stelligkeitsfunktion, Z eine Menge null–stelliger Symbole, $\Sigma \cap Z = \emptyset$ und $\widehat{\Sigma} = \Sigma \cup Z$. Die Klasse der *regulären Baumsprachen* $\mathcal{R}eg(T_\Sigma, Z)$ ist die kleinste Klasse von Baumsprachen über $\widehat{\Sigma}$ mit:

1. $\emptyset \in \mathcal{R}eg(T_\Sigma, Z)$,
2. $\{x\} \in \mathcal{R}eg(T_\Sigma, Z)$ für alle $x \in \Sigma_0 \cup Z$,
3. $L_1, L_2 \in \mathcal{R}eg(T_\Sigma, Z) \rightsquigarrow L_1 \cup L_2 \in \mathcal{R}eg(T_\Sigma, Z)$,
4. $L_1, L_2 \in \mathcal{R}eg(T_\Sigma, Z)$ und $z \in Z \rightsquigarrow L_1 \cdot^z L_2 \in \mathcal{R}eg(T_\Sigma, Z)$,
5. $L \in \mathcal{R}eg(T_\Sigma, Z)$ und $z \in Z \rightsquigarrow L^{*,z} \in \mathcal{R}eg(T_\Sigma, Z)$,
6. $n > 0, f \in \Sigma_n, L_1, \ldots, L_n \in \mathcal{R}eg(T_\Sigma, Z) \rightsquigarrow f(L_1, \ldots L_n) \in \mathcal{R}eg(T_\Sigma, Z)$.

Eine Sprache $L \subseteq T_\Sigma$ heißt *regulär*, falls es ein Hilfsalphabet Z von nullstelligen Symbolen gibt, so daß $L \in \mathcal{R}eg(T_\Sigma, Z)$ gilt. [3]

Satz 6.20 Für $L \subseteq T_\Sigma$ sind äquivalent:

1. L ist regulär.
2. L ist erkennbar.

Beweis:

"$1 \rightsquigarrow 2$" Ergibt sich unmittelbar aus den bereits gezeigten Abschlußeigenschaften erkennbarer Baumsprachen.

[3] Beachte: $\mathcal{R}eg(T_\Sigma, Z) \not\subseteq T_\Sigma$. Zur Konstruktion der regulären Baumsprachen sind in $\mathcal{R}eg(T_\Sigma, Z)$ auch Sprachen mit den zusätzlichen Hilfssymbole Z enthalten, die nicht in Σ enthalten sind.

"$2 \rightsquigarrow 1$" Es sei $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ein WB–Automat mit $L = L(\mathcal{A})$. Ohne Einschränkung sei $Q = \{1, \ldots, k\}$ und $Q \cap \Sigma = \emptyset$. Wir definieren $Z := Q$ (und geben jedem $q \in Z$ Stelligkeit 0). Wir definieren nun einen WB–Automaten $\mathcal{A}'$, zu diesem eine endliche Menge von Baumsprachen und werden zeigen, daß L sich als Vereinigung dieser Baumsprachen schreiben läßt.

Es sei $\mathcal{A}' = (Q, \Sigma \cup Z, I, \Delta, F')$ mit $F'(q) = \{q\}$ für alle $q \in Z$, $F'(a) = F(a)$ für alle $a \in \Sigma_0$. Für $K \subseteq Q$, $i \in Q$ und $0 \leq h \leq k$ sei $L(K, h, i)$ die Menge aller Bäume $t \in T_{\Sigma \cup K}$, für die es einen Lauf ℓ von $\mathcal{A}'$ auf t gibt mit:

- $\ell(\epsilon) = i$,
- $\ell(u) \leq h$ für alle $u \neq \epsilon$, die nicht Blätter sind,
- $\ell(u) \in F'(t(u))$ für alle Blätter u.

$L(K, h, i)$ enthält demnach Bäume, die zusätzlich Blätter haben können, die mit $q \in K$ markiert sind. Ein Lauf von $\mathcal{A}'$ darauf muß mit i an der Wurzel beginnen, an den Blättern mit einem Zustand $u \in F'(t(u))$ enden und darf dazwischen nur Zustände $\leq h$ verwenden. Für $t(u) = q$ ist $F'(t(u)) = \{a\}$. Im Unterschied zu einem Lauf von $\mathcal{A}$ kann hier nun "vorzeitig" abgebrochen werden. Offensichtlich gilt:

$$L(\mathcal{A}) = \bigcup_{i \in I} L(\emptyset, k, i).$$

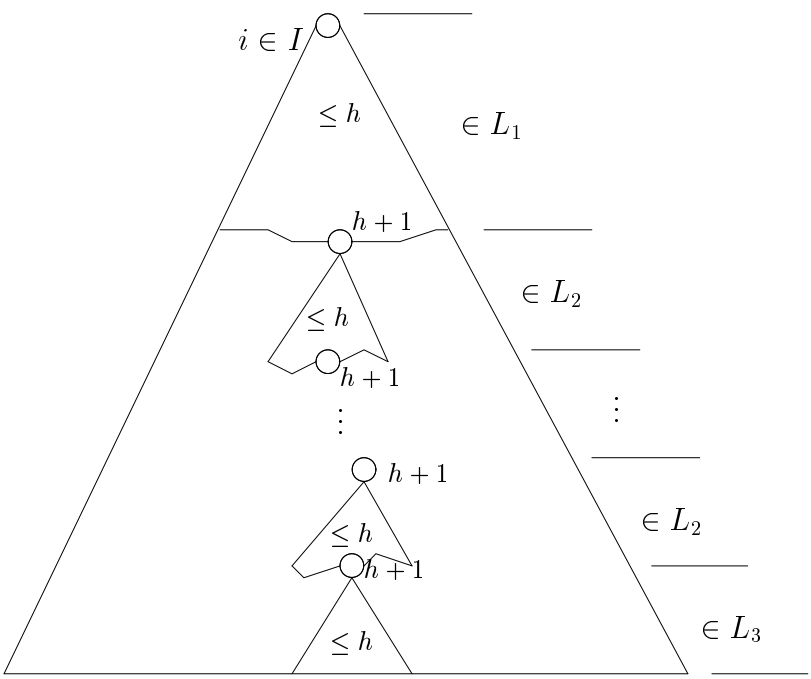
Es genügt also zu zeigen, daß alle $L(K, h, i)$ in $\mathcal{R}eg(T_\Sigma, Z)$ sind. Dies zeigen wir durch Induktion über h.

Induktionsanfang $h = 0$: In $L(K, 0, i)$ sind also keine Bäume enthalten, die einen Knoten u enthalten, der weder Blatt noch Wurzel ist (Der wäre sonst mit $q > 0$ markiert). Da $\Sigma \cup K$ endlich ist, ist auch $L(K, 0, i)$ endlich. Man zeigt leicht, daß jede endliche Teilmenge von $T_{\Sigma \cup Z}$ in $\mathcal{R}eg(T_\Sigma, Z)$ ist.

Induktionsschritt $h > 0$: Es gilt

$$\begin{aligned} L(K, h+1, i) \;=\; & L(K, h, i) \cup \\ & L(K \cup \{h+1\}, h, i) \cdot^{h+1} \\ & L(K \cup \{h+1\}, h, h+1)^{*, h+1} \cdot^{h+1} L(K, h, h+1). \end{aligned}$$

Den formalen Beweis dieser Gleichung lassen wir als Übung. Es sollte aber klar sein, daß $L(K,h,i) \subseteq L(K,h+1,i)$ gilt. Der Term der Form $L_1 \cdot^{h+1} L_2^{*,h+1} \cdot^{h+1} L_3$ erklärt sich intuitiv folgendermaßen (siehe auch nachfolgende Skizze): L_1 sind die Bäume, deren Wurzel mit i markiert sind, deren innere Knoten nur mit $j \leq h$ markiert sind und die Blätter aus $K \cup \{h+1\}$ haben können. Die Blätter $h+1$ werden dann durch Baum–Ketten aus L_2 ersetzt. Bäume aus L_2 haben eine mit $h+1$ markierte Wurzel und Blätter aus $K \cup \{h+1\}$. Zum Schluß kommt dann noch ein letzter Baum mit Wurzelmarkierung $h+1$, dessen innere Knoten aber nur mit $j \leq h$ markiert sind. Man sieht leicht, daß jeder Baum aus $L(K,h+1,i)$ in eine solche Kette von Bäumen zerlegt werden kann.



Nach Induktion und wegen der Definition von $\mathcal{R}eg(T_\Sigma, Z)$ liefert die obige Zerlegung von $L(K,h,i)$, daß $L(K,h,i) \in \mathcal{R}eg(T_\Sigma, Z)$ gilt. ∎

Eine weitere interessante Abschlußeigenschaft erkennbarer Baumsprachen ist der Abschluß unter *Alphabetumbenennung*. Es seien $\Sigma^{(1)}, \Sigma^{(2)}$ zwei Alphabe-

te mit Stelligkeitsfunktionen und $\phi : \Sigma^{(1)} \to \Sigma^{(2)}$ eine Abbildung, so daß $\phi(\Sigma_n^{(1)}) \subseteq \Sigma_n^{(2)}$ für alle $n \geq 0$. Für einen Baum $t : \mathrm{dom}(t) \to \Sigma^{(1)}$ aus $T_{\Sigma^{(1)}}$ ist $\phi(t) : \mathrm{dom}(t) \to \Sigma^{(2)}$ definiert durch $\phi(t)(u) = \phi(t(u))$, d.h. man ersetzt einfach (da ϕ stelligkeitserhaltend ist) Knoten σ durch Knoten $\phi(\sigma)$.

Satz 6.21 Ist $L \subseteq T_{\Sigma^{(1)}}$ erkennbar, so auch $\phi(L) = \{\phi(t) \mid t \in L\} \subseteq T_{\Sigma^{(2)}}$.

Beweis: Es sei $\mathcal{A} = (Q, \Sigma^{(1)}, I, \Delta, F)$ ein BW–Automat für L. Wir definieren $\mathcal{A}' = (Q, \Sigma^{(2)}, I', \Delta', F)$, wobei

- für $a \in \Sigma_0^{(2)}$:
$$I'(a) = \bigcup_{\substack{a' \in \Sigma_0^{(1)} \\ \phi(a') = a}} I(a'),$$
- für $a \in \Sigma_n^{(2)}$, $n > 0$:
$$\Delta'_a(q_1, \ldots, q_n) = \bigcup_{\substack{a' \in \Sigma_n^{(1)} \\ \phi(a') = a}} \Delta_{a'}(q_1, \ldots, q_n).$$

Man sieht leicht, daß $L(\mathcal{A}') = \phi(L)$ gilt. ∎

Als nächstes betrachten wir Entscheidbarkeitsprobleme für reguläre (d.h. erkennbare) Baumsprachen.

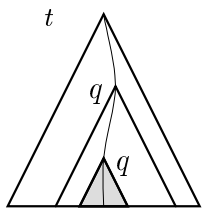
Satz 6.22 Für reguläre Baumsprachen sind das Äquivalenzproblem und das Leerheitsproblem entscheidbar.

Beweis: Da wir bereits wissen, daß reguläre Baumsprachen unter Booleschen Operationen abgeschlossen sind, kann man das Äquivalenzproblem auf das Leerheitsproblem reduzieren. Es sei $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ein BW–Automat mit $|Q| = k$ Zuständen.

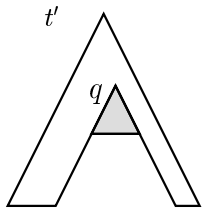
Behauptung: $L(\mathcal{A}) \neq \emptyset$ genau dann, wenn es einen Baum $t \in L(\mathcal{A})$ der Tiefe $\leq k$ gibt, wobei die Tiefe eines Baumes die Länge des längsten Pfades ist.

Da es für ein endliches Alphabet Σ nur endlich viele $t \in T_\Sigma$ der Tiefe $\leq k$ gibt, kann man für alle diese Bäume testen, ob sie von $\mathcal{A}$ akzeptiert werden.

Beweis der Behauptung:
Es sei $t \in L(\mathcal{A})$ ein Baum minimaler Größe (d.h. Gesamtzahl von Knoten). Angenommen, t hätte Tiefe größer k. Betrachte einen erfolgreichen Lauf ℓ von $\mathcal{A}$ auf t. Auf jedem Pfad mit Länge $> k$ gibt es zwei verschiedene Stellen u, u' im Baum, die mit demselben Zustand q markiert sind, d.h. $q := \ell(u) = \ell(u')$.



Ersetzt man in t den Unterbaum t_u durch den kleineren Unterbaum $t_{u'}$, so hat man auch auf dem so erhaltenen Baum t' einen erfolgreichen Lauf. Dies widerspricht der Minimalität von t. Es muß also auch einen Baum $t \in L(\mathcal{A})$ mit Tiefe $\leq k$ geben. ∎



Kapitel 7

Automaten auf unendlichen Bäumen

Auch bei Baumsprachen interessieren wir uns, ähnlich wie bei Sprachen über Wörtern, nun auch für unendliche Objekte. Wir werden deshalb nun unendliche Bäume definieren. Dies wird es uns unter anderem erlauben, Interpretationen der zugehörigen logischen Formeln mit unendlichem Interpretationsbereich zu betrachten.

Der Einfachheit halber betrachten wir hier nur *binäre unendliche Bäume*, d.h. wir gehen davon aus, daß wir ein Alphabet Σ mit Stelligkeitsfunktion haben, für das $\Sigma = \Sigma_2$ gilt. Alle Resultate lassen sich aber leicht auf den allgemeinen Fall übertragen.

Mit T_Σ^ω bezeichnen wir die Menge der unendlichen binären Bäume. Eine Teilmenge L von T_Σ^ω heißt *ω–Baumsprache* und ein $t \in T_\Sigma^\omega$ ω–Baum. Man kann ω–Baumsprachen aus Baumsprachen durch unendliche Iteration erzeugen.

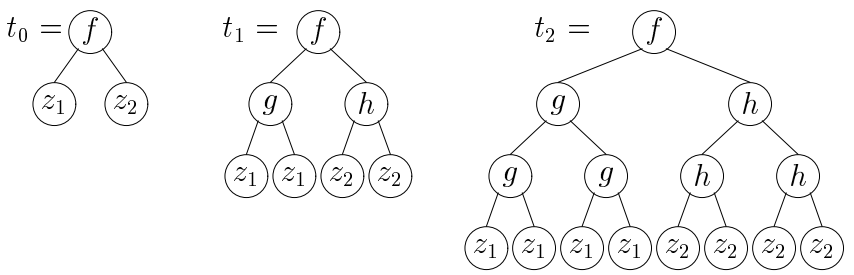
Definition 7.1 Es sei $Z = \{z_1, \ldots, z_k\}$ eine Menge nullstelliger Symbole und Σ ein Alphabet binärer Symbole. $U, U_1, \ldots, U_k \subseteq T_{\Sigma \cup Z}$ seien Baumsprachen über dem Alphabet $\Sigma \cup Z$. Die ω–Baumsprache

$$U \cdot^{(z_1,\ldots,z_k)} (U_1, \ldots, U_k)^{\omega,(z_1,\ldots,z_k)}$$

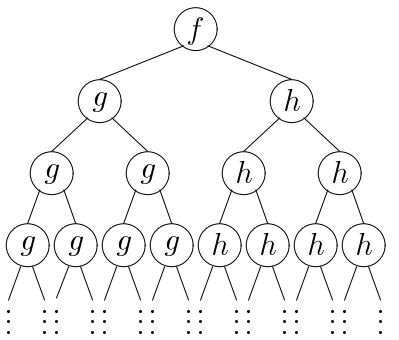
besteht aus allen ω–Bäumen $t \in T_\Sigma^\omega$, für die es eine Folge $t_0, t_1, t_2, \ldots$ von Bäumen aus $T_{\Sigma \cup Z}^\omega$ gibt mit:

1. $t_0 \in U$,
2. für alle $i \geq 0$ ist $t_{i+1} \in t_i \cdot^{(z_1,\ldots,z_k)} (U_1, \ldots, U_k)$,
3. t ist *Limes* dieser Folge, d.h. für alle $u \in \{0,1\}^*$ gibt es $m \geq 0$ mit:
   - $u \in \mathrm{dom}(t_m)$ und u kein Blatt, d.h. $t_m(u) \in \Sigma$,
   - $t(u) = t_m(u)$.

Beispiel 7.2 $Z = \{z_1, z_2\}, U = \{f(z_1, z_2)\}, U_1 = \{g(z_1, z_1)\}, U_2 = \{h(z_2, z_2)\}$



Der Limes der einzigen möglichen Folge ist:



7.1 Büchi– und Rabin–Baumautomaten

Da die betrachteten unendlichen Bäume keine Blätter haben, macht es nur Sinn, Baumautomaten zu betrachten, die bei der Wurzel beginnen. Es handelt sich daher um Verallgemeinerungen von WB–Automaten auf den unendlichen Fall.

Definition 7.3

1. Ein *Büchi–Baumautomat* über dem Alphabet Σ (mit $\Sigma = \Sigma_2$) ist von der Form $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$. Dabei sind
   - Q, I und Δ wie bei WB–Automaten definiert und
   - $F \subseteq Q$ eine Menge von Endzuständen.
2. Ein *Rabin–Baumautomat* über dem Alphabet Σ (mit $\Sigma = \Sigma_2$) ist von der Form $\mathcal{A} = (Q, \Sigma, I, \Delta, \Omega)$. Dabei sind
   - Q, I und Δ wie bei WB–Automaten definiert und
   - $\Omega = \{(F_1, G_1), \ldots, (F_n, G_n)\}$ eine endliche Menge von Paaren mit $F_i, G_i \subseteq Q$.

Ein *Lauf* ℓ eines (Büchi– oder Rabin–) Baumautomaten auf $t \in T^{\omega}_{\Sigma}$ ist wie im endlichen Fall definiert, d.h. $\ell : \mathrm{dom}(t) \to Q$ ($\mathrm{dom}(t) = \{\mathtt{0}, \mathtt{1}\}^*$) mit $(\ell(u\mathtt{0}), \ell(u\mathtt{1})) \in \Delta_f(\ell(u))$, falls $t(u) = f$. Ein Lauf ℓ ist also ein unendlicher Baum über dem Markierungsalphabet Q (wobei $\nu(q) = 2$ für alle $q \in Q$).

Ein Lauf ℓ eines Büchi–Automaten heißt *erfolgreich*, falls

- $\ell(\epsilon) \in I$ und
- jeder Pfad in ℓ enthält mindestens einen Zustand aus F unendlich oft.

Ein Lauf ℓ eines Rabin–Automaten heißt *erfolgreich*, falls

- $\ell(\epsilon) \in I$ und
- es für jeden Pfad in ℓ ein $(F_i, G_i) \in \Omega$ $(1 \leq i \leq n)$ gibt, so daß
  - der Pfad mindestens einen Zustand aus F_i unendlich oft enthält und
  - der Pfad keinen Zustand aus G_i unendlich oft enthält.

Die Akzeptanzbedingung für Büchi– bzw. Rabin–Automaten auf unendlichen Wörtern wird hier also auf Pfade im Baum angewendet. Beachte: Jeder Pfad in $t \in T_\Sigma^\omega$ liefert ein Wort aus Σ^ω.

Die von einem (Büchi– oder Rabin–) Baumautomaten *akzeptierte ω–Baumsprache* ist

$$L_\omega(\mathcal{A}) := \{t \in T_\Sigma^\omega \mid \text{Es gibt einen erfolgreichen Lauf von } \mathcal{A} \text{ auf } t\}.$$

$L \subseteq T_\Sigma^\omega$ heißt *Büchi–erkennbar* (*Rabin–erkennbar*), falls L von einem Büchi– (Rabin–) Baumautomaten akzeptiert wird.

Satz 7.4 Jede Büchi–erkennbare Sprache ist auch Rabin–erkennbar.

Beweis: Es sei $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ein Büchi–Automat mit $L = L_\omega(\mathcal{A})$. Wir definieren den Rabin–Automaten $\mathcal{A}' := (Q, \Sigma, I, \Delta, \{(F, \emptyset)\})$. Offensichtlich gilt $L_\omega(\mathcal{A}') = L_\omega(\mathcal{A}) = L$. ∎

Die folgenden beiden Beispiele sollen die Funktionsweise von Büchi– und Rabin–Automaten illustrieren.

Beispiel 7.5 Sei $\Sigma = \{a, b\}$ und $L_1 = \{t \in T_\Sigma^\omega \mid \text{Es gibt einen Pfad in } t, \text{ der unendlich viele } a \text{ enthält}\}$.

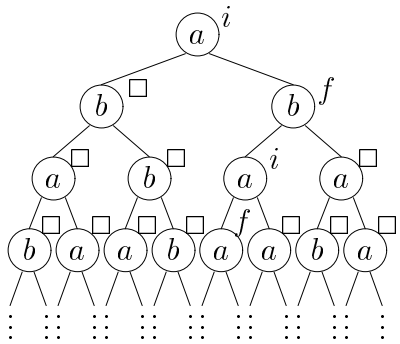
Der folgende nicht–deterministische Büchi–Automat für L_1 wählt (nichtdeterministisch) einen Pfad in t und nimmt jedesmal, nachdem dort a vorgekommen ist, den Endzustand f an. Bei den nicht ausgewählten Pfaden wird ein zweiter Endzustand $\Box$ stets reproduziert.

$$\mathcal{A}_1 = (Q_1, \Sigma, I_1, \Delta_1, F_1) \text{ mit } Q_1 = \{i, f, \Box\}, I_1 = \{i\}, F_1 = \{f, \Box\},$$

$$\Delta_{1a}: \begin{array}{l} i \mapsto \{(f, \Box), (\Box, f)\} \\ f \mapsto \{(f, \Box), (\Box, f)\} \\ \Box \mapsto \{(\Box, \Box)\} \end{array} \qquad \Delta_{1b}: \begin{array}{l} i \mapsto \{(i, \Box), (\Box, i)\} \\ f \mapsto \{(i, \Box), (\Box, i)\} \\ \Box \mapsto \{(\Box, \Box)\} \end{array}$$

Es gibt also genau einen Pfad, der nur mit Zuständen aus $\{i, f\}$ markiert wird. Alle anderen Pfade werden ab einer Stelle stets mit $\Box$ markiert. Der Endzustand f tritt nur unmittelbar unter einem a auf.

Gibt es einen Pfad mit unendlich vielen a, so kann man diesen wählen und erhält darauf im Lauf unendlich oft f. Alle anderen Pfade enthalten unendlich oft $\Box$. Wählt man einen Pfad, der a nur endlich oft enthält, so entsteht im Lauf ein Pfad, der nur endlich oft f enthält und $\Box$ überhaupt nicht.



Beispiel 7.6 Betrachten wir nun $\Sigma = \{a, b\}$ und $L_2 = \{t \in T_\Sigma^\omega \mid$ jeder Pfad in t enthält nur endlich viele a $\}$. Offensichtlich ist $L_2 = T_\Sigma^\omega \setminus L_1$ das Komplement von L_1. Wir geben einen Rabin–Baumautomaten für L_2 an: $\mathcal{A}_2 = (\{i, f\}, \Sigma, \{i\}, \Delta_2, \{(\{i, f\}, \underbrace{\{f\}}_{\substack{\text{endlich}\\ \text{oft}}})\})$ mit

$$\Delta_{2a}: \begin{array}{lcl} i & \mapsto & \{(f,f)\} \\ f & \mapsto & \{(f,f)\} \end{array} \qquad \Delta_{2b}: \begin{array}{lcl} i & \mapsto & \{(i,i)\} \\ f & \mapsto & \{(i,i)\}. \end{array}$$

Gibt es einen Pfad mit unendlich vielen a, so hat der zugehörige Pfad im Lauf unendlich viele Markierungen f, ist also nicht erfolgreich. Daß ein Zustand aus $\{i, f\}$ unendlich oft in jedem Pfad vorkommen muß, ist keine Einschränkung, da $\{i, f\}$ die gesamte Zustandsmenge ist.[1]

Satz 7.7 Die Sprache L_2 aus Beispiel 7.6 ist Rabin–erkennbar, aber nicht Büchi–erkennbar.

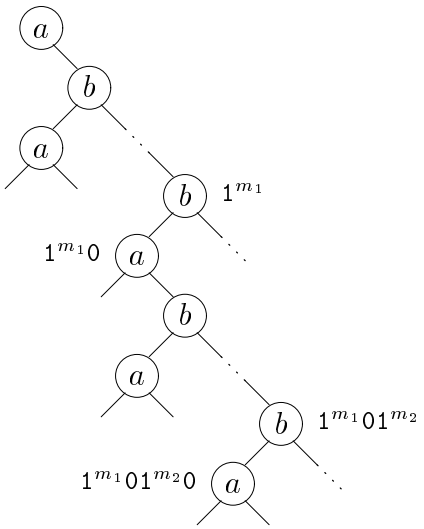
Beweis: Wir haben bereits gesehen, daß L_2 Rabin–erkennbar ist. Angenommen, sie ist auch Büchi–erkennbar und wird von $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ akzeptiert. Sei $n > |F|$. Definiere zu n den Baum $t^{(n)}$ folgendermaßen:

[1] Derselbe Automat mit der Endzustandsmenge $\Omega_2 = \{(\{i\}, \{f\})\}$ erkennt aber ebenfalls L_2.

$$
\begin{aligned}
U &= \{\epsilon\} \cup \\
&\quad \{\mathtt{1}^{m_1}\mathtt{0} \mid m_1 > 0\} \cup \\
&\quad \vdots \\
&\quad \{\mathtt{1}^{m_1}\mathtt{0}\mathtt{1}^{m_2}\mathtt{0}\cdots\mathtt{1}^{m_n}\mathtt{0} \mid m_i > 0 \text{ für alle } 1 \le i \le n\} \\
&= \{\epsilon\} \cup \bigcup_{1\le i\le n} (\mathtt{1}^{+}\mathtt{0})^i,
\end{aligned}
$$

$$
\begin{aligned}
t^{(n)} : \{\mathtt{0},\mathtt{1}\}^* &\to \Sigma \\
u &\mapsto \begin{cases} a & u \in U \\ b & u \notin U. \end{cases}
\end{aligned}
$$

Der Baum $t^{(n)}$ hat unendlich viele Knoten a, aber jeder Pfad hat höchstens $n+1$, d.h. $t^{(n)} \in L_2$.

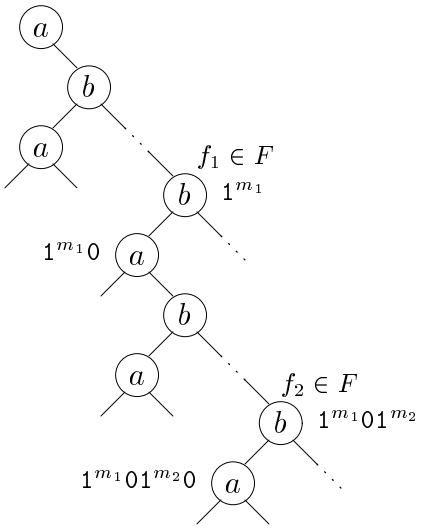


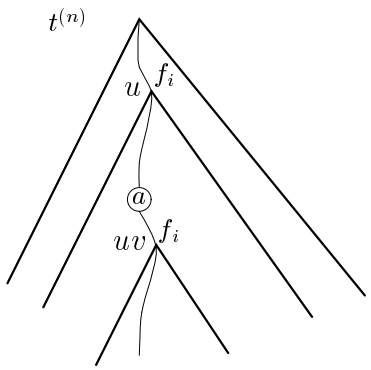
Allgemein gilt: Jeder Pfad beginnt mit a. Um ein weiteres a zu erreichen, muß man im Baum mindestens einmal nach rechts und dann genau einmal nach links gehen. Für das nächste a muß man wieder mindestens einmal nach rechts und dann genau einmal nach links gehen. Nachdem man n–mal nach links gegangen ist, hat man das letzte a auf diesem Pfad erreicht. Da $t^{(n)} \in L_2$ ist, gibt es einen erfolgreichen Lauf von $\mathcal{A}$ auf $t^{(n)}$.

Wir wählen nun einen Pfad wie folgt:

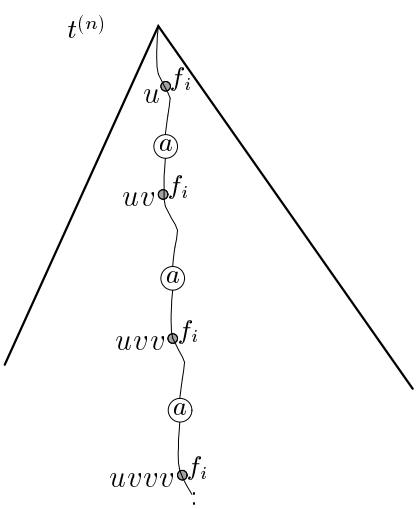
- Wähle ein minimales m_1 mit $\ell(\mathtt{1}^{m_1}) \in F$. So ein m_1 gibt es, da auf dem Pfad $\{\epsilon, \mathtt{1}, \mathtt{11}, \mathtt{111}, \ldots\}$ in ℓ unendlich oft ein Element aus F vorkommt. Sei $f_1 = \ell(\mathtt{1}^{m_1})$.

- Es seien $m_1, m_2, \ldots, m_i$ $(i < n)$ schon definiert. Wähle minimales m_{i+1} mit $\ell(1^{m_1}01^{m_2}0\cdots1^{m_i}01^{m_{i+1}}) \in F$. Sei $f_{i+1} = \ell(1^{m_1}01^{m_2}0\cdots1^{m_i}01^{m_{i+1}})$.





Dies definiert uns also $m_1, \ldots, m_n$ und einen Lauf wie in der linken Skizze. Da $n > |F|$ gewählt war, gibt es ein $i < j$ mit $f_i = f_j$. Wir haben daher die Situation wie in der rechten Skizze. Setzt man in $t^{(n)}$ an der Stelle uv den Baum $t_u^{(n)}$ (der Teilbaum von $t^{(n)}$ mit Wurzel u) ein, so erhält man einen Baum, der ebenfalls einen erfolgreichen Lauf besitzt.



Iteriert man dieses Vorgehen unendlich oft, so erhält man einen Baum t, der von $\mathcal{A}$ akzeptiert wird. Dieser Baum enthält aber auf einem Pfad unendlich viele a, da ja zwischen u und uv mindestens ein a vorkam. ∎

Korollar 7.8 Die Klasse der Büchi–erkennbaren Sprachen ist nicht unter Komplement abgeschlossen.

Beweis: L_1 ist Büchi–erkennbar, $L_2 = T_\Sigma^\omega \setminus L_1$ nicht. ∎

Satz 7.9 Die Klasse der Rabin–erkennbaren Baumsprachen ist unter Komplement abgeschlossen.

Der Beweis dieses Satzes ist *sehr* aufwendig und wird hier nicht geführt. Es gibt verschiedene Beweisansätze (für Literaturhinweise, siehe [Tho90a]). Spieltheoretische Ansätze haben sich hier wieder als sehr hilfreich erwiesen.

Warum ist dieses Problem viel schwerer als das für Automaten auf unendlichen Wörtern? Dies liegt an der Quantifizierung über Pfade in der Definition eines erfolgreichen Laufes:

"*Für alle* Pfade ist die Akzeptanzbedingung *erfüllt*."

Negiert man diese Aussage, so erhält man:

"*Es gibt* einen Pfad, der die Akzeptanzbedingung *nicht erfüllt*."

Man muß im Beweis der Abgeschlossenheit unter Komplement also nicht nur von "*nicht erfüllt*" zu "*erfüllt*" kommen, sondern auch noch von "*es gibt einen Pfad*" zu "*für alle Pfade*".

7.2 Entscheidbarkeitsresultate

Da wir wieder logische Erfüllbarkeitsprobleme auf das Leerheitsproblem der von Automaten akzeptierten Sprachen reduzieren möchten, interessieren wir uns zunächst für die Entscheidbarkeit der von Büchi– und Rabin–Baumautomaten akzeptierten ω–Baumsprachen. Beginnen wir mit den Büchi–Baumautomaten, da man hier eine zusätzliche Charakterisierung der von Büchi–Baumautomaten akzeptierten ω–Baumsprachen erhält.

Satz 7.10 Das Leerheitsproblem ist für Büchi–erkennbare ω–Baumsprachen entscheidbar.

Vorbereitung des Beweises: Es sei $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ein Büchi–Baumautomat, wobei $F = \{f_1, \ldots, f_m\}$ ist. Es sei $\ell : \{0,1\}^* \to Q$ ein erfolgreicher Lauf von $\mathcal{A}$ auf einem Baum $t \in T_\Sigma^\omega$. Der Baum t wird nun zerlegt in endliche Teilbäume, die dann von Automaten auf endlichen Bäumen akzeptiert werden. Dies liefert uns eine Zerlegung von $L_\omega(\mathcal{A}) \subseteq T_\Sigma^\omega$, die wir dann für den Beweis verwenden werden.

Es sei $u \in \{0,1\}^*$. Wir interessieren uns dafür, wo man in ℓ nach u wieder Endzustände erreicht:

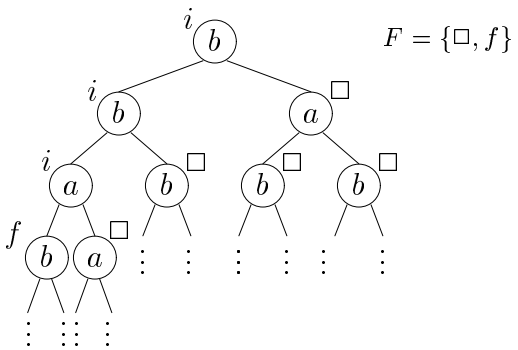
$$D_u := \{w \in \{0,1\}^* \mid \text{Für alle } v \text{ mit } \epsilon < v \leq w \text{ gilt } \ell(uv) \notin F\}.$$

Offensichtlich stellt D_u einen endlich verzweigten Baum dar, in dem es keine unendlichen Pfade gibt (sonst wäre ℓ ja nicht erfolgreich). Also ist D_u endlich. Es sei

$$D_u^+ := D_u \cup \{w\sigma \mid \sigma \in \{0,1\} \text{ und } w \in D_u \text{ und } w\sigma \notin D_u\}.$$

Nach Definition von D_u gilt $\ell(w) \in F$ für alle $w \in D_u^+ \setminus D_u$.

Beispiel: Betrachte einen Lauf des Büchi–Baumautomaten aus Beispiel 7.5 auf dem folgenden Baum:



Dann gilt:

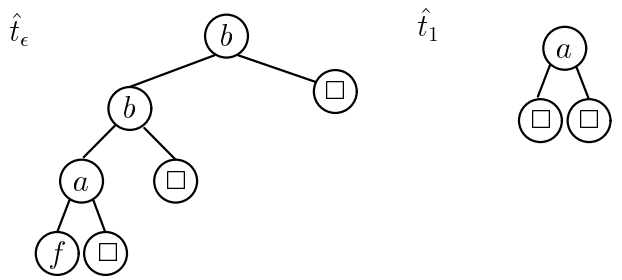
$$\begin{array}{rcl@{\qquad\qquad}rcl} D_\epsilon & = & \{\epsilon, \mathtt{0}, \mathtt{00}\} & D_1 & = & \{\epsilon\} \\ D_\epsilon^+ \setminus D_\epsilon & = & \{\mathtt{1}, \mathtt{01}, \mathtt{000}, \mathtt{001}\} & D_1^+ \setminus D_1 & = & \{\mathtt{0}, \mathtt{1}\}. \end{array}$$

Für $u \in \{\mathtt{0},\mathtt{1}\}^*$ definieren wir nun den endlichen Baum

$$\begin{array}{rrcll} & \widehat{t}_u : & D_u^+ & \to & \Sigma \cup F \\ \text{für alle } w \in D_u \text{ sei} & \widehat{t}_u : & w & \mapsto & t(uw) & (\in \Sigma) \\ \text{für alle } w \in D_u^+ \setminus D_u \text{ sei} & \widehat{t}_u : & w & \mapsto & \ell(uw) & (\in F). \end{array}$$

Offensichtlich ist $\widehat{t}_u \in T_{\Sigma \cup F}$, wobei $f \in F$ die Stelligkeit 0 erhält.

Beispiel:

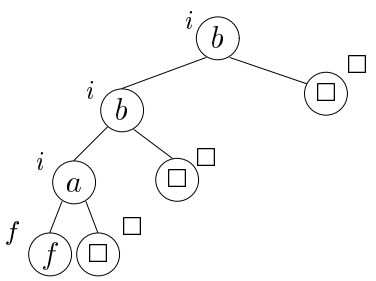


Für jedes $q \in Q$ betrachten wir nun den WB–Baumautomaten

$$\mathcal{A}_q = (Q, \Sigma \cup F, \{q\}, \Delta, \widehat{F}),$$

wobei Δ wie bei $\mathcal{A}$ definiert ist und für alle $f \in F$ gilt $\widehat{F}(f) = f$. Es sei nun $L_q = L(\mathcal{A}_q)$.

Es gilt: Ist $\ell(u) = q$, so ist $\widehat{t}_u \in L_q$.



Die nebenstehende Skizze zeigt z.B. den erfolgreichen Lauf von $\mathcal{A}_i$ auf dem Baum $\widehat{t}_\epsilon$.

Diese Konstruktion zeigt, daß wir zu $L = L_\omega(\mathcal{A})$ eine Baumsprache $\widehat{L}$ definieren können mit $\widehat{L} \supseteq L_\omega(\mathcal{A})$:

$$\widehat{L} = \left(\bigcup_{i \in I} L_i \right) .^{(f_1,\ldots,f_m)} (L_{f_1}, \ldots, L_{f_m})^{\omega,(f_1,\ldots,f_m)}.$$

Umgekehrt kann man leicht zeigen, daß jedes Element von $\widehat{L}$ auch in L ist, d.h. $L = \widehat{L}$.

Satz 7.11 Für eine Sprache $L \subseteq T_\Sigma^\omega$ sind äquivalent:

1. L wird von einem Büchi–Automaten mit Endzustandsmenge $F = \{f_1, \ldots, f_m\}$ erkannt.
2. Es gibt erkennbare Sprachen $L_0, \ldots, L_m \subseteq T_{\Sigma \cup F}$ für das Alphabet $F = \{f_1, \ldots, f_m\}$ von nullstelligen Symbolen, so daß gilt:
$$L = L_0 .^{(f_1,\ldots,f_m)} (L_1, \ldots, L_m)^{\omega,(f_1,\ldots,f_m)}.$$

Beweis:

"$1 \rightsquigarrow 2$" haben wir gerade gezeigt.

"$2 \rightsquigarrow 1$" Konstruiere aus WB–Automaten für $L_0, \ldots, L_m$ einen Büchi–Baumautomaten für L (Übung). ∎

Beweis von Satz 7.10: Es sei $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ ein Büchi–Baumautomat für $L \subseteq T_\Sigma^\omega$. Die Sprachen $L_q = L(\mathcal{A}_q)$ seien wie oben definiert. Wir eliminieren nun sukzessive Zustände, die nichts zu einem erfolgreichen Lauf beitragen können:

1. Der Automat $\mathcal{A}_1 = (Q_1, \Sigma, I_1, \Sigma_1, F_1)$ entsteht aus $\mathcal{A}$ durch Entfernen aller Zustände $q \in Q$ mit $L_q = \emptyset$. Da L_q eine erkennbare Baumsprache (endlicher Bäume) ist, kann man "$L_q = \emptyset$?" entscheiden (Satz 6.22):
$$\begin{array}{lcl} Q_1 & = & Q \setminus \{q \in Q \mid L_q = \emptyset\}, \\ I_1 & = & I \cap Q_1, \\ \Delta_{1a} & : & Q_1 \to 2^{Q_1 \times Q_1} : q \mapsto \Delta_a(q) \cap Q_1 \times Q_1, \\ F_1 & = & F \cap Q_1. \end{array}$$

Ist $L_q = \emptyset$, so kann q nicht in einem erfolgreichen Lauf ℓ vorkommen, denn wir hatten ja gesehen, daß für $\ell(u) = q$ gilt: $\widehat{t}_u \in L_q$. Daher ist $L_\omega(\mathcal{A}_1) = L_\omega(\mathcal{A})$. Beachte, daß es in $\mathcal{A}_1$ einen Zustand q' mit $L(\mathcal{A}_{1q'}) = \emptyset$ und $L(\mathcal{A}_{q'}) \neq \emptyset$ geben kann.

2. Iteriert man diese Vorgehensweise, so erhält man eine Folge $\mathcal{A}_1, \mathcal{A}_2, \ldots$ von Automaten mit $L_\omega(\mathcal{A}_i) = L_\omega(\mathcal{A})$. Da Q endlich ist, muß diese Folge abbrechen, d.h. man erreicht $\mathcal{A}_n$ mit $L_\omega(\mathcal{A}) = L_\omega(\mathcal{A}_n)$ und $L_q \neq \emptyset$ für alle $q \in Q_n$ (beachte: $Q_n = \emptyset$ ist möglich).

3. Wir zeigen: $L_\omega(\mathcal{A}_n) \neq \emptyset$ gdw $I_n \neq \emptyset$.

   Wir wissen aus Satz 7.11:

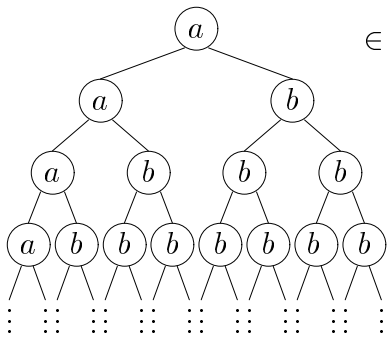
$$L_\omega(\mathcal{A}_n) = \left(\bigcup_{i \in I_n} L_i \right) .^{(f_1,\ldots,f_m)} (L_{f_1}, \ldots, L_{f_m})^{\omega,(f_1,\ldots,f_m)},$$

   wobei $\{f_1, \ldots, f_m\} = F_n$. Wenn $I_n = \emptyset$ ist, so ist der Ausdruck offensichtlich leer. Ist umgekehrt $I_n \neq \emptyset$, so folgt auch $F_n \neq \emptyset$ (da sonst $L_q = \emptyset$ für alle $q \in Q$). Damit enthält also der Ausdruck auf der rechten Seite einen unendlichen Baum, denn für $t_0 \in L_i$ mit $i \in I$, $t_1 \in L_{f_1}, \ldots, t_m \in L_{f_m}$ gilt

$$\{t_0\} .^{(f_1,\ldots,f_m)} (\{t_1\}, \ldots, \{t_m\})^{\omega,(f_1,\ldots,f_m)} \subseteq L_\omega(\mathcal{A}_n).$$

■

Beispiel: Automat aus Beispiel 7.5: Für alle Zustände $q \in Q_1 = \{i, f, \Box\}$ gilt $L_q \neq \emptyset$, denn $a(f, \Box) \in L_i$, $b(\Box, \Box) \in L_\Box$, $a(f, \Box) \in L_f$.



$$\in \{a(f, \Box)\} .^{(f,\Box)} (\{a(f, \Box)\}, \{b(\Box, \Box)\})^{\omega,(f,\Box)}$$

Betrachten wir nun noch Rabin–Baumautomaten.

Satz 7.12 Das Leerheitsproblem ist für Rabin–erkennbare ω–Baumsprachen entscheidbar.

Beweis: Es sei $\mathcal{A} = (Q, \Sigma, I, \Delta, \Omega)$ ein Rabin–Baumautomat. Ein Zustand $q \in Q$ heißt *aktiv*, falls er von einem Zustand aus erreichbar ist und nicht nur sich selbst reproduziert, d.h. wenn es $a, b \in \Sigma$ und Zustände $q_0, q_1, q_2, q' \in Q$ gibt mit

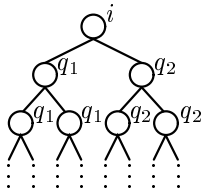
- $(q, q') \in \Delta_b(q_0)$ oder $(q', q) \in \Delta_b(q_0)$,
- $(q_1, q_2) \in \Delta_a(q)$ und
- $\{q_1, q_2\} \neq \{q\}$.

Sonst heißt q *passiv*. Passive Zustände q erlauben also nur Übergänge $\Delta_a(q) = \{(q, q)\}$ oder sie kommen nur an der Wurzel eines Laufs vor. Man kann offenbar sehr leicht feststellen, welche Zustände aktiv und welche passiv sind.

Die Entscheidbarkeit des Leerheitsproblems wird durch Induktion über die Anzahl der aktiven Zustände in $\mathcal{A}$ gezeigt.

Induktionsanfang: keine aktiven Zustände

Ein erfolgreicher Lauf ist also von der nebenstehenden Form. Dabei sind $i, q_1, q_2 \in Q$ Zustände mit



- $i \in I$,
- es gibt $a, b, c \in \Sigma$ mit $(q_1, q_2) \in \Delta_a(i)$, $(q_1, q_1) \in \Delta_b(q_1)$, $(q_2, q_2) \in \Delta_c(q_2)$,
- es gibt $(F, G), (F', G') \in \Omega$ mit $q_1 \in F$ und $q_1 \notin G$, $q_2 \in F'$ und $q_2 \notin G'$.

Man kann offensichtlich leicht feststellen, ob solche i, q_1, q_2 existieren.

Induktionsschritt: $n > 0$ aktive Zustände

Für einen erfolgreichen Lauf $\ell : \{0, 1\}^* \to Q$ gibt es drei Möglichkeiten.

1. Fall: Einer der aktiven Zustände q tritt nicht in ℓ auf.

Dann ist ℓ auch erfolgreicher Lauf für den Automaten $\mathcal{A}_q^-$, der aus $\mathcal{A}$ durch Streichen von q entsteht, d.h. $\mathcal{A}_q^- = (Q', \Sigma, I \cap Q', \Delta', \Omega')$ mit

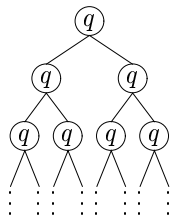
$$\begin{aligned} Q' &= Q \setminus \{q\}, \\ \Delta'_a : Q' &\to 2^{Q' \times Q'} : \\ \Delta'_a(q) &= \Delta_a(q) \cap Q'^2, \\ \Omega' &= \{(F', G') \mid \text{Es gibt } (F, G) \in \Omega \text{ und } F' = F \cap \Omega',\ G' = G \cap \Omega'\}. \end{aligned}$$

Das Auftreten dieses Falles kann also entschieden werden, indem man für jedes aktive $q \in Q$ den Automaten $\mathcal{A}_q^-$ betrachtet. Nach Induktionsvoraussetzung können wir für diese Automaten $\mathcal{A}_q^-$ das Leerheitsproblem entscheiden.

2. Fall: In ℓ gibt es eine Stelle u, so daß $\ell(u) = q$ aktiv ist, und es gibt einen aktiven Zustand q', der in dem Unterbaum ℓ_u (außer an der Wurzel) nicht vorkommt.

Es sei $\widehat{\ell}_q$ der Baum, der aus ℓ entsteht, indem man jeweils nach dem ersten q abschneidet.[2] Offensichtlich ist auch $\widehat{\ell}_q \cdot^q \ell_u$ erfolgreich. Die Existenz eines geeigneten $\widehat{\ell}_q$ und ℓ_u kann nun durch Betrachten von Automaten mit weniger aktiven Zuständen getestet werden:

1. $\widehat{\ell}_q$ wird zu $\tilde{\ell}_q$ modifiziert, indem an den Blättern (q) der nebenstehende unendliche Baum angehängt wird. Einen Automaten, der $\tilde{\ell}_q$ als erfolgreichen Lauf hat, erhält man aus $\mathcal{A}$ wie folgt: $\tilde{\mathcal{A}}_q = (Q, \Sigma, I, \Delta', \Omega')$ mit



   - $\Delta'_a(q) = \{(q, q)\}$ und $\Delta'_a(p) = \Delta_a(p)$ für alle $p \neq q$,
   - $\Omega' = \Omega \cup \{(\{q\}, \emptyset)\}$.

   Offensichtlich hat $\tilde{\mathcal{A}}_q$ einen aktiven Zustand weniger, da q nicht mehr aktiv ist.

[2] Dies ist nach unserer Definition weder ein endlicher noch ein unendlicher Baum, da er sowohl endliche als auch unendliche Pfade enthalten kann. Die nachfolgenden Konstruktionen sind aber trotzdem sinnvoll.

2. ℓ_u ist erfolgreicher Lauf des Automaten $\mathcal{A}^-_{q,q'}$, der aus $\mathcal{A}$ entsteht, indem man q' aus Δ entfernt und q zum einzigen Anfangszustand macht, d.h. $\mathcal{A}^-_{q,q'} = (Q, \Sigma, \{q\}, \Delta', \Omega)$ mit $\Delta'_a(p) = \Delta_a(p) \cap (Q' \times Q')$ für alle $p \in Q$, $a \in \Sigma$, $Q' = Q \setminus \{q'\}$. Auch $\mathcal{A}^-_{q,q'}$ hat einen aktiven Zustand weniger, da q' nicht mehr erreichbar, also nicht mehr aktiv ist. Beachte: Diese Konstruktion ist auch für $q = q'$ korrekt.

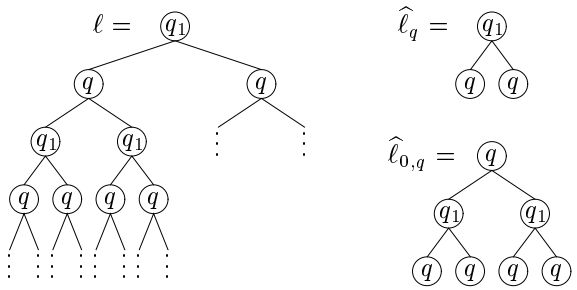
Man kann daher nach Induktion für alle Paare (q, q') aktiver Zustände überprüfen, ob $\tilde{\mathcal{A}}_q$ und $\mathcal{A}^-_{q,q'}$ nicht–leere Sprachen akzeptieren. Ist dies der Fall, so gibt es einen erfolgreichen Lauf, der unter diesen zweiten Fall fällt.

3. Fall: In diesem letzten Fall gibt es mindestens einen aktiven Zustand, und unter jedem u mit $\ell(u) = q$ aktiv kommen *alle* aktiven Zustände vor.

Man kann daher einen Pfad π in ℓ finden, auf dem *jeder* aktive Zustand unendlich oft vorkommt. Außer am Anfang kann offensichtlich kein passiver Zustand auf dem Pfad π liegen. Es muß daher ein Paar $(F, G) \in \Omega$ geben, das π akzeptiert. Damit enthält F einen aktiven Zustand und G enthält nur passive Zustände. Sei $q \in F$ also so ein aktiver Zustand.

Der Baum $\widehat{\ell}_q$ wird nun wie im zweiten Fall definiert. Es sei $u \in \{\mathtt{0}, \mathtt{1}\}^*$ mit $\ell(u) = q$. Der Baum $\widehat{\ell}_{u,q}$ entstehe aus ℓ_u, indem man jeweils nach dem ersten Vorkommen von q in ℓ_u echt unterhalb von ϵ abschneidet.

Beispiel:



Offensichtlich ist $\widehat{\ell}_q \cdot^q \widehat{\ell}_{u,q}{}^{\omega,q}$ ein Lauf des Automaten $\mathcal{A}$. Warum ist er erfolgreich?

Es sei π ein Pfad in diesem Lauf.

Fall a: π ist ein unendlicher Pfad in $\widehat{\ell}_q$ oder ein unendliches Endstück liegt in einem $\widehat{\ell}_{u,q}$. Dann kommt ein unendliches Endstück des Pfades auch in ℓ vor, wird also von einem Paar aus Ω akzeptiert.

Fall b: Sonst kommt q unendlich oft im Pfad π vor. Außerdem kann ein passiver Zustand nur an erster Stelle in π vorkommen. Dann wird π von (F, G) akzeptiert, da $q \in F$ und G nur passive Zustände enthält.

Die Existenz eines $\widehat{\ell}_q$ kann man wie im zweiten Fall mit Hilfe von $\tilde{\mathcal{A}}_q$ testen. Es bleibt zu zeigen, daß man die Existenz eines geeigneten $\widehat{\ell}_{u,q}$ entscheiden kann. Da q in $\widehat{\ell}_{u,q}$ in zwei unterschiedlichen Funktionen vorkommt (an der Wurzel und an den Blättern), müssen wir vor der Konstruktion eines Automaten noch den Zustand an der Wurzel umbenennen: $\widehat{\ell}^{q_0}_{u,q}$ entsteht aus $\widehat{\ell}_{u,q}$, indem die Wurzel mit dem neuen Zustand q_0 markiert wird. Konstruiere wie im Fall 2 aus $\widehat{\ell}^{q_0}_{u,q}$ den Baum $\tilde{\ell}^{q_0}_{u,q}$. Der Automat $\tilde{\mathcal{A}}_{q_0,q} = (Q \cup \{q_0\}, \Sigma, \{q_0\}, \Delta', \Omega')$ mit

- $q_0 \notin Q$,
- $\begin{aligned} \Delta'_a(q_0) &= \Delta_a(q), \\ \Delta'_a(q) &= \{(q, q)\}, \\ \Delta'_a(q') &= \Delta_a(q') \text{ für } q' \notin \{q, q_0\}, \end{aligned}$
- $\Omega' = \Omega \cup \{(\{q\}, \emptyset)\}$,

hat $\tilde{\ell}^{q_0}_{u,q}$ als erfolgreichen Lauf. $\tilde{\mathcal{A}}_{q_0,q}$ enthält einen aktiven Zustand weniger, da q und q_0 passiv sind.

Zusammenfassung
Für einen Rabin–Baumautomaten $\mathcal{A}$ ist also $L_\omega(\mathcal{A}) = \emptyset$ genau dann, wenn das folgende gilt:

- Falls $\mathcal{A}$ keine aktiven Zustände enthält, gibt es keine Zustände $i, q_1, q_2 \in Q$ mit
  - $i \in I$ und
  - es gibt $a, b, c \in \Sigma$ mit $(q_1, q_2) \in \Delta_a(i)$, $(q_1, q_1) \in \Delta_b(q_1)$, $(q_2, q_2) \in \Delta_c(q_2)$ und

  - es gibt $(F, G), (F', G') \in \Omega$ mit $q_1 \in F$ und $q_1 \notin G$, $q_2 \in F'$ und $q_2 \notin G'$;

- falls $\mathcal{A}$ mindestens einen aktiven Zustand enthält, gilt für alle aktiven Zustände q
  - $L(\mathcal{A}_q^-) = \emptyset$ und
  - $L(\tilde{\mathcal{A}}_q) = \emptyset$ oder für alle aktiven Zustände q' gilt $L(\mathcal{A}_{q,q'}^-) = \emptyset$ und
  - falls es ein Paar $(F, G) \in \Omega$ gibt mit $q \in F$ und G enthält nur passive Zustände, gilt $L(\tilde{\mathcal{A}}_q) = \emptyset$ oder $L(\tilde{\mathcal{A}}_{q_0,q}) = \emptyset$. ∎

Kapitel 8

Baumautomaten und logische Formeln

Wir werden nun Entscheidbarkeitsergebnisse für Baumsprachen auf entsprechende Erweiterungen der bisher behandelten Logiken S1S, WS1S und 1DPDL übertragen.

8.1 Die Logik S2S und ihre Varianten

Wir betrachten zunächst eine Logik, welche S1S dahingehend erweitert, daß zwei Nachfolgerfunktionen zur Verfügung stehen. Anstelle des Interpretationsbereichs $\mathbb{N}$ verwendet man dafür den (unendlichen) binären Baum.

Definition 8.1

1. Formeln der Logik S2S sind wie Formeln von S1S aufgebaut. Der einzige Unterschied ist, daß man statt einer Nachfolgerfunktion $\boldsymbol{s}$ zwei verschiedene Nachfolgerfunktionen $\boldsymbol{s}_0$ und $\boldsymbol{s}_1$ verwenden darf und daß man statt $\mathbf{0}$ die Konstante $\boldsymbol{\epsilon}$ verwendet.

2. Als Interpretationsbereich betrachtet man die Menge $\{0,1\}^*$ (Positionen im unendlichen, binären Baum), wobei

- $\boldsymbol{\epsilon}$ als ϵ interpretiert wird,
- $\boldsymbol{s}_0 : u \mapsto u\mathtt{0}$, $\boldsymbol{s}_1 : u \mapsto u\mathtt{1}$,
- $\dot{<}$ als Präfixordnung auf $\{\mathtt{0},\mathtt{1}\}^*$, d.h. $u \dot{<} v$ gdw es gibt $w \in \{\mathtt{0},\mathtt{1}\}^+$ mit $uw = v$,
- $P_1, \ldots, P_n$ als Teilmengen von $\{\mathtt{0},\mathtt{1}\}^*$.

3. WS2S entsteht aus S2S dadurch, daß Quantoren zweiter Stufe nur über endliche Teilmengen von $\{\mathtt{0},\mathtt{1}\}^*$ quantifizieren dürfen.

S2S–Formeln können dazu verwendet werden, ω–Baumsprachen zu definieren. Wie bei S1S verwenden wir als Alphabet $\Sigma = \{0,1\}^n$ (wenn n die Anzahl der vorhandenen einstelligen Prädikatssymbole ist) und benutzen $Q_a(x)$ als Abkürzung, wenn an Position x das Symbol $a \in \{0,1\}^n$ stehen soll. Jedes Element dieses Alphabets erhält Stelligkeit zwei. Eine S2S–Interpretation I kann nun als ein ω–Baum t_I mit Beschriftung aus Σ aufgefaßt werden:

$$\begin{array}{rcl} t_I : \{\mathtt{0},\mathtt{1}\}^* & \to & \Sigma \\ u & \mapsto & (b_1, \ldots, b_n) \text{ mit } b_i = \left\{ \begin{array}{ll} 1 & u \in {P_i}^I \\ 0 & u \notin {P_i}^I. \end{array} \right. \end{array}$$

Definition 8.2 Es sei ϕ eine geschlossene S2S–Formel. Die von ϕ charakterisierte ω–Baumsprache ist definiert durch

$$L_\omega(\phi) := \{t_I \in T_\Sigma^\omega \mid I \models \phi\}.$$

Beispiele für S2S–Formeln sind:

- Kette(X) beschreibt diejenigen Teilmengen X von $\{\mathtt{0},\mathtt{1}\}^*$, in denen je zwei Elemente präfixvergleichbar sind.

$$\text{Kette}(X) := \forall x. \forall y. X(x) \wedge X(y) \Rightarrow x \dot{<} y \vee x \doteq y \vee y \dot{<} x$$

- UnendlicheKette(X) beschreibt unendlich große Teilmengen X, für die auch Kette(X) gilt.

$$\text{UnendlicheKette}(X) := \text{Kette}(X) \wedge \forall x. X(x) \Rightarrow \exists y. x \dot{<} y \wedge X(y)$$

- Pfade sind maximale Ketten, d.h. sie sind unendlich und ohne Zwischenräume. Dabei ist $X \dot{\subseteq} Y := \forall x. X(x) \Rightarrow Y(x)$ und $X \doteq Y := \forall x. X(x) \Leftrightarrow Y(x)$.

$$\text{Pfad}(X) := \text{Kette}(X) \wedge \forall Y. \text{Kette}(Y) \wedge X \dot{\subseteq} Y \Rightarrow X \doteq Y$$

- Um endliche Teilmengen zu beschreiben, verwenden wir in Endlich(X) das Prädikat Z, das für den Präfixabschluß von X steht (für x mit $X(x)$ enthält Z alle Präfixe von x). Mit Hilfe des Lemmas von König kann man zeigen: Ist X endliche Teilmenge von $\{0,1\}^*$, so auch der Präfixabschluß Z von X, d.h. Z enthält keine unendliche Kette.

$$\begin{aligned}\text{Endlich}(X) := \neg\exists Y. \exists Z. (&Y \dot{\subseteq} Z \wedge \text{UnendlicheKette}(Y) \wedge \\ &\forall z. Z(z) \Leftrightarrow \exists x. X(x) \wedge (z \dot{<} x \vee z = x))\end{aligned}$$

Die letzte Formel zeigt, daß man wieder WS2S in S2S ausdrücken kann. Im Gegensatz zum Fall mit einem Nachfolger gilt die Umkehrung aber nicht (siehe Satz 8.6).

Beispiel 8.3 Sei $n = 1$, d.h. $\Sigma = \{0, 1\}$.

1. $L_1 = \{t \in T_\Sigma^\omega \mid$ es gibt einen Pfad in t, der unendlich viele 1 enthält$\}$ (vergleiche Beispiel 7.5) wird von ϕ_1 beschrieben, d.h. $L_1 = L_\omega(\phi_1)$ mit

$$\phi_1 = \exists Y. \text{Pfad}(Y) \wedge \forall x. Y(x) \Rightarrow \exists y. Y(y) \wedge x \dot{<} y \wedge P_1(y).$$

2. $L_2 = \overline{L_1} = \{t \in T_\Sigma^\omega \mid$ Jeder Pfad in t enthält nur endlich viele 1$\}$ wird natürlich von $\neg\phi_1$ beschrieben, d.h. $L_2 = L_\omega(\neg\phi_1)$.

Satz 8.4 [Rabin] Für eine ω–Baumsprache $L \subseteq T_\Sigma^\omega$ sind äquivalent:

1. L ist Rabin–erkennbar.
2. $L = L_\omega(\phi)$ für eine geschlossene S2S–Formel ϕ.

Beweis: Der Beweis ist sehr ähnlich zum Beweis von Satz 5.4.

"$1 \rightsquigarrow 2$" Sei $\mathcal{A} = (Q, \Sigma, I, \Delta, \Omega)$ ein Rabin–Baumautomat mit $Q = \{q_1, \ldots, q_m\}$. Für jedes q_i führen wir eine Variable Y_i ein; dabei soll $Y_i(x)$ gelten, wenn q_i an Position x eines Laufs steht. Die Existenz eines erfolgreichen Laufs wird beschrieben durch:

$$
\begin{aligned}
&\exists Y_1 \ldots \exists Y_m.\\
&\quad \Big(\bigvee_{q_i \in I} Y_i(\boldsymbol{\epsilon}) \Big) \wedge \\
&\quad \Big(\forall x. \bigwedge_{\substack{q_i, q_j \in Q \\ 1 \le i < j \le m}} \neg (Y_i(x) \wedge Y_j(x)) \Big) \wedge \\
&\quad \Big(\forall x. \bigvee_{(q_i, q_j) \in \Delta_a(q_k)} Y_k(x) \wedge Q_a(x) \wedge Y_i(\boldsymbol{s}_0(x)) \wedge Y_j(\boldsymbol{s}_1(x)) \Big) \wedge \\
&\quad \Big(\forall Z. \mathrm{Pfad}(Z) \Rightarrow \\
&\qquad \bigvee_{(F,G) \in \Omega} \Big(\bigvee_{q_i \in F} \forall x. Z(x) \Rightarrow \exists y. Z(y) \wedge x \dot{<} y \wedge Y_i(y) \Big) \wedge \\
&\qquad\quad \Big(\bigwedge_{q_i \in G} \exists x. \forall y. Z(y) \wedge x \dot{<} y \Rightarrow \neg Y_i(y) \Big) \Big) \Big)
\end{aligned}
$$

"$2 \rightsquigarrow 1$" Wie bei S1S reduziert man S2S–Formeln auf $\mathrm{S2S}_0$–Formeln und verwendet dann Induktion über den Aufbau von $\mathrm{S2S}_0$–Formeln. Wichtig ist hier: Abschluß unter Booleschen Operationen und Alphabetumbenennung von Rabin–erkennbaren Sprachen.

Das folgende Korollar ist direkte Konsequenz des Satzes 8.4.

Korollar 8.5 Gültigkeit in S2S ist entscheidbar.

Es gibt einen interessanten Zusammenhang zwischen Büchi–erkennbaren ω–Baumsprachen und WS2S.

Satz 8.6 (Rabin)

1. $L \subseteq T_\Sigma^\omega$ ist Büchi–erkennbar genau dann, wenn $L = L_\omega(\phi)$ für eine geschlossene S2S–Formel der Gestalt $\exists Y_1. \ldots \exists Y_m. \psi(Y_1, \ldots, Y_m)$ gilt, wobei ψ eine WS2S–Formel ist (d.h. nur die äußeren existentiellen Quantoren dürfen über unendliche Mengen quantifizieren).

2. $L \subseteq T_\Sigma^\omega$ kann durch eine WS2S–Formel charakterisiert werden genau dann, wenn L und $\overline{L}$ Büchi–erkennbar sind.

Insbesondere sind also WS2S–Formeln ausdrucksschwächer als Büchi–Baumautomaten, die wiederum schwächer als Rabin–Baumautomaten sind. Zum Beispiel kann L_1 aus 8.3 nicht durch eine WS2S–Formel definiert werden, da $\overline{L}_1$ nicht Büchi–erkennbar ist (natürlich ist $\overline{L}_1$ Rabin–erkennbar). Wir haben im Zusammenhang mit ω–Sprachen gesehen, daß S1S und WS1S die gleiche Ausdrucksstärke haben, während S2S also echt ausdrucksstärker als WS2S ist. Der Beweis dieses Satzes wird hier nicht geführt, Literaturhinweise finden sich in [Tho90a].

8.2 Dynamische Logiken

Anstelle binärer Bäume kann man auch Bäume mit Verzweigungsgrad $k \geq 1$ betrachten. Es gelten die entsprechenden Resultate wie für den Fall $k = 2$. Insbesondere ist also SkS (k ist die Anzahl der Nachfolgerfunktionen) entscheidbar. Man kann Büchi–Automaten für Bäume vom Verzweigungsgrad k dazu verwenden, Entscheidbarkeit der Logik DPDL zu zeigen (einer Verallgemeinerung von 1DPDL, mit k atomaren Programmen $p_0, \ldots, p_{k-1}$).

Definition 8.7 Die Formeln der Logik DPDL sind wie die bereits definierten von 1DPDL, mit dem einzigen Unterschied, daß man endlich viele atomare Programme $p_0, \ldots, p_{k-1}$ zum Aufbau von Programmformeln zuläßt. Eine Interpretation weist jedem Programm p_i eine funktionale Relation[1] $p_i^I \subseteq \Delta^I \times \Delta^I$ zu.

[1] Das erste "D" von DPDL steht für "deterministisch".

Wie im Fall $k = 1$ kann man Programmformeln als reguläre Sprachen (nun über dem Alphabet $\Sigma = \{p_0, \ldots, p_{k-1}\}$) beschreiben und damit wieder Programmformeln durch Zustände eines geeigneten Automaten $\mathcal{B}$ ersetzen, wir schreiben dann $\widehat{\phi}$ statt ϕ. Mit Hilfe dieses Automaten $\mathcal{B} = (Q, \{p_0, \ldots, p_{k-1}\}, \cdot, \Delta, F)$ kann man dann den *Fischer–Ladner–Abschluß* einer DPDL–Formel ϕ_0 in Negationsnormalform (NNF) entsprechend zum Fall $k = 1$ definieren, z.B.:

$$\begin{aligned}\langle q\rangle\phi \in FL(\phi_0) \rightsquigarrow\ & \phi \in FL(\phi_0) \text{ und} \\ & \langle q'\rangle\phi \in FL(\phi_0) \text{ für alle } q' \in Q \text{ mit } (q, p_i, q') \in \Delta \\ & \text{für ein } i \text{ mit } 0 \leq i < k.\end{aligned}$$

Die Rolle des Hintikka–Wortes im Fall $k = 1$ übernimmt für $k > 1$ ein Hintikka–*Baum*. Es sei $\Sigma = 2^{FL(\phi_0)}$.

Definition 8.8 Ein Baum $t : \{\mathtt{0}, \mathtt{1}, \ldots, k-\mathtt{1}\}^* \to \Sigma$ ist ein *Hintikka–Baum* für ϕ_0, falls für alle $u \in \{\mathtt{0}, \mathtt{1}, \ldots, k-\mathtt{1}\}^*$ und für alle $\phi, \phi \wedge \psi, \phi \vee \psi, [q]\phi, \langle q\rangle\phi \in FL(\phi_0)$ gilt:

1. $\widehat{\phi}_0 \in t(\epsilon)$,
2. $t(u) = \emptyset$ oder $A_i \in t(u)$ gdw $\neg A_i \notin t(u)$ für alle atomaren Eigenschaften A_i,
3. $\phi \wedge \psi \in t(u)$ gdw $\phi \in t(u)$ und $\psi \in t(u)$,
4. $\phi \vee \psi \in t(u)$ gdw $\phi \in t(u)$ oder $\psi \in t(u)$,
5. $[q]\phi \in t(u) \rightsquigarrow$
   (a) $\epsilon \in L(\mathcal{B}_q) \rightsquigarrow \phi \in t(u)$,
   (b) für alle i mit $0 \leq i < k$ gilt $t(ui) = \emptyset$ oder für alle q' mit $(q, p_i, q') \in \Delta$ gilt $[q']\phi \in t(ui)$,
6. $\langle q\rangle\phi \in t(u) \rightsquigarrow$ es gibt $w = p_{i_1} \cdots p_{i_r} \in L(\mathcal{B}_q)$ und Zustände $q_1, \ldots, q_r$ von $\mathcal{B}$, mit $q_r \in F$ und $q \xrightarrow{p_{i_1}}_{\mathcal{B}} q_1 \xrightarrow{p_{i_2}}_{\mathcal{B}} \cdots \xrightarrow{p_{i_r}}_{\mathcal{B}} q_r$ und $\langle q_j\rangle\phi \in t(ui_1 \cdots i_j)$ für $1 \leq j < r$ und $\phi \in t(ui_1 \cdots i_r)$,

7. $t(u) = \emptyset \rightsquigarrow t(ui) = \emptyset$ für alle i mit $0 \leq i < k$.

Wie im Fall von 1DPDL entspricht ein Hintikka–Baum einem Modell für DPDL–Formeln.

Satz 8.9 Eine DPDL–Formel ϕ_0 ist genau dann erfüllbar, wenn es einen Hintikka–Baum für ϕ_0 gibt.

Vorbereitung des Beweises: Man betrachtet zunächst wieder einen lokalen Büchi–Baumautomaten $\mathcal{A}_L(\phi_0)$, in dem statt der globalen Bedingung 6. die lokale Bedingung 6.' überprüft wird:

6'. $\langle q \rangle \phi \in t(u) \rightsquigarrow$
  6'.(a) $\epsilon \in L(\mathcal{B}_q)$ und $\phi \in t(u)$ oder
  6'.(b) es gibt ein i mit $0 \leq i < k$ und ein q' mit $(q, p_i, q') \in \Delta$ und $\langle q' \rangle \phi \in t(ui)$.

Definition 8.10 Der *lokale Büchi–Baumautomat* $\mathcal{A}_L(\phi_0) = (Q_L, \Sigma, I_L, \Delta_L, F_L)$ ist definiert durch

$$\begin{array}{rcl} Q_L & := & \{\alpha \in 2^{FL(\phi_0)} \mid \alpha \text{ erfüllt 2, 3, 4 von Definition 8.8}\}, \\ \Sigma & := & 2^{FL(\phi_0)}, \\ I_L & := & \{\alpha \in Q_L \mid \widehat{\phi}_0 \in \alpha\} \text{ (Damit ist 1 von Definition 8.8 erfüllt)}, \\ F_L & := & Q_L, \end{array}$$

für alle $\sigma \in \Sigma$ und $t(u) \in Q_L$

$$\begin{array}{rcl} t(u) \neq \sigma & \rightsquigarrow & \Delta_{L\sigma}(t(u)) := \emptyset \\ t(u) = \sigma & \rightsquigarrow & \Delta_{L\sigma}(t(u)) := \{(t(u\mathtt{0}), \ldots, t(u(k-\mathtt{1}))) \mid \text{ 5, 6', 7 von} \\ & & \text{Definition 8.8 sind erfüllt}\}. \end{array}$$

Nun haben wir also *beinahe* einen Büchi–Baumautomaten für eine DPDL–Formel ϕ_0 mit der Eigenschaft, daß die von ihm akzeptierte Sprache nicht leer ist, falls ϕ_0 erfüllbar ist — wir müssen nur noch erzwingen, daß die $\langle\rangle$–Teilformeln tatsächlich irgendwann erfüllt werden (und ihre Erfüllung nicht unendlich oft aufgeschoben wird). Deshalb erweitert man $\mathcal{A}_L(\phi_0)$ zu einem Hintikka–Baumautomaten, ganz ähnlich wie bei der Konstruktion für ω–Sprachen.

Definition 8.11 Der *Hintikka–Baumautomat* $\mathcal{A}_H(\phi_0) = (Q_H, \Sigma, I_H, \Delta_H, F_H)$ ist definiert durch:

$$\begin{array}{rcl} Q_H & := & Q_L \times 2^{FL_{\langle\rangle}(\phi_0)} \text{ mit} \\ & & FL_{\langle\rangle}(\phi_0) = \{\langle q\rangle\psi \mid \langle q\rangle\psi \in FL(\phi_0)\}, \\ \Sigma & := & 2^{FL(\phi_0)}, \\ I_H & := & I_L \times \{\emptyset\}, \\ F_H & := & F_L \times \{\emptyset\}. \end{array}$$

$\Delta_{H\sigma}((\alpha, s))$ ist folgendermaßen definiert:

1. Fall: $(s = \emptyset)$

$((\beta_0, s_0), \ldots, (\beta_{k-1}, s_{k-1})) \in \Delta_{H\sigma}((\alpha, \emptyset))$ gdw

- $(\beta_0, \ldots, \beta_{k-1}) \in \Delta_{L\sigma}(\alpha)$ und
- für jedes $\langle q\rangle\psi \in \sigma$ mit $(\epsilon \notin L(\mathcal{B}_q)$ oder $\psi \notin \alpha)$ gibt es ein i mit $0 \leq i < k$ und q' mit $(q, p_i, q') \in \Delta$ und $\langle q'\rangle\psi \in s_i \cap \beta_i$.

2. Fall: $(s \neq \emptyset)$

Wie der erste Fall, wobei "$\langle q\rangle\psi \in \sigma$" durch "$\langle q\rangle\psi \in s$" zu ersetzen ist.

Satz 8.12 Der Hintikka–Automat $\mathcal{A}_H(\phi_0)$ akzeptiert genau die Hintikka–Bäume für ϕ_0.

Der Beweis entfällt, da er sehr technisch ist und außerdem die Konstruktion der beiden Baumautomaten die Aussage dieses Satzes *beinahe* sofort liefert. Als Konsequenz ergibt sich dann folgende Tatsache:

Satz 8.13 Erfüllbarkeit in DPDL ist entscheidbar.

Beachte: Man kann zeigen, daß diese Entscheidung in exponentieller Zeit möglich ist, da das Leerheitsproblem für Büchi–Baumautomaten polynomiell entscheidbar ist. Dies ergibt sich allerdings nicht direkt aus dem vorne beschriebenen Verfahren.

Statt DPDL kann man auch PDL betrachten, hier können nun auch atomare Programme nicht–deterministisch sein. Die Formeln in PDL sind genauso definiert wie in DPDL, allerdings ändert sich die Semantik: atomare Programme können nun auch durch *nicht–funktionale* Relationen $p_i^I \subseteq \Delta_i^I \times \Delta_i^I$ interpretiert werden.

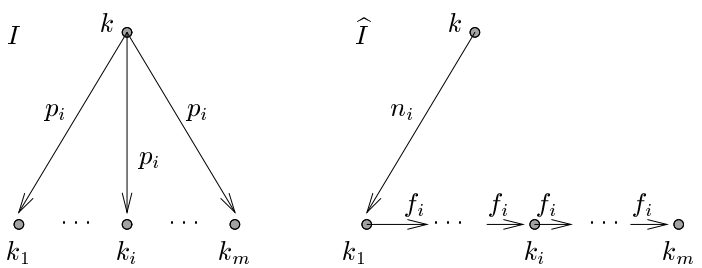
Satz 8.14 Erfüllbarkeit in PDL ist entscheidbar.

Beweis: Es seien $p_0, \ldots, p_{k-1}$ die in der PDL–Formel ϕ vorkommenden atomaren Programme. Für jedes i führen wir zwei neue Programmsymbole f_i, n_i für atomare deterministische Programme ein. Die DPDL–Formel $\widehat{\phi}$ entsteht aus ϕ, indem man jedes Vorkommen von p_i in ϕ durch $n_i; f_i^*$ ersetzt. Wir zeigen:

ϕ hat ein Modell genau dann, wenn $\widehat{\phi}$ ein Modell hat.

"$\rightsquigarrow$" Zu jedem Modell I von ϕ gibt es ein Modell $\widehat{I}$ von $\widehat{\phi}$:

1. Es gilt: Gibt es ein Modell, so gibt es auch ein Baummodell. In Baummodellen bilden die Relationen p_i^I einen Baum und keinen allgemeinen gerichteten Graphen, d.h.
   - es gibt eine Wurzel k_0, für die für alle $k \in \Delta^I$ und alle i gilt: $(k, k_0) \notin p_i^I$ und
   - für alle $k \in \Delta^I \setminus \{k_0\}$ gilt: es gibt genau ein $k' \in \Delta^I$ und genau ein i mit $(k', k) \in p_i^I$.
2. Für ein Baummodell I' von ϕ definiert man das zugehörige Modell $\widehat{I}$ von $\widehat{\phi}$ wie folgt:
   Es sei $\{k_1, \ldots, k_m\} = \{k' \mid (k, k') \in p_i^I\}$. Dann setze $(k, k_1) \in n_i^{\widehat{I}}$ und $(k_i, k_{i+1}) \in f_i^{\widehat{I}}$.



"$\rightsquigarrow$" Ist $\widehat{I}$ Modell von $\widehat{\phi}$, so definiert man in I: $p_i^I = n_i^{\widehat{I}} \circ (f_i^{\widehat{I}})^*$.

Es ist also ϕ erfüllbar genau dann, wenn $\widehat{\phi}$ erfüllbar ist. ∎

Schlußwort

Wir haben nun verschiedene Klassen formaler Sprachen, ω–Sprachen, Baumsprachen und ω–Baumsprachen kennengelernt, sowie unterschiedliche Charakterisierungen dieser Klassen durch Ausdrücke, Automaten, Monoide, Halbgruppen und logische Formeln. Diese unterschiedlichen Charakterisierungen erlauben uns, je nach Situation, unterschiedliche Techniken anzuwenden um z.B. zu zeigen, daß eine Sprache nicht in einer Klasse liegt, daß eine Abschlußeigenschaft für eine Klasse gilt oder daß ein Problem für eine Klasse entscheidbar ist.

Insbesondere konnten wir durch die Charakterisierung durch logische Formeln Entscheidbarkeitsresultate aus der Automatentheorie auf Logiken übertragen. Außerdem haben wir dynamische Logiken betrachtet und konnten zeigen, daß diese entscheidbar sind, indem wir für Formeln dieser Logiken entsprechende Automaten konstruiert haben. Mindestens ebenso wichtig wie die Resultate war das Erlernen der verwendeten Methoden (wie Ehrenfeucht-Fraïssée-Spiele, Fischer-Ladner-Abschluß, Ramsey-Theorem, etc.)

Danksagungen

Wir danken Stefan Neskakis und Felix Wolf für das aufmerksame Korrekturlesen und die vielen Hinweise, die nun zu einem besseren Verständnis beitragen.

Literaturverzeichnis

[Eil76] Samuel Eilenberg: *Automata, Languages, and Machines*, Band A & B. Academic Press, New York, 1976.

[GS84] Ferenc Gécseg und Magnus Steinby: *Tree Automata.* Akadémia Kiadó, Budapest, 1984.

[Lad77] Richard E. Ladner: Application of Model Theoretic Games to Discrete Linear Orders and Finite Automata. *Information and Control*, 33:281–303, 1977.

[Ros82] Joseph G. Rosenstein: *Linear Orderings.* Academic Press, New York, 1982.

[Sch92] Uwe Schöning: *Logik für Informatiker.* Reihe Informatik, Band 56. BI Wissenschaftsverlag, Mannheim, 3. Auflage, 1992.

[Tho79] Wolfgang Thomas: Star-Free Regular Sets of ω-Sequences. *Information and Control*, 42:148–156, 1979.

[Tho81] Wolfgang Thomas: A Combinatorial Approach to the Theory of ω-Automata. *Information and Control*, 48:261–283, 1981.

[Tho82] Wolfgang Thomas: Classifying Regular Events in Symbolic Logic. *J. of Computer and System Sciences*, 25:360–376, 1982.

[Tho90a] Wolfgang Thomas: Automata on Infinite Objects. In: J. van Leeuwen (Herausgeber): *Handbook of Theoretical Computer Science*, Kapitel 4. Elsevier Science Publisher, 1990.

[Tho90b] Wolfgang Thomas: Grundzüge der Theoretischen Informatik. Skriptum zur Vorlesung im Wintersemester 1989/90. Fachgruppe Informatik, RWTH Aachen, 1990.

Notationstabelle

	Formale Sprachen
Σ	Endliches Alphabet
Σ^*	Die Menge aller endlichen Wörter über Σ
Σ^+	Die Menge aller nicht–leeren endlichen Wörter über Σ
Σ^ω	Die Menge aller unendlichen Wörter über Σ
T_Σ	Die Menge aller endlichen Bäume über Σ
T_Σ^ω	Die Menge aller unendlichen Bäume über Σ
$L, L_1, \ldots$	Formale Sprachen
$L_{p,q}$	Menge der Wörter, die Pfadbeschriftung von p nach q sind
$L_{p,q}^F$	Menge der Wörter, die Pfadbeschriftung von p nach q über einen Zustand aus F sind
L^ω	Menge der unendlichen Wörter, für die es eine Zerlegung in Faktoren aus L gibt
$\lim L$	Menge aller unendlichen Wörter mit unendlich vielen Präfixen in L
$L_{k,n}$	Formeln Logik erster Stufe mit k freien Variablen, Quantorentiefe n
$\alpha(m,n)$	Endliches Segment eines ω–Wortes α
$\alpha(m,\omega)$	Unendliches Segment eines ω–Wortes α
$\sim_L$	Syntaktische Kongruenz bezüglich L

$[x]_\sim$	Die Äquivalenzklasse von x bezüglich $\sim$
$\cdot$	Konkatenation von Wörtern, oft weggelassen
$\nu(a)$	Stelligkeit des Symbols $a \in \Sigma$ (Σ Baumalphabet)
t_u	Der Unterbaum des Baumes t, der an Position u hängt
$\textcircled{x}$	Der Baum, der nur aus der Wurzel x besteht
$\cdot^{\bar{x}}$	Konkatenation von Bäumen bzgl. den Blättern $\bar{x}$
$\cdot^{n,\bar{x}}$	n–fache Iteration von Bäumen bzgl. den Blättern $\bar{x}$
$\cdot^{\omega,\bar{x}}$	Unendliche Iteration von Bäumen bzgl. den Blättern $\bar{x}$
	Automaten
$\mathcal{A}, \mathcal{B}$	Automaten
$\mathcal{A}_M$	Minimaler Automat
$\mathcal{B}_q$	Automat mit Anfangszustand q
$\mathcal{A}_L$, $\mathcal{A}_H$	Lokaler Automat, Hintikka–Automat
Q	Zustandsmenge eines Automaten
F	Endzustandsmenge eines Automaten, Endzuweisung eines WB–Baumautomaten
I	Anfangszustandsmenge eines Automaten, Anfangszuweisung eines BW–Baumautomaten
Ω	Menge von Tupeln (F, G) von Endzustandsmengen (in Rabin–Automaten)
Δ	$\subseteq Q \times \Sigma \times Q$, Übergangsrelation eines Automaten
δ	$: Q \times \Sigma \to Q$ Übergangsfunktion eines Automaten
Δ	Übergangszuweisung eines Baumautomaten, die für jedes $a \in \Sigma$ ein Δ_a enthält
Δ_a	Übergangsfunktion eines BW–Baumautomaten: $Q^n \to 2^Q$, eines WB–Baumautomaten: $Q \to 2^{Q^n}$ oder eines Rabin– oder Büchi–Baumautomaten: $Q \to 2^{Q^2}$
$p \xrightarrow{w}_{\mathcal{A}} q$	In $\mathcal{A}$ gibt es einen mit w beschrifteten Pfad von p nach q

$p \xrightarrow{w,F}_{\mathcal{A}} q$	Es gilt $p \xrightarrow[\mathcal{A}]{w} q$, und dieser Pfad geht über einen Zustand in F
$L(\mathcal{A})$	Die von $\mathcal{A}$ akzeptierte Menge endlicher Wörter oder Bäume
$L_\omega(\mathcal{A})$	Die von $\mathcal{A}$ akzeptierte Menge unendlicher Wörter oder Bäume
$M_\mathcal{A}$	Übergangsmonoid des Automaten $\mathcal{A}$
M_L	Syntaktisches Monoid von L, d.h. Übergangsmonoid des minimalen Automaten von L
S_L	Syntaktische Halbgruppe von L
$\Sigma^*/_{\sim_L}$	Quotientenmonoid bezüglich der syntaktischen Kongruenz
	Klassen
B_0	Klasse der verallgemeinert–definiten Sprachen
SF	Klasse der sternfreien Sprachen
$\mathcal{R}eg(T_\Sigma, Z)$	Klasse der regulären, endlichen Bäume bzgl. Σ und Z
$\widehat{\mathbb{D}}$	Klasse der endlichen Halbgruppen S mit: e idempotent $\rightsquigarrow eSe = e$
Ap	Klasse der aperiodischen Monoide
	Logik
$\mathbf{0}$	Logische Konstante, zu interpretieren als 0
$\boldsymbol{s}, \boldsymbol{s}_0, \boldsymbol{s}_1$	Logische Funktionskonstanten, zu interpretieren als Nachfolgerfunktionen
$\dot{<}$	Binäre Relation, zu interpretieren als übliches "$<$" auf ω oder Präfixordnung auf Wörtern
$\doteq$	Gleichheit in logischen Formeln
$\dot{\subseteq}$	Teilmengenbeziehung zwischen Prädikaten bzw. Variablen zweiter Stufe in logischen Formeln
$\dot{\equiv}$	Gleichheit von Prädikaten bzw. Variablen zweiter Stufe in logischen Formeln
$Q_a(x)$	Abkürzung für "an der Stelle x steht das Symbol $a \in \Sigma$"
$\pi, \pi_1, \ldots$	Programmformeln

$\pi_1;\pi_2$	Hintereinanderausführen von Programmformeln
$\pi_1 \cup \pi_2$	Alternative Ausführung von Programmformeln
π_1^*	Iteration von Programmformeln
$[\pi]\phi$	Konfigurationsformel, "in allen π–Folgekonfigurationen gilt ϕ"
$\langle\pi\rangle\phi$	Konfigurationsformel, "es gibt eine π–Folgekonfiguration, in der ϕ gilt"
$\Delta^I, \cdot^I$	Interpretationsbereich und Interpretationsfunktion
$\widehat{\phi}$	Entsteht aus ϕ Konfigurationsformel, indem Programmformeln in [] und $\langle\rangle$ durch Zustände des zugehörigen Automaten ersetzt werden
$FL(\phi)$	Der Fischer–Ladner Abschluß von ϕ
	Allgemeines
$\circ$	Komposition von Funktionen oder Relationen
$\times$	Kartesisches Produkt
$\bullet_M, \bullet$	Monoidoperationen
$1_M, 1$	Einselemente von Monoiden
ϕ, ψ	Homomorphismen
$\overline{n}$	Abkürzung für das kleinste gemeinesame Vielfache $kgV(1,\ldots,n)$
$\dot\cup$	Disjunkte Vereinigung
$\simeq$	Isomorphie
$\equiv_{k,n}$	Metalogische Äquivalenz logischer Formeln in $L_{k,n}$
$\sim_{k,n}$	Äquivalenzrelation zu mit k Zügen angefangenen Ehrenfeucht–Fraïssé Spielen der Länge n
$\prec$	Präfixrelation auf ω^*
$[M]^2$	Die Menge aller zweielementigen Teilmengen von M
ω	Der Ordnungstyp der natürlichen Zahlen

Index

$\approx_L$, 98
$\prec$, 114
$\sim_{\mathcal{A}}$, 74
$[M]^2$, 77
1DPDL, 100
ω–Baumsprache, 132
    Büchi–erkennbar, 135
    Rabin–erkennbar, 135
ω–Sprache
    sternfrei, 97
ω–Sprache, 62
    ω–reguläre, 69
    Büchi–erkennbar, 65

äquivalent, 102
akzeptierte ω–Sprache
    von einem Automat, 65
akzeptierte ω–Sprache
    von einer Formel, 84
akzeptierte Sprache
    von Automaten, 8
    von Monoiden, 11
Alphabet mit Stelligkeitsfunktion, 114
Alphabetumbenennung, 129
Ausdruck
    ω–regulär, 69
    regulär, 7
Automat, *siehe auch* Baumautomat
    Büchi-, 64
    deterministisch, 9
    endlich, 7
    Hintikka-, 109
    lokal, 108
    minimal, 10
    Muller-, 80

Büchi–Automat, 64
Baum
    Σ-, 114
    endlich, 114
    unendlich, 132
Baumautomat
    Büchi-, 134
    BW–Baumautomat, 116
    deterministisch, 116, 118
    Hintikka-, 156
    lokal, 155
    Rabin-, 134
    WB–Baumautomat, 118
Baumkonkatenation, 123
Baumsprache, 114
    erkennbar, 121
    regulär, 127
    Sternoperator, 125
    von $\mathcal{A}$ akzeptiert, 116, 118
Blatt, 114

Direktes Produkt, 21
DPDL, 153

Ehrenfeucht–Fraïssé Spiele, 54
erfüllbar, 102

Fischer–Ladner–Abschluß, 154
Fischer–Ladner–Abschluß, 104
formale Sprache, 1

gültig, 102
Gewinnstrategie, 56
Gleichung, 22
Gruppe in M, 44

Halbgruppe
    frei, 25
    syntaktisch, 25
Hintikka–Baum, 154
Hintikka–Baumautomat, 156
Hintikka–Automat, 109
Hintikka–Wort, 106
homogene Teilmenge für eine Partition, 77
Homomorphes Bild, 21
Homomorphismus, 11

idempotent, 32
Index, endlich, 10
Interpretation, 101

Klasse formaler Sprachen, 1
Kleene, Satz von, 8
Konfigurationsformeln, 100
Kongruenz
    syntaktisch, 13

Lauf, 116, 118, 134
    erfolgreich, 116, 118, 134
Limes, 133

M–Varietät, 21

Modell, 102
Monadische Logik zweiter Stufe eines Nachfolgers, 83
Monoid
    aperiodisch, 44
    erfüllt Gleichung, 22
    frei, 11
    syntaktisch, 13
Muller–Automat, 80

Negationsnormalform, 104
Nerode, Satz von, 10
Nerode–Rechtskongruenz, 10
NNF, 104

PDL, 157
Pfad, 7
    durch t, 115
    erfolgreich, 7, 64
    unendlich, 64
Potenzmengenkonstruktion, 9
Präfixrelation, 114
Presburger Arithmetik, 95
Programmformeln, 100

Quantorentiefe, 48

Rabin–Baumautomat, 134
Ramsey, Satz von, 77
$\mathcal{R}eg_\Sigma$, 6
$\mathcal{R}eg(T_\Sigma, Z)$, 127

S–Varietät, 25
S1S, 83
S2S, 149
saturierte Relation, 98
schließlich definiert, 22
Segment, 63

Sprache, *siehe auch* ω–Sprache, Baumsprache, ω–Baumsprache
    erkennbar, 8
    regulär, 6, 42
    sternfrei, 4, 43
    verallgemeinert–definit, 30
    von Formel definiert, 27

T_Σ, 114
T_Σ^ω, 132
t_u, 115
$\mathcal{TOT}$, 61

Übergangsfunktion, 9
Übergangsmonoid, 13
Unterbaum, 115
Untermonoid, 21

Wort
    unendlich, 62
WS1S, 94
WS2S, 150

Zug, 54
Zustand
    aktiv, 144
    passiv, 144