11

Entscheidungsverfahren für ω-Automaten

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

Proposition 17. *Das Leerheitsproblem für einen nicht-deterministischen Rabin-Automaten mit n Zuständen und Index k ist in Zeit $O(n^2k)$ lösbar.*

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

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

Korollar 4. *Das Leerheitsproblem für nicht-deterministische Büchi-Automaten mit n Zuständen ist in Zeit $O(n^2)$ lösbar.*

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

Korollar 5. *Das Leerheitsproblem für nicht-deterministische Paritätsautomaten mit n Zuständen und Index k ist in Zeit $O(n^2k)$ lösbar.*

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

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

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

Korollar 6. *Das Universalitätsproblem für deterministische Streett-Automaten mit n Zuständen und Index k ist in Zeit $O(n^2k)$ lösbar.*

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

Proposition 18. *Das Leerheitsproblem für nicht-deterministische Streett-Automaten mit n Zuständen und Index k ist in Zeit $O(n^2k^2)$ lösbar.*

Beweis. Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, \{(G_1, F_1), \ldots, (G_k, F_k)\})$ ein Streett-Automat. Jetzt gilt: $L(\mathcal{A}) \neq \emptyset$ gdw. es einen Lauf $\pi = q_0q_1 \ldots$ gibt, auf dem die Streett-Bedingung erfüllt ist. Zuerst behaupten wir, dass es ausreicht, sich auf ultimativ periodische Läufe der Form uv^ω mit $u \in Q^*$, $v \in Q^+$ zu beschränken. Dass dies hinreichend ist, ist offensichtlich.

Für die Notwendigkeit betrachte einen nicht-ultimativ-periodischen Lauf $\pi = q_0q_1 \ldots$. Da es nur endlich viele Zustände gibt, enthält dieser ein kleinstes Teilstück der Form $p_1 \ldots p_m p_1$, so dass für alle $i = 1, \ldots, k$ gilt: $G_i \cap \{p_1, \ldots, p_m\} = \emptyset$ oder $F_i \cap \{p_1, \ldots, p_m\} \neq \emptyset$. Dann erfüllt auch der Lauf $(p_1 \ldots p_m)^\omega$ die Streett-Bedingung. Schliesslich sieht man leicht, dass das Hinzufügen endlicher Präfixe nichts daran ändert.

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

Wir verwenden einen Hilfsalgorithmus $\mathcal{H}$, welcher als Eingabe einen Graphen $\mathcal{G} = (V, E)$ und eine Menge $\mathcal{S}$ von Streetbedingungen (G, F) mit $G, V \subseteq V$ erhält und entscheidet, ob es in $\mathcal{G}$ eine starke Zusammenhangskomponente C gibt, sodass für alle $(G, F) \in \mathcal{S}$ gilt: $G \cap C \neq \emptyset \Rightarrow F \cap C \neq \emptyset$.

Wir wenden dann diesen Hilfsalgorithmus auf den erreichbaren Teil des Zustandsgraphen von $\mathcal{A}$ an, also $V = \{q \mid q \text{ ist von } q_0 \text{ aus erreichbar } \}$ und $E = \{(q, q') \mid \exists a.q' \in \delta(q, q)\}$. Die Menge $\mathcal{S}$ schließlich umfasst die Streetbedingungen von $\mathcal{A}$ jeweils eingeschränkt auf den erreichbaren Teil.

Es ist aufgrund der vorhergegangenen Analyse klar, dass $L(\mathcal{A}$ nichtleer ist, genau dann wenn der Hilfsalgorithmus den so gebildeten Graphen akzeptiert.

Nun also der Algorithmus $\mathcal{H}(\mathcal{G}, \mathcal{S})$.

- Eingabe: $\mathcal{G} = (V, E)$ und $\mathcal{S} \subseteq \mathcal{P}(V) \times \mathcal{P}(V)$.
- Ausgabe: `True`, falls $\mathcal{G}$ eine starke Zusammenhangskomponente C enthält, sodass $G \cap C \neq \emptyset \Rightarrow F \cup C \neq \emptyset$ für alle $(G, F) \in \mathcal{S}$.
- Zerlege $\mathcal{G}$ in starke Zusammenhangskomponenten.
- Falls es in $\mathcal{G}$ eine starke Zusammenhangskomponente C gibt, mit $C \cap F \neq \emptyset$ für alle $(G, F) \in \mathcal{S}$, so antworte `True`.
- Anderenfalls führe für jede Zusammenhangskomponente C folgendes durch:
  - Wähle $(G, F) \in \mathcal{S}$ mit $C \cap S = \emptyset$.
  - Lösche alle Knoten in G. Formal: Bilde $(\mathcal{G}', \mathcal{S}')$ durch Einschränkung von $(\mathcal{G}, \mathcal{S} \setminus (G, F))$ auf $V \setminus G$.

  - Rufe $\mathcal{H}(\mathcal{G}', \mathcal{S}')$ auf. Falls `True` zurückgeliefert wird, so antworte ebenfalls `True`; falls `False` geantwortet wird, so fahre fort.
- Falls für keine Zusammenhangskomponente `True` zurückgeliefert wurde, so antworte `False`.

Zur Analyse der Laufzeit beachte, dass sich starke Zusammenhangskomponenten z.B. mit Tarjans Algorithmus in Zeit $O(n)$ finden lassen. Sei $T(n,k)$ die worst-case Laufzeit des Algorithmus. Offensichtlich gilt $T(n,0) = O(1)$. Desweiteren gilt für $k > 0$:

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

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

Korollar 7. *Das Universalitätsproblem für nicht-deterministische Büchiautomaten mit n Zuständen ist in Zeit $2^{O(n \log n)}$ lösbar.*

Beweis. Sei $\mathcal{A}$ ein nicht-deterministische Büchiautomat mit n Zuständen. Laut Korollar 3 existiert ein deterministischer Rabin-Automat $\mathcal{A}'$ mit höchstens $2^{O(n \log n)}$ vielen Zuständen und Index höchstens n, so dass $L(\mathcal{A}') = L(\mathcal{A})$ gilt. Sei $\mathcal{A}' = (Q, \Sigma, q_0, \delta, \mathcal{F})$. Aufgrund des Determinismus läßt sich $\mathcal{A}'$ leicht zu dem Streett-Automaten $\overline{A} := (Q, \Sigma, q_0, \delta, \mathcal{F})$ komplementieren, also $L(\overline{A}) = \Sigma^\omega \setminus L(\mathcal{A})$.

Die Behauptung folgt dann sofort aus Proposition 18, da $L(\mathcal{A}) = \Sigma^\omega$ gdw. $L(\overline{A}) = \emptyset$ gilt.

Es gibt noch ein alternatives Verfahren für das Universalitätsproblem für Büchiautomaten, welches auf der in Abschnitt 8.2 definierten Äquivalenzrelation für Büchiautomaten beruht.

Sei für den Rest dieses Abschnitts ein Büchiautomat $\mathcal{B} = (Q, \Sigma, q_0, \delta, F)$ fixiert. Zur Erinnerung:

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

Eine Äquivalenzklasse L dieser Relation ist durch zwei Mengen von Zustandspaaren gekennzeichnet: Die Menge V_L aller Paare (q, q'), sodass $w : q \to q'$ für alle $w \in L$, sowie die Menge F_L aller Paare (q, q'), sodass $w : q \to_F q'$ für alle $w \in L$.

Wie üblich bezeichnet man mit $[w]$ die Äquivalenzklasse des Wortes $w \in \Sigma^*$. Für Mengen von Paaren (Relationen) $R, S \subseteq Q \times Q$ definiert man wie üblich $R; S = \{(q, q') \mid \exists \check{q}.(q, \check{q}) \in R, (\check{q}, q') \in S\}$.

Lemma 23. *Es gilt $V_{[\epsilon]} = \{(q,q) \mid q \in Q\}$ und $V^F_{[\epsilon]} = \{(q,q) \mid q \in F\}$.*

Für $a \in \Sigma$ gilt $V_{[a]} = \{(q, q') \mid q' \in \delta(a, q)\}$ und $V^F_{[a]} = \{(q, q') \mid q' \in \delta(a,q) \wedge (q \in F \vee q' \in F)\}$.

Für $u, v \in \Sigma^*$ gilt: $V_{[uv]} = V_u; V_v$ und $V^F_{[uv]} = V^F_{[u]}; V_{[v]} \cup V_{[u]}; V^F_{[v]}$.

Beweis. Übung.

Man kann sich nun mithilfe dieses Lemmas eine Übersicht über die endlich vielen Äquivalenzklassen verschaffen: Ausgehend von $[a]$ für $a \in \Sigma$ konstruiert man durch sukzessive Anwendung des Lemmas Repräsentationen für weitere Wörter, bis keine neuen Klassen mehr entstehen.

Mithilfe des folgenden Lemmas lässt sich dann die Universalität unmittelbar entscheiden.

Proposition 19. *Es gilt $L(\mathcal{B}) = \Sigma^\omega$ genau dann, wenn für alle Klassen $[u], [v]$ mit $[uv] = [u]$ und $[vv] = [v]$ mit $v \neq \epsilon$ gilt: es gibt einen Zustand q mit $(q_0, q) \in V_{[u]}$ und $(q, q) \in V^F_{[v]}$.*

Beweis. Nehmen wir zunächst an, dass $L(\mathcal{B}) = \Sigma^\omega$. Seien zwei Klassen $[u], [v]$ mit $[uv] = [u]$ und $[vv] = [v]$ mit $v \neq \epsilon$ vorgelegt. Betrachte einen akzeptierenden Lauf von $\mathcal{B}$ auf uv^ω. Wir finden einen Zustand q und Zahlen $i_0 \geq 0, i_1 > 0$ etc. finden, sodass $uv^{i_0} : q_0 \to q$ und $v^{i_1} : q \to_F q$. Wegen $[uv] = [u]$ und $[vv] = [v]$ bedeutet das aber, dass $(q_0, q) \in V_{[u]}$ und $(q, q) \in V^F_{[v]}$.

Sei umgekehrt die Bedingung an die Klassen $[u], [v]$ erfüllt und $w \in \Sigma^\omega$ beliebig. Wie im Beweis von Lemma 18 finden wir mithilfe des Satzes von Ramsey Äquivalenzklassen $[u], [v]$, sowie eine Indexfolge $i_1 < i_2 < i_3 < \ldots$, sodass $[w_{0 \ldots i_0}] = [u]$ und $[w_{i_j+1, \ldots i_{j'}}] = [v]$ für $j' > j$.

Daraus folgt unmittelbar $[vv] = [v]$ und, indem wir u ggf. durch uv ersetzen, auch $[uv] = [u]$.

Also gibt es nach Annahme einen Zustand q mit $(q_0, q) \in V_{[u]}$ und $(q, q) \in V^F_{[v]}$. Dies liefert einen akzeptierenden Lauf von $\mathcal{B}$ auf w.

In der Praxis prüft man für jede idempotente Äquivalenzklasse, also $[v]$ mit $[vv] = [v]$, ob es eine Klasse u gibt mit $[uv] = [u]$ und $(q_0, q) \in V_{[u]} \Rightarrow (q, q) \notin V^F_{[v]}$. Findet man solche Klassen $[u], [v]$, so akzeptiert $\mathcal{B}$ nicht alle Wörter, nämlich insbesondere nicht uv^ω. Findet man keine solchen Beispiele, dann akzeptiert $\mathcal{B}$ jedes beiebige Wort.

11.1 Size-change Termination

In diesem Abschnitt betrachten wir eine Anwendung (The Size-Change Principle for Program Termination, Lee-Jones-Ben Amram, ACM POPL 2001) von Büchiautomaten auf die Frage, ob ein gegebenenes rekursives Programm terminiert. Im Rahmen dieser Anwendung wurde von Jones et al ein neues Entscheidungsverfahren für die Inklusion von Büchiautomaten entwickelt.

11.1.1 Rekursive Programme

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

$$f(x_1, \ldots, x_n) = f_1(\mathbf{t}_1), \ldots, f_l(\mathbf{t}_l)$$

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

Multiplikation:

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

$$\begin{aligned} p(x, y) &= p(x, y - 1) \\ m(x, y, u) &= p(x, u), m(x, y - 1, u) \end{aligned}$$

Ackermann:

Ein dreistelliges Symbol a:

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

Künstlich I:

$$\begin{aligned} f(x, y, z) &= g(x, x, y) \\ g(x, y, z) &= h(x, y - 1, z - 1) \\ h(x, y, z) &= i(x, y - 1, z - 1) \\ i(x, y, z) &= k(x, y - 1, z - 1) \\ k(x, y, z) &= f(x, y - 1, z - 1) \end{aligned}$$

Künstlich II:

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

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

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

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

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

Es gilt: $t(3,10,11,4) \rightsquigarrow t(2,11,3,9) \rightsquigarrow t(3,1,11,8) \rightsquigarrow t(3,3,11,7) \rightsquigarrow t(11,2,3,6) \rightsquigarrow t(11,11,3,5) \rightsquigarrow t(3,10,11,4)$ und damit ist $t(3,10,11,4)$ nicht definiert. Hingegen ist $t(10,10,10,10)$ definiert, wie man mithilfe eines Rechners (dynamische Programmierung!) feststellt.

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

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

11.1.2 Termination als Sprachinklusion

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

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

Wir lassen Klammern und Kommas weg und schreiben so ein Wort als $f_0d_0f_1d_1\ldots$. Im Beispiel mit $F = \{f,g,h,i,k\}$ enthält $CALL$ fünf Wörter, nämlich $w_f = (f1g1h1i1k1)^\omega$, $w_k = k1w_f$, $w_i = i1w_k$, $w_h = h1w_i$, $w_g = g1w_h$.

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

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

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

Es gilt im Beispiel "Multiplikation", dass $(m2)^\omega \in TERM$, da hier die Variable y immer wieder dekrementiert wird. Im Beispiel "Künstlich II" gilt

$(t1)^\omega \in TERM$, ja sogar $(t1 + t3)^\omega \subseteq TERM$. Auf der anderen Seite ist $(t2t3t1t3t1t3)^\omega \notin TERM$.

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

11.1.3 Terminationsanalyse mit Büchiautomaten

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

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

Im Beispiel "Multiplikation" haben wir konkret:

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

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

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

Die vollständige Übergangsfunktion geben wir hier nicht an.

Es sollte nunmehr klar sein, dass $TERM = L(\mathcal{A})$ und somit P für alle Eingaben terminiert, genau dann, wenn $CALL \subseteq L(\mathcal{A})$. Diese Frage kann man direkt mit den im letzten Kapitel gegebenen Algorithmen durchführen. Günstiger ist es jedoch, einen Büchiautomaten $\mathcal{A}'$ zu konstruieren, der die Sprache $\overline{CALL} \cup TERM$ erkennt. Das geht auch ohne Komplementierung von Büchiautomaten, da ein Büchiautomat für $\overline{CALL}$ direkt angegeben werden kann (deterministisch bis auf "Steckenbleiben; nur Endzustände).

Man testet den so entstandenen Büchiautomaten auf Universalität.

12

Alternierende Automaten

12.0.4 Alternierende Büchi-Automaten

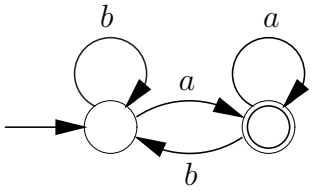
Definition 31. *Ein* alternierender Büchi-Automat *(ABA) ist ein Tupel* $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ *wie bei einem AFA. Ein Lauf eines ABA auf einem Wort* $w \in \Sigma^\omega$ *is ein unendlicher,* Q*-beschrifteter Baum – definiert wie bei einem Lauf eines AFA auf einem endlichen Wort. Solch ein Lauf heißt* akzeptierend*, falls es auf jedem Ast des Baumes unendlich viele Knoten* $n_0, n_1, \ldots$*, so dass* $r(n_i) \in F$ *für alle* $i \in \mathbb{N}$ *gilt.*

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

Proposition 20. *Für jeden NBA* $\mathcal{A}$ *mit* n *Zuständen existiert ein ABA* $\mathcal{A}'$ *mit höchstens* n *Zuständen, so dass* $L(\mathcal{A}') = L(\mathcal{A})$ *gilt.*

Beweis. Analog zum Beweis von Satz **??**.

Beispiel 16. Betrachte die ω-reguläre Sprache $L := \{w \mid |w|_a = \infty\}$. Diese wird z.B. von dem NBA



erkannt. Somit existiert nach Satz 20 auch ein ABA, der L erkennt. Man kann jedoch L auch unter echter Verwendung von Alternierung mit einem ABA erkennen. Das Prinzip dabei ist das folgende. Der ABA liest das nächste Symbol des Eingabeworts. Ist dies ein a, so kann er im selben Zustand verbleiben, denn er muss lediglich prüfen, ob danach wieder ein a vorkommt, usw. Ist dieses jedoch nicht ein a, dann verzweigt er universell. Ein Teil verbleibt in dem

Zustand, in dem auch im nächsten Schritt geprüft wird, ob später wieder ein a vorkommt. Der andere testet, ob wirklich irgendwann noch ein a vorkommt.

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

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

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

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

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

Definition 32. *Sei t ein Lauf eines ABA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ auf einem Wort $w \in \Sigma^\omega$. Der* Rang *eines Knoten n, $rang(n)$ ist das maximale k in einer Sequenz $n_0, \ldots, n_k$, so dass*

- $n = n_0$,
- *für alle $i = 0, \ldots, k-1$ gilt: n_{i+1} ist ein Nachfolger von n_i,*
- $n_k \in F$ *und* $n_i \notin F$ *für alle* $i < k$.

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

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

Lemma 24. *Sei $\mathcal{A}$ ein ABA über Σ und t ein Lauf auf einem $w \in \Sigma^\omega$. Die folgenden Aussagen sind äquivalent.*

a) Der Lauf t ist akzeptierend.
b) Jeder Knoten in t hat endlichen Rang.
c) Jeder Level in t hat endlichen Rang.
d) Unendlich viele Level haben endlichen Rang.

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

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

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

"(c) ⇒ (b)" und "(c) ⇒ (d)" sind trivial.

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

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

Lemma 25. *Sei $\mathcal{A}$ ein ABA über dem Alphabet Σ und $w \in \Sigma^\omega$. Falls $w \in L(\mathcal{A})$ dann gibt es einen gedächtnislosen und akzeptierenden Lauf von $\mathcal{A}$ auf w.*

Beweis. Angenommen $w \in L(\mathcal{A})$, d.h. es gibt einen akzeptierenden Lauf t von $\mathcal{A}$ auf w. Seien $N_0, N_1, \ldots$ die jeweiligen Level von t. Laut Lemma 24 gilt $\mathit{rang}(N_i) < \infty$ für alle $i \in \mathbb{N}$. Falls t nicht gedächtnislos ist, dann gibt es ein kleinstes $i \in \mathbb{N}$, und zwei verschiedene Knoten $n_1, n_2 \in N_i$, deren Unterbäume verschieden sind, für die aber $t(n_1) = t(n_2)$ gilt. O.B.d.A. gelte $\mathit{rang}(n_1) \leq \mathit{rang}(n_2)$. Man sieht leicht, dass das Ersetzen des Unterbaums von n_1 an die Stelle von n_2 nicht die Tatsache zerstört, dass t akzeptierend ist. Dies wird von Lemma 24 gezeigt, denn im entstehenden Lauf haben alle Levels erst recht endlichen Rang.

Sei $t_0 := t$ und t_1 derjenige Lauf, der dadurch entsteht, dass diese Ersetzung auf t_0 solange durchgeführt wird, bis es auf dem i-ten Level keine Knoten mehr gibt, die die Gedächtnislosigkeit verletzen. Da jeder Level eines Lauf nur endlich viele Knoten hat, muss so ein t' existieren. Dieses Verfahren lässt sich nun ins Unendliche iterieren, wobei Läufe $t_0, t_1, \ldots$ erzeugt werden. Sei t^* der Limit dieser Konstruktion, d.h. der eindeutige Baum, der gemeinsame Präfixe

mit allen t_i hat. Beachte, dass sogar gilt: Für jedes $level_i$ in t^* existiert ein $k \in \mathbb{N}$ so dass $level_i$ auch das i-te Level von allen t_j mit $j \geq k$ ist.

Wir behaupten, dass t^* ein akzeptierender und gedächtnisloser Lauf von $\mathcal{A}$ auf w ist. Dazu betrachten wir einen beliebigen Level $level_j$ aus t^*. Da $level_j$ in fast allen t_i ebenso vorkommt, kann es keine zwei Knoten enthalten, die die Gedächtnislosigkeit verletzen. Außerdem ist Rang von $level_j$ auch endlich, denn mit steigendem i kann dieser in den jeweiligen t_i nur abnehmen. Da dies für alle Level gilt, ist t^* insgesamt gedächtnislos. Laut Lemma 24 ist t^* aber auch ein akzeptierender Lauf.

Theorem 32 (Miyano-Hayashi). *Sei $\mathcal{A}$ ein ABA mit n Zuständen. Dann gibt es einen NBA $\mathcal{A}'$ mit höchstens 3^n vielen Zuständen, so dass $L(\mathcal{A}') = L(\mathcal{A})$ gilt.*

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

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

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

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

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

"$\supseteq$" Sei $w = a_0 a_1 \ldots \in L(\mathcal{A})$, d.h. es gibt einen akzeptierenden Lauf von $\mathcal{A}$ auf w. Laut Lemma 25 gibt es dann auch einen gedächtnislosen und akzeptierenden Lauf. Dieser lässt sich als DAG darstellen, so dass jeder Level dieses Lauf eine Teilmenge $P \subseteq Q$ bildet. Seien $P_0, P_1, \ldots$ diese Ebenen. Offensichtlich gilt $P_0 = \{q_0\}$. Außerdem gilt wegen Lemma 5 für alle $i \in \mathbb{N}$:

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

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

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

Man sieht leicht mithilfe von Lemma 5, dass $(P_0, O_0), (P_1, O_1), \ldots$ ein Lauf von $\mathcal{A}'$ auf w ist. Es bleibt lediglich zu zeigen, dass dieser auch akzeptierend ist. Dazu beobachten wir zunächst, dass für alle $i \in \mathbb{N}$ gilt: wenn $O_i \neq \emptyset$, dann ist $\mathit{rang}(O_{i+1}) < \mathit{rang}(O_i)$, wobei wir die O_i als Level in dem Lauf auf w ansehen. Außerdem gilt $\mathit{rang}(O_i)$ nur, wenn $O_i = \emptyset$. Da der Rang eines jeden O_i aber endlich sein muss, denn $|O_i| < \infty$, und laut Lemma 24 hat jeder Knoten in O_i nur endlichen Rang, folgt daraus sofort, dass es unendlich viele i geben muss, so dass $O_i = \emptyset$ ist. Damit ist der Lauf aber akzeptierend.

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

Somit gibt es also unendlich viele Level mit endlichem Rang. Laut Lemma 24 ist der Lauf dann aber akzeptierend.

12.1 Komplementierung von NBAs mittels Alternierung

Definition 33. *Ein* alternierender co-Büchi-Automat *(AcoBA) ist ein Tupel* $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$*, welches definiert ist wie bei einem ABA. Ebenso ist der Lauf eines AcoBA auf einem Wort* $w \in \Sigma^\omega$ *definiert. Solch ein Lauf ist im Gegensatz zu dem Lauf eines ABA jedoch akzeptierend, wenn auf allen Pfaden nur endlich viele Zustände* $q \notin F$ *vorkommen.*

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

Lemma 26. *Für jeden ABA* $\mathcal{A}$ *mit* n *Zuständen gibt es einen AcoBA* $\overline{\mathcal{A}}$ *mit höchstens* n *Zuständen, so dass* $L(\overline{\mathcal{A}}) = \Sigma^\omega \setminus L(\mathcal{A})$ *gilt. Die Umkehrung gilt ebenso.*

Beweis. Übung.

Korollar 8. *Zu jedem NBA* $\mathcal{A}$ *mit* n *Zuständen gibt es einen AcoBA* $\overline{\mathcal{A}}$ *mit höchstens* n *Zuständen, so dass* $L(\overline{\mathcal{A}}) = \Sigma^\omega \setminus L(\mathcal{A})$ *gilt.*

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

Lemma 27. *Sei $\mathcal{A}$ ein AcoBA und $w \in \Sigma^\omega$. Es gilt: Wenn $w \in L(\mathcal{A})$, dann gibt es einen gedächtnislosen Lauf von $\mathcal{A}$ auf w.*

Beweis. Übung.

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

Definition 34. *Sei t ein gedächtnisloser Lauf eines AcoBA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ auf einem Wort $w \in \Sigma^\omega$ – repräsentiert als DAG, dessen Level jeweils höchstens $|Q|$ viele Elemente haben. Zuerst definieren wir induktiv Mengen $R_0^t \supseteq R_1^t \supseteq R_2^t \supseteq \ldots$, von Knoten in t. R_0^t ist die Menge aller Knoten in t, und für alle $i \in \mathbb{N}$:*

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

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

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

Lemma 28. *Sei t ein gedächtnisloser Lauf eines AcoBA $\mathcal{A}$ mit n Zuständen auf einem beliebigen Wort $w \in \Sigma^\omega$. Dann gilt: $R_{2n+1}^t = \emptyset$.*

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

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

Lemma 29. *Sei t ein gedächtnisloser und akzeptierender Lauf eines AcoBA $\mathcal{A}$ auf einem beliebigen Wort $w \in \Sigma^\omega$. Dann gibt es auf jedem Pfad unendlich viele Zustände mit ungeradem Rang.*

Beweis. Übung.

Theorem 33 (Kupferman-Vardi). *Für jeden AcoBA $\mathcal{A}$ mit n Zuständen gibt es einen ABA $\mathcal{A}'$ mit höchstens $2n^2+1$ vielen Zuständen, so dass $L(\mathcal{A}') = L(\mathcal{A})$ gilt.*

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

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

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

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

"$\supseteq$" Man sieht leicht, dass man einen Lauf von $\mathcal{A}'$ auf einem $w \in \Sigma^\omega$ dadurch erhält, indem man in einem akzeptierenden Lauf von $\mathcal{A}$ auf diesem w jeden Knoten mit seinem Rang annotiert. Dass dies möglich ist, zeigt Lemma 28. Laut Lemma 29 kommen auf jedem Pfad unendlich viele Zustände mit ungeradem Rang vor. Damit ist der Lauf des ABA aber akzeptierend.

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

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

Korollar 9. *Für jeden NBA $\mathcal{A}$ mit n Zuständen existiert ein NBA $\overline{\mathcal{A}}$ mit höchstens 2^{4n^2+2} vielen Zuständen, so dass $L(\overline{\mathcal{A}}) = \Sigma^\omega \setminus L(\mathcal{A})$ ist.*

Beweis. Sei $\mathcal{A}$ ein NBA mit n Zuständen. Dieser ist insbesondere ein ABA. Laut Lemma 26 existiert dann ein AcoBA $\overline{\mathcal{A}}$ mit n Zuständen, der dessen Komplement erkennt. Mit Satz 33 lässt dieser sich in einen ABA $\overline{\mathcal{A}}'$ mit $2n^2+1$ vielen Zuständen umwandeln. Schließlich liefert Satz 32 dafür einen NBA mit höchstens der geforderten Zustandszahl. □

Korollar 9 liefert zwar eine asymptotisch schlechtere Komplementierungskonstruktion für NBAs als die von Safra aus Proposition 15. Diese hat jedoch den Vorteil, dass sie symbolisch implementiert werden kann.

12.2 Schwache Automaten

Definition 35. *Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein Automat (nicht-det. / alternierend, Büchi / co-Büchi). Im Falle der Alternierung schreiben wir auch $q' \in \delta(q, a)$, falls q' syntaktisch in der positiv booleschen Formel $\delta(q, a)$ vorkommt.*

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

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

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

Man beachte, dass die in Satz 33 konstrukierten ABAs in Wirklichkeit sogar WABAs sind. Somit existiert zu jedem AcoBA auch ein äquivalenter WABA. Es stellt sich die Frage, ob dies auch für ABAs und WAcoBAs gilt. Dies folgt aus den beiden folgenden Lemmas.

Lemma 30. *Für jeden WABA mit n Zuständen existiert ein äquivalenter WAcoBA mit n Zuständen und umgekehrt.*

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

Lemma 31. *Für jeden WABA $\mathcal{A}$ mit n Zuständen existiert ein äquivalenter WABA $\overline{\mathcal{A}}$ mit höchstens n Zuständen, so dass $L(\overline{\mathcal{A}}) = \Sigma^\omega \setminus L(\mathcal{A})$ gilt.*

Beweis. Folgt sofort aus Lemmas 26 und 30.

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

Proposition 21. *Für jeden ABA $\mathcal{A}$ mit n Zuständen existiert ein äquivalenter WABA $\mathcal{A}'$ mit höchstens $2n^2 + 1$ vielen Zuständen.*

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

Proposition 22. *Es gibt ω-reguläre Sprachen, die nicht von einem WNBA erkannt werden.*

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

Definition 36. *Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein alternierender Büchi-Automat mit $Q = \{q_0, \ldots, q_n\}$. Dieser heißt* schleifenfrei *(oder auch* zählerfrei*, bzw.* sehr schwach*), wenn es eine lineare Ordnung $<$ auf seiner Zustandsmenge gibt, so dass für alle $q, q' \in Q$ gilt: wenn $q' \in \delta(q,a)$ für ein $a \in \Sigma$, dann gilt $q' \leq q$.*

Wir bezeichnen einen solchen Automaten als VWABA.

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

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

Theorem 34. *Für jede LTL-Formel φ existiert ein VWABA $\mathcal{A}_\varphi$ mit $L(\mathcal{A}_\varphi) = L(\varphi)$ und $|\mathcal{A}_\varphi| = O(|\varphi|)$.*

Beweis. Übung.

Theorem 35. *Für jeden VWABA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ über einem Alphabet Σ existiert eine LTL-Formel $\varphi_\mathcal{A}$ mit $L(\varphi_\mathcal{A}) = L(\mathcal{A})$ und $|\varphi_\mathcal{A}| = O(|Q| \cdot 2^{|\Sigma| \cdot m})$, wobei $m := \max\{|\delta(q,a)| \mid q \in Q, a \in \Sigma\}$.*

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

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

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

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

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

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

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

13

Temporale Logik

13.1 Syntax und Semantik

Definition 37. *Sei Σ ein Alphabet. Formeln der* Linear Time Temporal Logic *(LTL) über Σ sind gegeben durch die folgende Grammatik.*

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

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

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

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

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

Proposition 23. *LTL $\leq$ FO.*

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

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

Lemma 32. *Jede LTL-Formel φ ist äquivalent zu einer LTL-Formel φ' in positiver Normalform, so dass $|\varphi'| = O(|\varphi|)$ gilt.*

Beweis. Übung.

Lemma 33. *Für alle $\varphi, \psi \in LTL$ gilt: $\varphi \mathtt{U} \psi \equiv \psi \vee (\varphi \wedge \bigcirc(\varphi \mathtt{U} \psi))$.*

Beweis. Übung.

13.2 Verallgemeinerte Büchi-Automaten

Definition 38. *Ein* verallgemeinerter Büchi-Automat *ist ein Tupel $\mathcal{A} = (Q, \Sigma, I, \delta, F_1, \ldots, F_k)$ mit $I \subseteq Q$ und $F_1, \ldots, F_k \subseteq Q$. Ein Lauf eines solchen Automaten auf einem Wort $w \in \Sigma^\omega$ ist definiert wie bei einem Büchi-Automaten, mit dem Unterschied, dass er in einem beliebigen Zustand $q \in I$ anfängt.*

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

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

Lemma 34. *Für jeden verallgemeinerten Büchi-Automaten $\mathcal{A} = (Q, \Sigma, I, \delta, F_1, \ldots, F_k)$ gibt es einen Büchi-Automaten $\mathcal{A}'$, so dass $L(\mathcal{A}') = L(\mathcal{A})$ und $|\mathcal{A}'| = 1 + |Q| \cdot (k + 1)$.*

Beweis. Übung.

13.3 LTL und Büchi-Automaten

Definition 39. *Der* Fischer-Ladner-Abschluss *einer LTL-Formel φ_0 in positiver Normalform ist die kleinste Menge $FL(\varphi_0)$, für die gilt:*

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

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

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

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

Proposition 24. *Für jede LTL-Formel φ gibt es einen Büchi-Automaten $\mathcal{A}_\varphi$, so dass $L(\mathcal{A}_\varphi) = L(\varphi)$ und $|\mathcal{A}_\varphi| = O(|\varphi| \cdot 2^{2|\varphi|})$ gilt.*

Beweis. Sei $\varphi \in$ LTL. Wegen Lemma 32 nehmen wir an, dass φ in positiver Normalform vorliegt. Seien $\psi_1 \mathtt{U} \chi_1, \ldots, \psi_k \mathtt{U} \chi_k$ alle in φ vorkommenden $\mathtt{U}$-Formeln. Wir definieren einen verallgemeinerten Büchi-Automaten $\mathcal{A}_\varphi := (\mathcal{H}(\varphi), \Sigma, I, \delta, F_1, \ldots, F_k)$ mit

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

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

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

Beachte: $FL(\varphi) \leq 2|\varphi|$. Somit gilt $|\mathcal{H}(\varphi)| \leq 2^{2|\varphi|}$. Laut Lemma 34 gibt es dann einen zu $\mathcal{A}_\varphi$ äquivalenten Büchi-Automaten, der höchstens $O(|\varphi| \cdot 2^{2|\varphi|})$ viele Zustände hat.

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

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

1. Jedes M_i ist eine konsistente Hintikka-Menge.
2. $\varphi \in M_0$.
3. Wenn $\bigcirc\psi \in M_i$, dann $\psi \in M_{i+1}$.
4. Falls der i-te Buchstabe von w ein a ist, dann ist $M_i \cap \Sigma \subseteq \{a\}$.
5. Für jedes $j \in \{1, \ldots, k\}$ und jedes $i \in \mathbb{N}$ gilt: wenn $\psi_j \mathtt{U} \chi_j \in M_i$ dann existiert ein $i' \geq i$, sodass $\{\psi_j \mathtt{U} \chi_j, \chi_j\} \subseteq M_{i'}$.

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

"$\Rightarrow$" Sei $w \in L(\mathcal{A}_\varphi)$. D.h. es existiert ein akzeptierender Lauf $M_0, M_1, \ldots$ von $\mathcal{A}$ auf w. Man zeigt nun leicht durch Induktion über den Formelaufbau,

dass für alle $i \in \mathbb{N}$ und alle $\psi \in FL(\varphi)$ folgendes gilt. Wenn $\psi \in M_i$ dann $w, i \models \psi$.

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

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

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

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

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

Betrachten wir schließlich den Fall $\psi = \theta \mathtt{R} \chi$. Wir beachten zunächst, dass $w, i \models \theta \mathtt{R} \chi$ genau dann, wenn $w, i \models \mathtt{G}\chi \vee \chi \mathtt{U} \theta$.

Wenn jetzt also $\theta \mathtt{R} \chi \in M_i$ ist, so muss auch $\chi \in M_i$ sein und außerdem $\theta \in M_i$ oder (wegen der Transitionsrelation) $\theta \mathtt{R} \chi \in M_{i+1}$; dann also $\theta \in M_{i+1}$ etc. Entweder haben wir also $\theta \in M_j$ für alle $j \geq i$ und somit unter Verwendung der Induktionshypothese $w, i \models \mathtt{G}\theta$ oder aber es kommt irgendwann $\chi \in M_j$, aber $\theta_k \in M_k$ für alle $i \leq k < j$. In diesem Fall haben wir $w, i \models \chi \mathtt{U} \theta$.

13.4 Model-Checking

Übungsaufgaben

Übung 14. Beweise Lemma 10.

Übung 15. Beweise Lemma 11.

Übung 16. Betrachte die folgende Aussage: Sei $\mathcal{A}$ ein NFA, der die Sprache L erkennt. Wenn man diesen als NBA auffasst, so erkennt er die Sprache L^ω.

a) Finde ein Gegenbeispiel für diese Aussage.
b) Zeige, dass diese Aussage wahr ist, wenn $\mathcal{A}$ deterministisch ist.

Übung 17. Seien Σ, Δ Alphabete und $h : \Sigma \to \Delta$ eine Funktion. Man erweitert h auf endliche und unendliche Wörter in der üblichen Weise zu einem Homomorphismus. Ist $L \subseteq \Sigma^\omega$ so bezeichnet $h(L) \subseteq \Delta^\omega$. Zeige, dass die Klasse der ω-regulären Sprachen unter Homomorphismen und inversen Homomorphismen abgeschlossen ist.

Übung 18. Beweise Theorem 20.

Übung 19. Beweise Lemma 19.

Übung 20. Seien m, k gegeben, so dass $2 \leq m \leq k$. Zeige, dass es für jeden NPA $\mathcal{A}$ der Größe n und vom Index (m, k) einen äquivalenten NPA $\mathcal{A}'$ der Größe n und vom Index $(m-2, k-2)$ gibt.

Übung 21. Beweise Theorem 23.

Übung 22. Beweise Theorem 25.

Übung 23. Sei $\mathcal{A}$ ein NPA bzw. ein NSA. Definiere jeweils einen äquivalenten NMA derselben Größe. Lässt sich dessen Index in Abhängigkeit von der Größe und des Index von $\mathcal{A}$ abschätzen?

Übung 24. Beweise Thm. 28.

Übung 25. Beweise Thm. 30.

Übung 26. Zeige, dass es zu jedem NBA $\mathcal{A}$ der Größe n und NcoBA $\mathcal{A}'$ der Größe n' einen NPA $\mathcal{B}$ der Größe $n \cdot n'$ und vom Index $(1, 3)$ gibt, so dass $L(\mathcal{B}) = L(\mathcal{A}) \cap L(\mathcal{A}')$.

Übung 27. Beweise Thm. 29. *Hinweis:* Dazu lässt sich die Miyano-Hayashi-Konstruktion aus Thm. 32 verwenden.

Übung 28. Formuliere die in Bsp. 15 genannte Bedingung jeweils als Streett- und Muller-Bedingung, so dass der dort gezeigte Automat mit diesen Akzeptanzbedingungen jeweils zu einem Streett- bzw. Muller-Automat wird, der die dort genannte Sprache akzeptiert.

Teil IV

Endliche Bäume

14

Automaten auf endlichen Bäumen

Am Anfang des vorherigen Kapitels haben wir gesehen, dass sich ein Wort als eine Menge von Positionen mit einer Nachfolgerfunktion auffassen lässt. Dies kann man in natürlicher Weise auf mehrere Nachfolgerfunktionen erweitern. So erhalt man eben Bäume, die in der Informatik mindestens so eine wichtige Rolle wie Wörter spielen, siehe z.B. Parse-Bäume kontext-freier Grammatiken, abstrakte Datentypen, XML-Dokumente, etc.

14.1 Top-down vs. Bottom-Up

Definition 40. *Ein endlicher Baum ist eine präfix-abgeschlossene, endliche Teilmene T von $\mathbb{N}^*$, d.h. ist $wn \in T$ für ein $w \in \mathbb{N}^*$ und ein $n \in \mathbb{N}$, dann ist auch $w \in T$.*

Im folgenden ist Σ wie üblich ein endliches Alphabet, aber mit einer Funktion $\sigma : \Sigma \to \mathbb{N}$, die jedem Symbol eine Stelligkeit *zuordnet. Ein Σ-Baum ist eine partielle Abbildung $t : T \to \Sigma$ für einen Baum T, so dass für alle $w \in \mathbb{N}^*$, $a \in \Sigma$ und $n \in \mathbb{N}$ gilt: $w \in T$ und $t(w) = a$ und $\sigma(a) = n$ gdw. für alle $i \in \mathbb{N}$ gilt: $wi \in T$ gdw. $i < n$. .*

Sei $\Sigma_n := \{a \in \Sigma \mid \sigma(a) = n\}$ und $\mathcal{T}_\Sigma$ die Menge aller Σ-Bäume.

Beachte: Bäume sind hier endlich-verzweigend, nicht nur, weil sie sowieso nur endlich sind, sondern auch, weil die Stelligkeitsfunktion des Alphabets auch nur endlich viele Nachfolgerknoten zulässt.

Beispiel 17. Sei $\Sigma = \{+, *, -, 0, 1, \cdot\}$ mit $\sigma(+) = \sigma(*) = \sigma(\cdot) = 2$, $\sigma(-) = 1$ und $\sigma(0) = \sigma(1) = 0$. Der arithmetische Ausdruck $-(7*3)+(-4+9)$ lässt sich in binärer Kodierung z.B. wie in Abbildung 14.1 als Σ-Baum modellieren.

Definition 41. *Ein* Bottom-Up-Baumautomat *(BUBA) ist ein Tupel $\mathcal{A} = (Q, \Sigma, \delta, F)$ mit*

- *endlicher Zustandsmenge Q,*

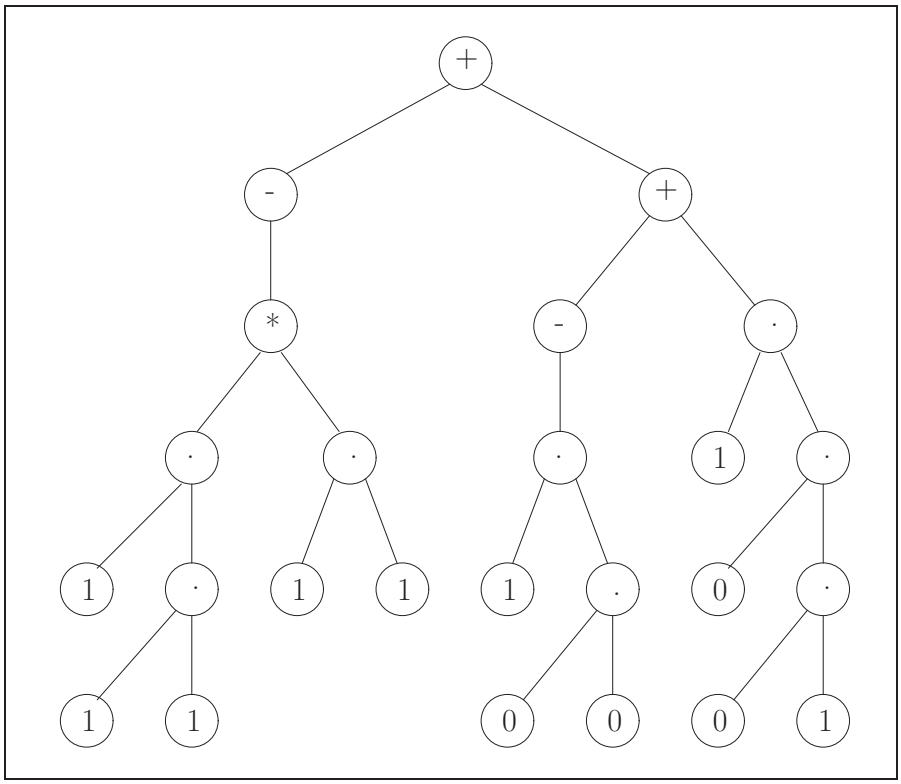


Abb. 14.1. Ein arithmetischer Ausdruck als endlicher Baum.

- *Alphabet Σ mit Stelligkeitsfunktion σ,*
- *Transitionsrelationen $\delta = \{\delta_a \mid a \in \Sigma\}$ wobei $\delta_a : Q^n \to 2^Q$, falls $\sigma(a) = n$,*
- *Endzustandsmenge $F \subseteq Q$.*

Ein Lauf von $\mathcal{A}$ auf einem Σ-beschrifteten Baum t ist eine Abbildung $r : dom(t) \to Q$, so dass für alle $w \in dom(t)$ gilt:

- *$r(w) \in \delta_a(r(w0), \ldots, r(w(n-1)))$ mit $a = t(w)$ und $n = \sigma(a)$ sonst.*

Solch ein Lauf heißt akzeptierend*, falls $r(\epsilon) \in F$. Wie üblich definieren wir $L(\mathcal{A}) \subseteq \mathcal{T}_\Sigma$ als die von $\mathcal{A}$ akzeptierte (Baum-)Sprache, also die Menge aller Σ-Bäume, auf den es einen akzpetierenden Lauf von $\mathcal{A}$ gibt.*

Ein BUBA beschriftet also die Knoten eines Baums mit Zuständen. Er fängt bei den Blättern an und hört an der Wurzel auf.

Beispiel 18. Sei $\Sigma = \{\vee, \wedge, \neg, 0, 1\}$ mit $\sigma(\vee) = \sigma(\wedge) = 2$, $\sigma(\neg) = 1$ und $\sigma(0) = \sigma(1) = 0$. Die Sprache aller booleschen Ausdrücke, die zu 1 auswerten, ist BUBA-erkennbar. Dies wird z.B. von dem BUBA $\mathcal{A} = (\{0, 1\}, \Sigma, \delta, \{1\})$ mit

- $\delta_0() = \{0\}$,
- $\delta_1() = \{1\}$,
- $\delta_\neg(0) = \{1\}$, $\delta_\neg(1) = \{0\}$,
- $\delta_\vee(q_1, q_2) = \{q_1 \vee q_2\}$,
- $\delta_\wedge(q_1, q_2) = \{q_1 \wedge q_2\}$

getan.

Dieser Automat hat sogar noch eine besondere Eigenschaft: Er ist deterministisch.

Definition 42. *Ein BUBA $\mathcal{A} = (Q, \Sigma, \delta, F)$ ist* deterministisch *(DBUBA), wenn für alle $a \in \Sigma$ und alle $q_1, \ldots, q_n \in Q$ mit $n = \sigma(a)$ gilt: $|\delta_a(q_1, \ldots, q_n)| = 1$.*

Proposition 25. *Eine Baumsprache wird von einem BUBA erkannt, gdw. sie von einem DBUBA erkannt wird.*

Beweis. Richtung $\Leftarrow$ wird durch eventuelles Hinzufügen eines Zustands, aus dem nichts erkannt wird, erreicht. Richtung $\Rightarrow$ wird wie bei endlichen Wörtern durch eine einfache Potenzmengenkonstruktion gezeigt.

Definition 43. *Ein* Top-Down-Baumautomat *(TDBA) ist ein Tupel $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ mit*

- *endlicher Zustandsmenge Q,*
- *Alphabet Σ mit Stelligkeitsfunktion σ,*
- *Anfangszustand $q_0 \in Q$,*
- *Transitionsfunktionen $\delta = \{\delta_a \mid a \in \Sigma, \sigma(a) > 0\}$, wobei $\delta_a : Q \to 2^{Q^n}$ falls $\sigma(a) = n > 0$,*
- *Endzustandszuweisung $F : \Sigma_0 \to 2^Q$.*

Ein TDBA heißt deterministisch*, falls $|\delta_a(q)| = 1$ für alle $a \in \Sigma$ und alle $q \in Q$.*

Ein Lauf eines TDBA auf einem Baum t ist ein $r : dom(t) \to Q$ mit

- *$r(\epsilon) = q_0$,*
- *$(r(w0), \ldots, r(w(n-1))) \in \delta_a(r(w))$ für alle $a \in \Sigma$ und alle $w \in dom(t)$ mit $t(w) = a$, $\sigma(a) = n$.*

Solch ein Lauf heisst akzeptierend*, falls $r(w) \in F(a)$ für alle Blätter w mit $t(w) = a$.*

Ein TDBA beschriftet einen Baum also beginnend mit der Wurzel und endend in den Blättern.

Beispiel 19. Sei $\Sigma = \{a, b, c\}$ mit $\sigma(a) = \sigma(b) = 2$, $\sigma(c) = 0$. Die Sprache L aller Bäume, in denen mindestens ein b vorkommt, ist TDBA-erkennbar, z.B. von dem Automaten $\mathcal{A} = (\{-, +\}, \Sigma, -, \delta, F)$ mit

- $\delta_a(-) = \{(-,+),(+,-)\}$, $\delta_a(+) = \{(+,+)\}$,
- $\delta_b(-) = \{(+,+)\}$, $\delta_b(+) = \{(+,+)\}$,
- $\delta_b(-) = \{(+,+)\}$, $\delta_b(+) = \{(+,+)\}$,

und $F(c) = \{+\}$.

Theorem 36. *Die folgenden Aussagen sind für endliche Baumsprachen L über einem Alphabet Σ äquivalent.*

1. *L wird von einem BUBA erkannt.*
2. *L wird von einem TDBA erkannt.*

Beweis. Übung.

Die Äquivalenz von TDBA und BUBA beruht darauf, dass Nichtdeterminismus zur Verfügung steht. Sei $a \in \Sigma_n$ für ein $n \in \mathbb{N}$. Dann ist die Transitionsrelation δ_a eines BUBA vom Typ $Q^n \times Q$. Beachte, dass $Q^n \times Q \simeq Q \times Q^n$ ist, welches der Typ einer Transitionsrelation δ_a eines TDBA ist. Bei einem DTDBA hat δ_a jedoch den Typ $Q \to Q^n$, und i.A. gilt $Q \to Q^n \not\simeq Q^n \to Q$.

Insbesondere gilt i.A.

$$|Q \to Q^n| \;=\; (|Q|^n)^{|Q|} \;=\; |Q|^{n\cdot|Q|} \;<\; |Q|^{(|Q|^n)} \;=\; |Q^n \to Q|$$

Theorem 37. *Es gibt reguläre Baumsprachen, die nicht von einem DTDBA erkannt werden.*

Beweis. Sei $\Sigma = \{a,b,c\}$ mit $\sigma(a) = 2$ und $\sigma(b) = \sigma(c) = 0$. Betrachte die endliche Baumsprache $L := \{a(b,c), a(c,b)\}$. Es ist leicht zu sehen, dass diese von einem TDBA erkannt wird.

Angenommen, $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ist ein DTDBA mit $L(\mathcal{A}) = L$. Da $a(b,c) \in L(\mathcal{A})$ gibt es Zustände $q_1, q_2 \in Q$ mit $\delta_a(q_0) = \{(q_1,q_2)\}$, und es muss auch noch $q_1 \in F(b)$ und $q_2 \in F(c)$ gelten. Da aber auch $a(c,b) \in L(\mathcal{A})$ ist, gilt ebenfalls $q_1 \in F(c)$ und $q_2 \in F(b)$. Dann ist aber auch $\{a(b,b), a(c,c)\} \subseteq L(\mathcal{A})$. □

14.2 Reguläre Baumsprachen, Abschlusseigenschaften

Eine Baumsprache (Menge von Bäumen über dem gleichen Alphabet) ist *erkennbar*, wenn sie die Sprache eines BUBA ist.

Proposition 26. *Erkennbare Baumsprachen sind abgeschlossen unter den Operationen Vereinigung, Durchschnitt, Komplement.*

Beweis. wie bei Wörtern. Übung.

Definition 44 (Baumkonkatenation). *Sei x ein 0-stelliges Element aus dem Alphabet Σ. Seien A und B Baumsprachen über Σ. Die Baumsprache $A[x{:=}B]$ (alternative Notation $A \cdot_x B$) ist wie folgt definiert:*

$$\begin{aligned} A[x{:=}B] &:= \bigcup_{t\in A} t[x{:=}B] \\ x[x{:=}B] &:= B \\ y[x{:=}B] &:= y \quad \textit{wenn } y \neq x \\ f(t_1 \dots t_k)[x{:=}B] &:= f(t_1[x{:=}B] \dots t_k[x{:=}B]) \end{aligned}$$

Intuitiv besteht $A[x{:=}B]$ aus allen Bäumen, die man erhält, indem man in einem Baum t aus A alle mit x beschrifteten Blätter durch Bäume aus B ersetzt (es dürfen unterschiedliche Bäume für verschiedene Vorkommen von x sein.).

Analog definiert man $A[x_1{:=}B_1, \dots, x_n{:=}B_n]$:

$$\begin{aligned} A[x_1{:=}B_1, \dots, x_n{:=}B_n] &:= \bigcup_{t\in A} t[x_1{:=}B_1, \dots, x_n{:=}B_n] \\ x_i[x_1{:=}B_1, \dots, x_n{:=}B_n] &:= B_i \\ y[x_1 = B_1, \dots, x_n = B_n] &:= y \quad \text{, wenn } y \notin \{x_1, \dots, x_n\} \\ f(t_1, \dots, t_k)[x_1{:=}B_1, \dots, x_n{:=}B_n] &:= f(t_1[s], \dots, t_k[s]), \text{ wobei } s = t_1{:=}B_1, \dots, t_k{:=}B_k \end{aligned}$$

Bemerkung: $A[x{:=}B, y{:=}C]$ ist im allgemeinen nicht dasselbe wie $A[x{:=}B][y{:=}C]$. Z.B.: $x[x{:=}y, y{:=}b] = y$, aber $x[x{:=}y][y{:=}b] = b$.

Proposition 27. *Sind $A, B_1, \dots, B_n$ erkennbar, so auch $A[x_1{:=}B_1, \dots, x_n{:=}B_n]$.*

Beweis. durch Konstruktion eines TDBA.

Definition 45 (Baumstern). *Sei x ein einzelnes Symbol, L eine Baumsprache. Wir definieren die Baumsprache L^{*x} durch $L^{*x} = \bigcup_k L_k$, wobei $L_0 = \{x\}, L_{k+1} = L_k \cup L[x{:=}L_k]$.*

Proposition 28. *Ist L erkennbar, so auch L^{*x}.*

Beweis. durch Konstruktion eines TDBA.

Definition 46 (reguläre Baumsprache). *Eine Baumsprache L ist regulär, wenn sie aus den endlichen Sprachen mit Vereinigung, Baumkonkatenation, Baumstern erzeugt werden kann. Man schreibt $L \in Reg(\Sigma)$.*

Proposition 29. *Eine Baumsprache L über Σ ist erkennbar genau dann, wenn eine endliche Menge 0-stelliger Hilfssymbole Z existiert, sodass $L \in Reg(\Sigma \cup Z)$. Beachte: die Symbole von Z kommen in L selbst nicht vor.*

Beweis. L werde von TDBA $\mathcal{A} = (\dots Q \dots)$ erkannt. Wir nehmen $Z = Q$ und definieren einen Automaten $\mathcal{A}'$ auf dem erweiterten Alphabet $\Sigma \cup Q$ (disjunkte Vereinigung) durch $F'(x) = F(x)$, falls $x \in \Sigma$ und $F'(q) = \{q\}$, falls $q \in Z$.

O.B.d.A. sei $Q = \{1, \dots, k\}$.

Sei K eine Teilmenge von Q, $h \leq k$ (möglicherweise $h = 0$), $i \in Q$. Wir bezeichnen mit $L(K, h, i)$ die Sprache über $\Sigma + K$, die von $\mathcal{A}'$ beginnend in i erkannt wird durch einen Lauf mit der Eigenschaft, dass innere Knoten (weder Wurzel noch Blatt) mit Zuständen $\leq h$ beschriftet sind. Insbesondere bedeutet das, dass Blätter $x \in \Sigma$ mit Endzuständen aus $F(x)$ beschriftet sind und dass Zustandsblätter $q \in K$ mit sich selbst beschriftet sind (wg. $F'(q) = \{q\}$).

Die Sprache $L(K, 0, q)$ ist endlich und somit erkennbar. Außerdem ist

$$L(K, h{+}1, q) = L(K, h, q) \cup L(K \cup \{h{+}1\}, h, q)[h{+}1 := U]$$
$$\text{wobei } U = L(K \cup \{h{+}1\}, h, h{+}1)^{*(h+1)}[h{+}1 := L(K, h, h{+}1)]$$

Dies sieht man dadurch, dass man in einem Element von $L(K, h+1, q)$ alle Vorkommnisse des Zustands $h+1$ herauspräpariert.

Somit sind alle $L(K, h, q)$ regulär und damit auch $L(\mathcal{A}) = L(\emptyset, k, q_0)$.

Proposition 30. *Für reguläre Baumsprachen L_1, L_2 (präsentiert durch Automat oder Ausdruck) sind folgende Probleme entscheidbar: $L_1 = \emptyset, L_1 = L_2, L_1 \subseteq L_2$.*

Beweis. Es genügt zu entscheiden, ob ein DBUBA die leere Sprache erkennt. Alles andere lässt sich darauf zurückführen. Sei also ein DBUBA $\mathcal{A}$ gegeben. Wir bestimmen iterativ die Menge M aller Zustände q, sodass es einen Baum t gibt mit $\mathcal{A}(t) = q$. Sei U_0 die Menge aller Zustände, die als Anfangsbeschriftung in Erscheinung treten. Ist U_n schon definiert, so sei U_{n+1} die Menge U_n zusammen mit allen Zuständen q, derart, dass es ein Symbol f gibt und Zustände $q_1, \dots q_k \in U_n$ mit $\delta(f, q_1, \dots q_k) = q$. Die gesuchte Menge M ist die Vereinigung aller U_n. Nach spätestens $|Q|$-Schritten ist aber diese Vereinigung erreicht, sodass sie in polynomialer Zeit berechnet werden kann. Der Automat akzeptiert nun die leere Sprache, genau dann, wenn $M \cap F = \emptyset$.

Dieses Verfahren läuft in polynomialer Zeit, wenn die Ausgangssprache als DBUBA gegeben ist. Ist er hingegen als regulärer Ausdruck (möglicherweise gar mit Negation) oder als nichtdet. Automat gegeben, so muss zunächst determinisiert werden. Man kann zeigen [Seidl90], dass das Leerheitsproblem für in diesen Formen gegebene Sprachen EXPTIME vollständig ist (zum Vergleich PSPACE vollständig bei Wörtern).

14.3 Komplementierung von Topdown-Baumautomaten

Deterministische BUBA sind leicht zu komplementieren durch Vertauschen von End- und Nicht-Endzuständen. Für Topdown-Automaten ist es nicht so

leicht, da sie erstens i.a. nichtdeterministisch sind und zweitens die Akzeptanzbedingung lautet: jedes Blatt ist mit einem passenden Endzustand beschriftet; davon ist das Komplement aber "es gibt ein Blatt, das mit einem unpassenden Endzustand beschriftet ist", was nicht von derselben Form ist.

Es ist nützlich, sich zu überlegen, wie man Topdown-Automaten direkt, ohne den Umweg über Bottomup-Automaten komplementieren kann, da sich nur die Topdown-Automaten auf unendliche Bäume verallgemeinern lassen.

Sei also ein nichtdeterministischer Topdownautomat $\mathcal{A}$ und ein Baum $t = (T, t)$ gegeben. Sei n die maximale Stelligkeit und $D = \{0, \ldots, n-1\}$ die Menge der möglichen Richtungen, in die man in einem Knoten verzweigen kann. Positionen im Baum lassen sich so als bestimmte Wörter über D^* auffassen. Wenn $w \in D^*$, so bezeichnen wir wie üblich mit $t(w)$ die Beschriftung des durch w adressierten Knotens in t. Wenn $t(w) = a$ und $\sigma(a) = k$, dann ist $k \leq n$ und $\{i \mid wi \in T\} = \{0, \ldots, k-1\}$.

Wir betrachten folgendes Zweipersonenspiel zwischen den Spielern A ("Automat") und P ("Pfadfinder").

1. Positionen des Spiels sind Paare der Form (w, q) wobei $w \in D^* \cap T$ und $q \in Q$, sowie Paare der Form $(w, \mathbf{q})$, wobei $w \in D^* \cap T$ und $\mathbf{q} \in Q^{\sigma(t(w))}$. Bei Positionen der Form (w, q) ist A am Zug; bei Positionen der Form $(w, \mathbf{q})$ ist P am Zug.
2. Startposition ist (ϵ, i), wobei i der Anfangszustand ist.
3. In der Position (w, q) wählt A ein Tupel $\mathbf{q} \in \delta_{t(w)}(q)$ und erreicht die Position $(w, \mathbf{q})$.
4. In der Position $(w, \mathbf{q})$ wählt P eine Richtung $i < m$, wobei $m = \sigma(t(w))$ die Stelligkeit des Symbols $t(w)$ ist und es wird die Position (wi, q') eingenommen, wobei q' die i-te Komponente des Vektors $\mathbf{q}$ ist.
5. Das Spiel endet, wenn eine Position (w, q) erreicht wird, wobei $t(w)$ ein Blatt ist, also $\sigma(t(w)) = 0$. Es gewinnt dann A, falls $q \in F(t(w))$ und es gewinnt P, falls $q \notin F(t(w))$.

Proposition 31. *Ein Baum wird genau dann von $\mathcal{A}$ akzeptiert, wenn Spieler A eine Gewinnstrategie besitzt.*

Beweis. Übung.

Proposition 32. *Hat A keine Gewinnstrategie, so besitzt P eine positionale Gewinnstrategie, d.h. es gibt eine Funktion*

$$s : D^* \times Q^{\leq n} \to D$$

sodass P gewinnt, falls er in der Position $(w, \mathbf{q})$ die Richtung $s(w, q, \mathbf{q})$ wählt.

Beweis. Durch Induktion über die Tiefe des Baumes.

Unendliche Spiele

Die Existenz positionaler Gewinnstrategien gilt bei allen endlichen Spielen und kann durch Induktion über die Spieldauer nachgewiesen werden (ist man in einer Position von der aus man gewinnen kann, so ziehe man auf eine Position von der aus das auch noch der Fall ist). Bei unendlichen Spielen kann es erforderlich sein, sich gewisse Daten über die Spielhistorie zu merken und in die Entscheidungen über den nächsten Zug einfließen zu lassen. Ein Beispiel bildet das Dziembowski-Jurdziński-Walukiewicz-Spiel, bei dem zwei Spieler abwechselnd und ad infinitum Buchstaben in $\{A, B, C, D\}$ (Buchstabenspieler) und Zahlen aus $\{1, 2, 3, 4\}$ (Zahlenspieler) nennen. Der Zahlenspieler gewinnt, wenn die größte von ihm unendlich oft gespielte Zahl mit der Anzahl der vom Buchstabenspieler unendlich oft gespielten Buchstaben übereinstimmt. Man kann sich überlegen, dass der Zahlenspieler hier eine Gewinnstrategie hat, aber keine positionale.

Anwendung von Currying

Die Gewinnstrategie in unserem Spiel kann als Funktion aufgefasst werden, die zu jeder Baumposition $w \in D^*$ eine Funktion $s = s_w : Q^{\leq n} \to D$ liefert.

Wir definieren nun $S := D^{Q^{\leq n}}$ als die *endliche* Menge aller solcher Funktionen.

Nun wird also t nicht von A akzeptiert, wenn P eine Gewinnstrategie besitzt, es also eine Beschriftung des Baumes t mit Elementen von S gibt, derart, dass für alle (von A gespielten) Transitionsfolgen $\mathbf{m} \in (Q^{\leq n})^*$ der von $\mathbf{m}$ und der Strategiebeschriftung induzierte Pfad w in einer Gewinnposition für P landet. Wir können nun die Quantifikation über die Pfade nach außen ziehen:

Der Baum t wird nicht von A akzeptiert, genau dann wenn für alle Pfade w in t folgendes gilt:

Für alle Transitionsfolgen $\mathbf{m} \in (Q^{\leq n})^*$, die erstens legal sind und zweitens (zusammen mit den Strategiebeschriftungen) den gegebenen Pfad w induzieren, gilt, dass eine Gewinnposition für P erreicht wird.

Dies soll durch einen nichtdeterministischen Topdownbaumautomaten ausgedrückt werden. Die Form sieht schon mal ganz gut aus: Eine universelle Quantifizierung über Pfade steht ganz außen. Wir gehen wie folgt vor. Zunächst bauen wir einen Automaten, der auf mit $\Sigma \times S$ beschrifteten Bäumen arbeitet. Intuitiv muss dieser Automat für jeden Pfad w abprüfen, ob für jedes gleichlange Wort aus Transitionsfolgen welches diesen Pfad induziert, die induzierte Zustandsfolge in A in einem Zustand q mit $q \notin F(x)$ landet (x die Blattbeschriftung am Ende des Pfades).

Diese Sprache über Pfaden ist regulär und kann deshalb mit einem endlichen Wortautomaten, den man auf jedem Pfad mitlaufen lässt, überprüft werden. Genauer gesagt, sei $\mathcal{M}$ ein DFA über $\Sigma \times S \times D$, der folgende Sprache erkennt:

$$L(\mathcal{M}) = \{(a_1, s_1, d_1) \dots (a_l, s_l, d_n) \mid \forall \mathbf{q}_1 \dots \mathbf{q}_l . \forall q_1 \dots q_n .$$
$$q_1 = i \Rightarrow \forall 1 \leq i \leq l . \mathbf{q}_i \in \delta_{a_i}(q_i) \Rightarrow \forall 1 < i \leq l . d_i = s_i(\mathbf{q}_{i-1}) \Rightarrow$$
$$q_i = (\mathbf{q}_{i-1})_{d_i} \Rightarrow \sigma(a_l) = 0 \Rightarrow q_l \notin F(a_l)\}$$

Wir bauen als nächstes einen Automaten $\mathcal{A}'$, der einen Baum beschriftet mit $\Sigma \times S$ erkennt, genau dann, wenn alle Pfade in $L(\mathcal{M})$ sind. Dieser Automat hat dieselben Zustände wie $\mathcal{M}$ und die Übergangsfunktion (der Automat ist deterministisch!)

$$\delta^{\mathcal{A}'}_{(a,s)}(q) = (\delta^{\mathcal{M}}((a, s, 0), q), \dots, \delta^{\mathcal{M}}((a, s, \sigma(a) - 1), q))$$

Den ersehnten Automaten $\mathcal{A}''$ über Σ für das Komplement von $\mathcal{A}$ erhalten wir aus $\mathcal{A}'$, indem wir die Strategiebeschriftung raten. Die Zustände von $\mathcal{A}''$ sind dieselben, wie die von $\mathcal{A}'$, die Übergangsrelation ist definiert durch

$$\delta^{\mathcal{A}''}_a(q) = \{\mathbf{q} \mid \exists s . \mathbf{q} = \delta^{\mathcal{A}'}_{(a,s)}(q)\}$$

Dieser Automat erkennt also wie gewünscht das Komplement von $L(\mathcal{A})$.

Wir können die verwendeten Konstruktionen etwas abstrakter fassen:

Lemma 35. *Sei $L \in Reg(\Sigma \times D)$ eine reguläre Wortsprache. Die assoziierte Baumsprache L^t besteht aus allen Bäumen t, sodass jeder Pfad in t geschrieben als Wort über $\Sigma \times D$ in L ist. Die Sprache L^t ist reguläre Baumsprache.*

Beweis. Übung

Lemma 36. *Sei $p : \Sigma \to \Sigma'$ stelligkeitserhaltende Funktion und $L \in Reg(\Sigma)$ reguläre Baumsprache. Die Baumsprache $p(L) = \{t \mid \exists t' \in L . p(t') = t\}$ ist regulär.*

Hier bezeichnet $p(t')$ den Baum, den man aus t' erhält, indem man die Beschriftungen in t' gemäß p ersetzt.

Beweis. Übung.

15

Anwendungen

15.1 Higher-order matching

Wir betrachten den einfach typisierten Lambdakalkül (funktionale Programme ohne Rekursion mit Typen

$$\tau ::= o \mid \tau_1, \ldots, \tau_n \to \tau$$

Wir schreiben $\tau_1^n \to \tau_2$ als Abkürzung für $\underbrace{\tau_1, \ldots, \tau_1}_{n \text{ Stück}} \to \tau_2$. Wir setzen ein Baumalphabet Σ voraus, wobei wir f mit Stelligkeit n identifizieren mit einer Funktion des Typs $o^n \to o$. Insbesondere identifizieren wir ein Blattsymbol mit einer Konstanten des Typs o. Wir wollen nun Matchinggleichungen dritter Ordnung lösen, insbesondere interessieren wir uns für die Lösungsmenge einer Gleichung

$$x(s_1, s_2, \ldots, s_n) = t$$

wobei s_i variablenfreie Terme von Typen der Form $\tau_i = o, \ldots, o \to o$ sind und t variablenfreier Term vom Typ o ist. Demgemäß ist x eine Variable vom Typ $\tau_1, \ldots, \tau_n \to o$. Die Symbole aus Σ gelten nicht als Variablen.

Beispiel 20. Man bestimme alle Terme, die die Gleichung

$$x(\lambda y_1, y_2.y_1, \lambda y_3.f(y_3, y_3)) = f(a, a)$$

lösen. Zwei mögliche Lösungen sind $x = \lambda x_1, x_2.x_2(a)$ und $x = \lambda x_1, x_2.x_1(f(a, a), a)$. Eine andere Lösung ist $x = \lambda x_1, x_2.x_1(x_2(x_1(x_1(a, \Box), \Box)), \Box)$ wobei für jedes Vorkommen von $\Box$ ein beliebiger Term eingesetzt werden kann.

Fassen wir die λ-gebundenen Variablen als 0-stellige Symbole auf und das Präfix $\lambda x_1, \ldots, x_k.$ als einstelliges Symbol, so können wir λ-Terme mit Baumautomaten verarbeiten. Hier ist ein Baumautomat mit dem zusätzlichen Symbol $\Box$, der gerade alle schematischen Lösungen der gegebenen Gleichungen erkennt.

Es gibt vier Zustände, A, B, F, E
Die Übergänge sind:

$$\begin{array}{ll} a \to A & f(A,A) \to F \\ \Box \to B & x_1(A,B) \to A \\ x_1(F,B) \to F & x_2(A) \to F \\ \lambda x_1, x_2.F \to E & \end{array}$$

Endzustand ist der Zustand E.

Wir verarbeiten $\lambda x_1, x_2.x_1(f(a,a), \Box)$ wie folgt:

$$\begin{array}{l} \lambda x_1, x_2.x_1(f(a,a), \Box) \to \\ \lambda x_1, x_2.x_1(f(A,A), B) \to \\ \lambda x_1, x_2.x_1(F, B) \to \\ \lambda x_1, x_2.F \to E \end{array}$$

Wir verarbeiten $\lambda x_1, x_2.x_1(x_2(x_1(x_1(a, \Box), \Box)), \Box)$ wie folgt:

$$\begin{array}{l} \lambda x_1, x_2.x_1(x_2(x_1(x_1(a, \Box), \Box)), \Box) \to \\ \lambda x_1, x_2.x_1(x_2(x_1(x_1(A, B), B)), B) \to \\ \lambda x_1, x_2.x_1(x_2(x_1(A, B)), B) \to \\ \lambda x_1, x_2.x_1(x_2(A), B) \to \\ \lambda x_1, x_2.x_1(F, B) \to \\ \lambda x_1, x_2.F \to E \end{array}$$

Versuchen wir dagegen $\lambda x_1, x_2.x_2(x_1(f(a,a), \Box))$ zu verarbeiten, so erhalten wir

$$\lambda x_1, x_2.x_2(x_1(f(a,a), \Box)) \to \lambda x_1, x_2.x_2(x_1(f(A,A), B)) \to \lambda x_1, x_2.x_2(x_1(F, B)) \to \lambda x_1, x_2.x_2(F)$$

und wir bleiben stecken. Die Zustände entsprechen Teiltermen der rechten Seite: $A = a$, $F = f(a,a)$. Außerdem haben wir den Sonderzustand B für $\Box$ und E für Ende. Die Idee ist, dass ein Term t mit zwei Variablen x_1 und x_2 im Automaten den Zustand A (bzw. F) liefert, gdw $t[x_1 := \lambda y_1, y_2.y_1, x_2 := \lambda y_3.f(y_3, y_3)]$ sich zu a (bzw. $f(a,a)$) reduziert (per $\beta\eta$-Reduktion). Außerdem ist der Wert von t der Zustand B, falls sich $t[x_1 := s_1, x_2 := s_2]$ zu $\Box$ reduziert.

Dies beweist man durch Induktion über t (in $\beta\eta$-Normalform vorausgesetzt.)

Betrachten wir nun den allgemeinen Fall einer Matchinggleichung $x(s_1, s_2, \ldots, s_n) = t$.

Für jeden Teilterm u der rechten Seite t führen wir einen Zustand q_u ein und dazu noch zwei Sonderzustände $q_\Box$ und E. Die Regeln sind wie folgt:

$\lambda x_1, \ldots, x_n.q_t \to E$
$\Box \to q_\Box$
$f(q_{u_1}, \ldots, q_{u_n}) \to q_v$, falls $v = f(u_1, \ldots, u_n)$ ein Teilterm der rechten Seite ist. Hier $u_i, v \neq \Box$
$x_i(q_{u_1}, \ldots, q_{u_n}) \to q_v$, falls $s_i(u_1, \ldots, u_n) =_{\beta\eta} v$. Hier $v \neq \Box$, aber vielleicht $u_i = \Box$

Diese Methode lässt sich auf beliebige Matchinggleichungen dritter Ordnung verallgemeinern, also Gleichungen der Form a(x1,...,xn) = b, wobei x_i Variablen dritter Ordnung sind, etwa mit Typen wie $(o \to o) \to o$, und a, b variablenfrei sind.

Die Entscheidbarkeit des Matchingproblems für beliebige Ordnung ist ein berühmtes offenes Problem. In 2006 hat C. Stirling eine Arbeit vorgelegt, in der die Entscheidbarkeit gezeigt wird. Sollte diese Arbeit Bestand haben, so wäre dieses Problem gelöst.

15.2 Baumautomaten und XML

Weitere wichtige Anwendungen für Baumautomaten ergeben sich im Bereich semistrukturierter Daten (XML). Sie werden hier insbesondere zur Typüberprüfung (Typen = XML Schemas) von Dokumenten und Transformationsskripten eingesetzt. Sind solche Transformationsskripten in funktionalem Stil geschrieben, so kann man eine Kombination von ML-Typüberprüfung und Baumautomaten einsetzen um eine approximative Typprüfung im folgenden Sinne durchzuführen:

Gegeben seien Schemas `In` und `Out`, sowie ein Transformationsskript P. Bestätigt die Typüberprüfung für P den Typ `In` $\to$ `Out`, so liefert P für Eingaben des Typs `In` stets Ausgaben des Typs `Out`. Es gibt aber Skripten P, die diese semantische Eigenschaft haben und dennoch von der Typüberprüfung zurückgewiesen werden.

Der Vorteil dieses Ansatzes ist, dass die Typüberprüfung recht einfach ist und gleichzeitig die einsetzbare Programmiersprache universell ist.

Alternativ kann man eine exakte Analyse erreichen, indem man die erlaubten Skripten einschränkt, etwa auf Baumautomaten mit Ausgabe und gleichzeitig die Analyse komplizierter macht.

Als inzwischen etwas veraltetes konkretes Beispiel zeigen wir hier die von Pierce und Hosoya entwickelte Programmiersprache Xduce.

Xduce ist eine funktionale Programmiersprache zur Verarbeitung von XML-Dokumenten. XML-Daten können sowohl als Ein- als auch als Ausgabe einer Funktion erscheinen. Es gibt aber auch noch andere Datentypen wie float, String etc. Für XML-Daten gibt es benutzerdefinierte Typen, die im wesentlichen den XML-Schemas entsprechen.

Dies ist eine Xduce Typdefinition:

```
type Addrbook = addrbook[Person*]
type Person = person[(Name,Tel?,Email* )]
type Name = name[String]
type Tel = tel[String]
type Email = email[String]

type Addrbook2 = addrbook[Person*]
type Person = person[(Name,Tel*,Email* )]
```

```
type Name = name[String]
type Tel = tel[String]
type Email = email[String]
```

Hier ist ein XML-Dokument, das zu ihr passt, also den Typ `Addrbook` hat:

```
<addrbook>
  <person>
    <name>Haruo Hosoya</name>
    <email>hahosoya</email>
    <email>haruo</email>
  </person>
  <person>
    <name>Jerome Vouillon</name>
    <tel>123-456-789</tel>
    <email>vouillon</email>
  </person>
  <person>
    <name>Benjamin Pierce</name>
    <email>pierce</email>
  </person>
</addrbook>
```

Und hier ist dieselbe Typdefinition als DTD (Document Type Definition).

```
<!DOCTYPE Addrbook
 [ <!ELEMENT addrbook( person*) >
   <!ELEMENT person (name, tel?, email*) >
   <!ELEMENT name (\# PCDATA)>
   <!ELEMENT tel (\# PCDATA)>
   <!ELEMENT email (\# PCDATA)>
]>
```

und natürlich können wir auch einen Baumautomaten verwenden, dessen Sprache gerade die XML-Dokumente des Typs Addrbook sind. Ein kleines Problem hierbei ist, dass die Symbole in XML, wie `addrbook`, variable Stelligkeit besitzen. Man kann das dadurch umgehen, dass man alle Symbole nur ein oder zweistellig erlaubt und ggf. neue künstliche Typabkürzungen einführt, vgl. Chomsky Normalform kontextfreier Grammatiken. In Xduce werden XML-Dokumente *intern* in die feststellige Notation umgerechnet und verarbeitet. Alternativ kann man auch die Baumautomaten auf die variable Stelligkeit erweitern.

Die folgende Xduce Funktion übersetzt nun eine Folge von Adresseinträgen in ein Telefonbuch. Die gegebenen Typannotate werden automatisch überprüft.

```
fun mkTelList (val e as (Name,Addr,Tel?)*) : (Name,Tel)* =
  match e with
    name[val n], addr[val a], tel[val t], val rest
      -> name[n], tel[t], mkTelList(rest)
  | name[val n], addr[val a], val rest
```

```
    -> mkTelList(rest)
| ()
    -> ()
```

Startpunkte für weiterführende Lektüre zu diesen Themen sind die WWW-Seiten von Haruo Hosoya (approximative Analyse für beliebige Programme), Thomas Schwentick (exakte Analyse für Transducer), Frank Neven (allgemeines zu XML und Automatentheorie), Michael Schwartzbach (allgemeines zu praktischen Aspekten von XML aus Sicht eines Theoretikers).

Teil V

Unendliche Bäume

16

Automaten auf unendlichen Bäumen

Unter unendlichen Bäumen versteht man hier solche, die zwar endlichen Verzweigungsgrad haben, aber i.a. unendliche viele Knoten haben und somit auch unendlich lange Äste. Es ist klar, dass sich nur das Top-down-Modell auf unendliche Bäume verallgemeinern lässt.

Man kann dann verschiedene Akzeptanzbedingungen studieren, z.B. Büchi, Muller, Rabin, Parität. Wie bei endlichen Baumautomaten hat diese Akzeptanzbedingung jeweils auf jedem Ast des Eingabebaumes zu gelten. Es zeigt sich dann, dass Automaten mit Büchi-Akzeptanzbedingung echt schwächer sind als die (zueinander äquivalenten) Akzeptanzbedingungen Muller und Parität. Allerdings erreichen auch diese ihre volle Stärke nur in der nichtdeterministischen Version, es gibt hier also keine "Safrakonstruktion". Den Nichtdeterminismus braucht man, um im Baum zu suchen; z.B. ist die Sprache aller Bäume, die einen mit a beschrifteten Knoten enthalten (über $\Sigma = \{a, b\}, \sigma(a) = \sigma(b) = 2$) nicht durch einen deterministischen Muller-Baumautomaten zu erkennen.

Die Büchi-erkennbaren Baumsprachen sind nicht unter Komplement abgeschlossen. Ein konkretes Beispiel für eine Baumsprache, die nicht Büchi-erkennbar ist, lautet wie folgt: $\Sigma = \{a, b\}$, $\sigma(a) = \sigma(b) = 2$, $L = \{t \mid$ auf jedem Pfad finden sich nur endlich viele $b\}$. Man beachte, dass $\overline{L}$ Büchi-erkennbar ist.

Man könnte es etwa mit dem Automaten $\mathcal{A} = (\{q_0, q_1\}, \{a, b\}, q_0, \delta, \{q_1\})$ wobei $\delta_a(q_0) = \delta_b(q_0) = Q^2$ und $\delta_b(q_1) = \emptyset$ und $\delta_a(q_1) = \{(q_1, q_1)\}$, versuchen.

Betrachtet man aber den Baum t mit $t(w) = b \iff w \in 1^+0$, so sieht man leicht, dass $\mathcal{A}$ ihn nicht erkennt, obwohl er zu L gehört. Auf dem Pfad 1^ω muss ja irgendwann der Zustand q_1 kommen und dann kann man nicht mehr die b's verarbeiten, die links von ihm abzweigen. Ein rigoroser Beweis, dass L nicht Büchi-erkennbar ist, verläuft ähnlich.

Ein Paritätsbaumautomat (PBA) für L ist dagegen leicht anzugeben, in diesem Fall sogar deterministisch: $\mathcal{A} = (\{q_0, q_1\}, q_0, \delta, \Omega)$, wobei $\delta_a(q) = \{(q_0, q_0)\}$ und $\delta_b = \{(q_1, q_1)\}$ und $\Omega(q_0) = 0, \Omega(q_1) = 1$. Ein Lauf des PBA ist definitionsgemäß akzeptierend, wenn auf jedem Pfad die größte unendlich oft

vorkommende Priorität gerade ist. Vorkommen von a signalisieren wir durch Priorität 0; Vorkommen von b durch die Priorität 1. Diese muss also auf jedem Pfad nach endlicher Zeit aussterben.

Wir werden uns im folgenden auf das Studium der Paritätsbaumautomaten beschränken.

16.1 Unendliche Bäume

Im folgenden sei Σ ein Baumalphabet ohne Blätter, d.h., $\sigma(a) > 0$ für alle $a \in \Sigma$. Das ist eine technische, vereinfachende Bedingung; wenn gewünscht, kann man Blätter durch einstellige Symbole simulieren mit der Maßgabe, dass der Teilbaum unterhalb eines solchen Symbols "nicht gilt".

Definition 47. *Ein unendlicher Baum über Σ ist eine nichtleere, präfixabgeschlossene Menge $T \subseteq \mathbb{N}^*$ und eine Funktion $t : T \to \Sigma$, sodass $\forall w \in T. \forall i < \sigma(t(w)). wi \in T$.*

Beachte, dass T stets unendlich ist.

Ein Pfad in einem unendlichen Baum t ist ein Wort $\pi \in \mathbb{N}^\omega$ derart, dass jedes endliche Präfix u von w in $\mathrm{dom}(t)$ ist. Der Pfad definiert ein unendliches Wort $t(\pi) \in \Sigma^\omega$ durch $t(\pi)_i = t(w_0, \ldots, w_{i-1})$.

16.2 Paritätsbaumautomaten

Definition 48. *Ein Paritätsbaumautomat (PBA) ist ein Tupel $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$, wobei Q eine endliche Menge von Zuständen ist, Σ ein Alphabet wie oben, $q_0 \in Q$, $\delta_a : Q \to 2^{Q^{\sigma(a)}}$ für $a \in \Sigma$, $\Omega : Q \to \mathbb{N}$.*

Ein Lauf des PBA auf einem unendlichen Baum t ist eine Funktion $r : dom(t) \to Q$ sodass $r(\epsilon) = q_0$ und $(r(w0), \ldots, r(w(\sigma(t(w))-1))) \in \delta_{t(w)}(r(w))$ für alle $w \in dom(t)$.

Ein Lauf ist akzeptierend, wenn für alle Pfade π in t gilt, dass die größte Zahl, die in der Folge $\Omega(r(\pi))$ vorkommt, gerade ist.

Die von $\mathcal{A}$ erkannte Sprache besteht wie üblich aus allen Bäumen, zu denen ein akzeptierender Lauf existiert. Analog definiert man Läufe und akzeptierende Läufe von anderen Zuständen als q_0

16.2.1 Latest Appearance Records

Wir müssen nun noch ein Lemma nachholen, welches eigentlich zu den Wortautomaten gehört.

Proposition 33. *Sei $L \subseteq \Sigma^\omega$ eine ω-reguläre Sprache. Dann ist L von einem* deterministischen *Paritäts(wort)automaten erkennbar.*

Beweis. Mithilfe der Safrakonstruktion können wir annehmen, dass L von einem deterministischen Mullerautomaten $\mathcal{A} = (\Sigma, Q, q_0, \delta, \mathcal{F})$ erkannt wird. Wir konstruieren einen deterministischen Paritätsautomaten $\mathcal{P} = (\Sigma, Q', q_0', \delta', \Omega)$ wie folgt.

Die Zustandsmenge Q' ist gegeben als $Q! \times |Q|$. Ein Zustand ist also ein Paar (l, i), wobei l eine wiederholungsfreie Liste (Permutation) der Zustände von $\mathcal{A}$ ist und i eine Zahl im Bereich $0 \dots |Q| - 1$ ist.

Der Anfangszustand ist eine beliebige Liste, an deren Anfang der Anfangszustand des Mullerautomaten steht. Auch die Positionszahl i ist beliebig.

Die Übergangsfunktion δ' ist wie folgt erklärt:

$$\delta'(((q_0, q_1, ..., q_{n-1}), i), a) = ((q_j, q_0, q_1, ..., q_{j-1}, q_{j+1}, ...q_{n-1}), j), \text{ falls } \delta(q_0, a) = q_j$$

NB: q_0 bezeichnet hier nicht den Anfangszustand, sondern den ersten Zustand in der Liste.

Man sucht sich also den Folgezustand, der dem ersten Zustand in der Liste entspricht und schreibt diesen nach vorne. Die anderen Zustände rücken entsprechend nach hinten. Den Positionszeiger setzen wir auf die Position, von der der neue Zustand geholt wurde.

Das bedeutet, dass nach einer Einschwingzeit die Permutationskomponente des Zustands immer die Reihenfolge des letzten Vorkommens aller Zustände visualisiert. Man bezeichnet sie deshalb als Latest Appearance Record (LAR).

Es verbleiben noch die Prioritäten. Wir setzen fest:

$$\Omega((q_0, q_1, ..., q_{n-1}), i) = 2i, \text{ falls } \{q_0, q_1, ..., q_i\} \in \mathcal{F}$$
$$\Omega((q_0, q_1, ..., q_{n-1}), i) = 2i + 1, \text{ falls } \{q_0, q_1, ..., q_i\} \in \mathcal{F}$$

Es verbleibt zu zeigen, dass dieser Automat dieselben Wörter, wie $\mathcal{A}$ erkennt. Sei $w \in L(\mathcal{A})$ und $U = \inf(w) \in \mathcal{F}$. Beim Abarbeiten des Wortes w werden sich ab einem gewissen Zeitpunkt die Zustände aus U am Anfang des LAR befinden und nur noch unter diesen werden Zustände nach vorne geholt, ausserdem wird jeder immer wieder nach vorne geholt. Das bedeutet, dass genau die Positionen $\{0 \dots, i\}$ immer wieder auftreten; die größeren sterben aus. Die größte Position, die immer wieder auftritt, ist i und bei ihrem Auftreten ist $\{q_0, \dots, q_i\} = U \in \mathcal{F}$, somit ist die korrespondierende Priorität gerade und der Paritätsautomat akzeptiert.

Sei jetzt w nicht in $L(\mathcal{A})$. Dann ist $\inf(w) \notin \mathcal{F}$ und deshalb mit demselben Argument die größte unendlich oft auftretende Priorität ungerade. Also ist w auch nicht in $L(\mathcal{P})$.

Wir bemerken, dass man mit dem LAR auch das DJW Spiel lösen kann. Der Zahlenspieler führt einen LAR mit, in den er immer die gespielten Buchstaben einträgt; er antwortet jeweils mit dem aktuellen Positionszeiger.

Proposition 34. *Die Klasse der PBA-erkennbaren Baumsprachen ist unter Vereinigung und Projektion entlang stelligkeitserhaltender Funktionen $\Sigma' \to \Sigma$ zwischen Baumalphabeten abgeschlossen.*

Beweis. Wie im endlichen Falle unter Verwendung von Nichtdeterminismus.

Proposition 35. *Sei Σ Baumalphabet mit maximaler Stelligkeit n und $D = \{0, \ldots, n-1\}$. Sei L eine ω-reguläre Wortsprache über $\Sigma \times D$. Die assoziierte Baumsprache L^t besteht aus allen Bäumen t, sodass jeder Pfad in t geschrieben als Wort über $\Sigma \times D$ in L ist. Die Sprache L^t ist PBA-erkennbare Baumsprache.*

Beweis. Man bestimt mit Satz 33 einen deterministischen Paritätsautomaten $\mathcal{P} = (\Sigma \times D, Q, q_0, \delta, \Omega)$. Der folgende PBA erkennt offensichtlich L^t:

$$(\Sigma, Q, q_0, \delta', \Omega)$$

wobei

$$\delta'_a(q) = (\delta(q, (a, 0)), \delta(q, (a, 1)), \ldots, \delta(q, (a, n-1)))$$

Der entstehende PBA ist hier also sogar deterministisch. Es ist allerdings leider nicht möglich, dieselbe Konstruktion ausgehend von einem nichtdeterministischen Paritätsautomaten durchzuführen, selbst dann nicht, wenn man in Kauf nimmt, dass der entstehende PBA nichtdeterministisch wird. Der Grund ist, dass dann der gewählte Pfad von den nichtdeterministischen Entscheidungen des Wortautomaten abhängig gemacht werden kann.

Für ein konkretes Gegenbeispiel betrachte man $\Sigma = \{a, b\}, \sigma(a) = \sigma(b) = 2$, $L = (b \times D)^*(a \times D)^\omega$. Hier ist L^t gerade die in der Einführung erwähnte nicht-Büchi-erkennbare Baumsprache. Ein nichtdeterministischer Büchiautomat würde hier einfach raten, wann die a's vorbei sind. Lässt man ihn auf allen Pfaden laufen, dann müsste er aber in der Lage sein, festzustellen, wann die a's auf allen Folgepfaden vorbei sind. Das muss aber möglicherweise nie passieren!

17

Paritätsspiele

17.1 Definition

Wir wollen für die Komplementierung von PBA wiederum ein Spiel zwischen dem Automaten und dem Pfadfinder einsetzen. Dafür ist es nötig, die dabei auftretenden Spiele zunächst separat zu studieren.

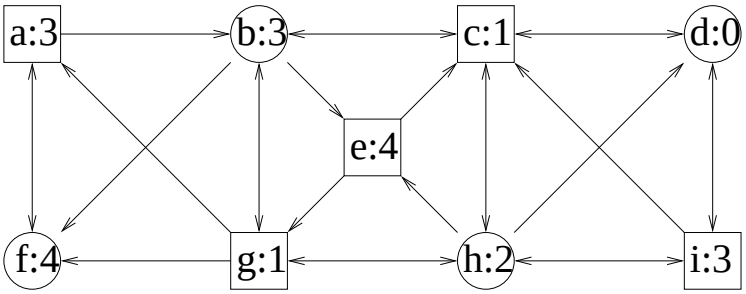


Abb. 17.1. Ein Paritätsspiel

Definition 49 (Paritätsspiel). *Ein Paritätsspiel ist ein Tupel* $G = (Pos_A, Pos_P, \delta, \Omega)$, *wobei* Pos_A *und* Pos_P *zwei disjunkte, nicht notwendigerweise endliche Mengen sind, die Positionen des Spiels. Man schreibt* $Pos := Pos_A \cup Pos_P$.

Es wird zusätzlich gefordert, dass $\forall x \in Pos.\exists y \in Pos.\delta(x, y)$. *Die Relation* $\delta \subseteq Pos \times Pos$ *modelliert die möglichen Züge und* $\Omega : Pos \to \{0, \ldots, n\}$ *weist den Positionen Prioritäten aus einer stets endlichen Teilmenge von* $\mathbb{N}$ *zu.*

Das Spiel wird von einer gegebenen Startposition $p \in Pos$ *so gespielt, dass eine Figur auf* p *gesetzt wird und dann von den Spielern* A *und* P *entlang der Kanten* δ *verschoben wird. Ist die aktuelle Position in* Pos_A, *so ist Spieler* A *am Zug, ansonsten* P. *Aufgrund der Forderung* $\forall x \in Pos.\exists y \in Pos.\delta(x, y)$ *gibt es keine Sackgassen.*

Wir nehmen stets an, dass es keine Sackgassen gibt, d.h. Kann ein Spieler nicht mehr ziehen, so hat er verloren, ansonsten wird das Spiel ad infinitum fortgesetzt. Die resultierende unendliche Partie $p = p_0, p_1, p_2, p_3, \ldots$ *wird von Spieler A gewonnen, falls die größte Zahl, die in der Folge* $\Omega(p_0), \Omega(p_1), \ldots$ *unendlich oft auftritt, gerade ist. Ansonsten, also wenn sie ungerade ist, gewinnt P.*

Eine Gewinnposition für A ist eine Position, von der aus A das Spiel bei beliebigem Spiel von P gewinnen kann. Die Menge der Gewinnpositionen für A wird mit W_A *bezeichnet. Analog definiert man* W_P*.*

Eine positionale Gewinnstrategie für A ist eine Funktion $s : W_A \cap Pos_A \to W_A$*, die für jede Gewinnposition p für A, in der auch A am Zuge ist, einen legalen Zug* $s(p)$*, also* $(p, s(p)) \in \delta$ *ausweist, sodass A von jeder beliebigen Gewinnposition aus tatsächlich gewinnt, wenn er nur die von der Strategie s empfohlenen Züge durchführt.*

In Figure 17.1 ist ein Paritätsspiel grafisch dargestellt. Die Positionen, in denen Spieler A am Zug ist, sind hier als Kreise dargestellt; die Positionen von Spieler P als Kästchen, also $\text{Pos}_A = \{b, d, f, h\}$ und $\text{Pos}_P = \{a, c, g, e, i\}$. Die Prioritäten sind gegeben durch $\Omega(a) = 3, \Omega(b) = 3, \Omega(c) = 1, \ldots$. Die erlaubten Übergänge sind $\delta = \{(a, b), (a, f), (f, a), \ldots\}$.

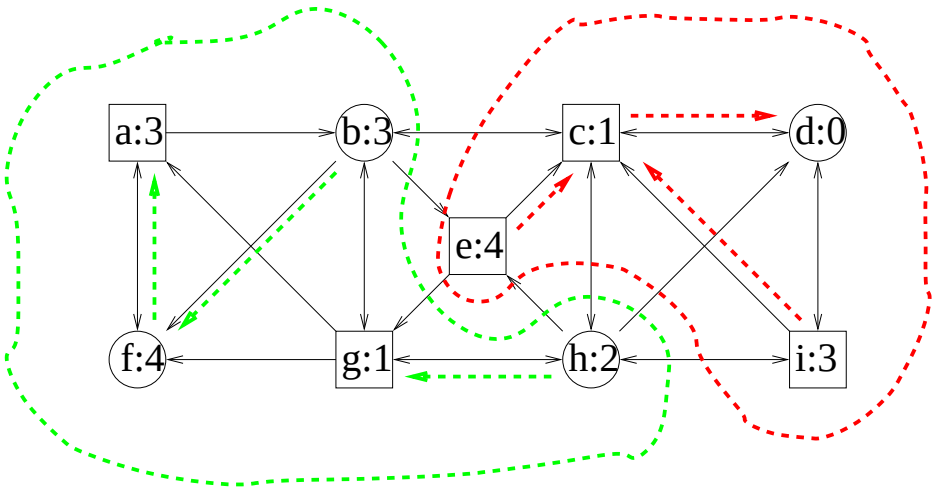


Abb. 17.2. Paritätsspiel mit Gewinnbereichen und positionalen Gewinnstrategien

In Figur 17.2 sind die Gewinnbereiche W_A und W_P nebst zugehörigen positionalen Gewinnstrategien eingezeichnet. Es ist: $W_A = \{a, b, f, g, h\}$ und $W_P = \{c, d, e, i\}$.

Theorem 38 (Positionale Determiniertheit von Paritätsspielen). *Sei $G = (Pos_A, Pos_P, \delta, \Omega)$ ein Paritätsspiel. Es gilt $W_A \cup W_P = Pos$ und es existieren positionale Gewinnstrategien für A und für P.*

Beweis. Die Tatsache, dass von jeder Position entweder A oder P das Spiel gewinnen kann, wird als Determiniertheit bezeichnet und folgt aus einem allgemeinen Satz (Martins Determiniertheitssatz). Die Determiniertheit ist nicht offensichtlich und in der Tat folgt aus dem Auswahlaxiom die Existenz unendlicher Spiele, die nicht determiniert sind.

Die positionale Determiniertheit ist eine Besonderheit der Paritätsspiele und gilt z.B. nicht für die in offensichtlicher Weise definierten Mullerspiele.

Wir beweisen sie nunmehr direkt ohne Rückgriff auf den Determiniertheitssatz durch Induktion über n, die höchste vorkommende Priorität.

Ist $n = 0$, so gewinnt A bei beliebigem Spiel positional. Sei also $n > 0$. Indem wir ggf. die Rollen von A und P vertauschen, dürfen wir o.B.d.A. voraussetzen, dass n gerade ist.

Wir zeigen zunächst:

Lemma 37. *Sei U eine Menge von Positionen, sodass A von jeder Position $p \in U$ eine positionale Gewinnstrategie s_p besitzt. Dann gibt es auch eine einzige Gewinnstrategie s, deren Befolgung Gewinn für A von allen Positionen $p \in U$ sichert.*

Beweis. Wir nummerieren die Positionen durch: $p_0, p_1, p_2, \ldots$ (falls überabzählbar viele, so verwende man Ordinale). Jetzt betrachten wir s_{p_0}. Diese Strategie sichert den Gewinn für A von p_0 aus, aber auch von allen Positionen aus, die bei Befolgung von s_0 von p_0 aus erreichbar sind bei beliebigem Spiel von P, denn was auch immer P macht, muss ja die Gewinnstrategie funktionieren. Natürlich sind das nicht alle Positionen; aber es gibt ja noch s_{p_1}. Dies sichert Gewinn auf p_1 und allen Positionen, die von p_1 erreichbar sind. Somit können wir die universelle Gewinnstrategie s definieren durch $s(p) = s_{p_i}(p)$, wobei i das kleinste i ist, sodass p von p_i aus bei Befolgung von s_{p_i} erreicht werden kann.

Sei jetzt U die Menge derjenigen Positionen, von denen aus P eine positionale Gewinnstrategie besitzt. Nach Lemma 37 gibt es dann eine einzige Gewinnstrategie, die P's Gewinn auf allen Positionen in U sichert. In Figur 17.3 ist die Menge U durch eine gepunktete Linie eingeschlossen. Wir möchten zeigen, dass von allen Positionen in $\mathrm{Pos} \setminus U$ der Spieler A eine positionale Gewinnstrategie hat.

Wir betrachten das Teilspiel auf den Positionen außerhalb von U, d.h. alle Spielzüge, die aus $\mathrm{Pos} \setminus U$ herausführen, werden verboten. Wir beachten, dass es keinen so herausführenden Spielzug für P geben kann, da ja sonst die entsprechende Position zu U gegeben hätte werden müssen. Außerdem wird Spieler A durch diese Wegnahme nicht in eine Sackgasse geführt, denn gäbe es eine Position $p \in \mathrm{Pos}_A \setminus U$ derart, dass jeder Zug von p nach U hineinführt, so hätte man diese Position p auch der Menge U hinzurechnen müssen.

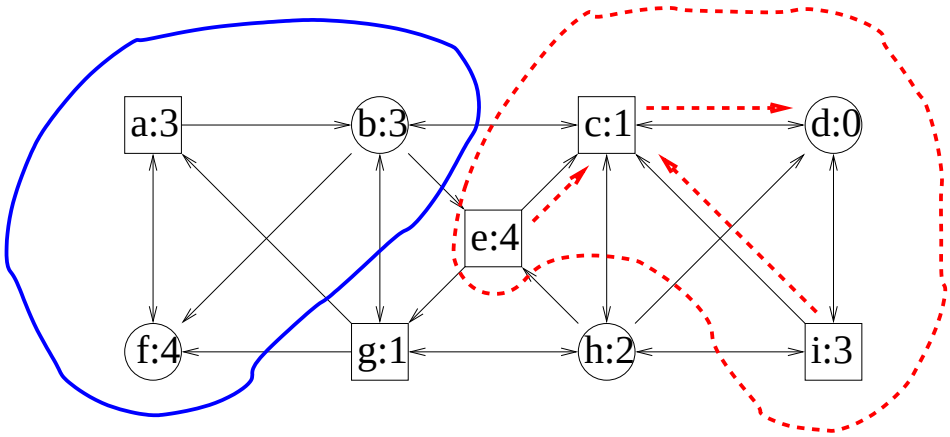


Abb. 17.3. Paritätsspiel mit Gewinnbereich und Attraktor

Kommt in Pos$\setminus U$ die höchste Priorität nicht vor, so erlaubt uns die Induktionshypothese die Aufteilung von Pos $\setminus U$ in zwei Bereiche W_A, W_P, wobei A das Teilspiel in W_A gewinnt und P es in W_P gewinnt (jeweils positional). Nachdem P den Bereich Pos$\setminus U$ nicht verlassen kann, gewinnt A auf W_A sogar das gesamte Spiel. Andererseits gewinnt P auf W_P auch das gesamte Spiel, denn sollte A sich in den Bereich U absetzen, so würde P natürlich sofort seine dortige Strategie zum Einsatz bringen. Aufgrund der Annahme an U bedeutet dies aber, dass W_P leer ist und somit A auf ganz Pos $\setminus U$ eine positionale Gewinnstrategie besitzt, w.z.b.w.

Es bleibt der Fall zu betrachten, wo die höchste Priorität n (o.B.d.A. n gerade) in Pos $\setminus U$ vorkommt. Sei N die Menge der Positionen in Pos $\setminus U$ mit Priorität n. Sei $Attr_A(N)$ die Menge derjenigen Positionen, von denen aus A einen Besuch von N erzwingen kann. Man nennt diese Menge den Attraktor von N. Man beachte, dass es keine A-Züge nach $Attr_A(N)$ hin gibt und keine P-Züge aus $Attr_A(N) \setminus N$ heraus. Es gilt stets $N \subseteq Attr_A(N)$. Im Beispiel ist $N = \{f\}$ und $Attr_A(N) = \{a, b, f\}$ (in Figur 17.1c) durchgezogen eingeschlossen).

Wir betrachten nunmehr das Teilspiel auf $Z = \text{Pos} \setminus U \setminus Attr(N)$. Im Beispiel ist $Z = \{g, h\}$. Wieder handelt es sich hierbei tatsächlich um ein Spiel, denn würden von einer gewissen Position alle Züge von P nach $Attr(N)$ hineinführen, so hätte man diese Position dem Attraktor zuschlagen müssen. Wir teilen es nach Induktionshypothese auf in Gewinnbereiche W_A und W_P. Auf W_P hat P sogar für das gesamte Spiel eine positionale Gewinnstrategie, denn in den Attraktor kann A nicht entkommen und ein Entkommensversuch in den Bereich U führt zum Verlust. Daher ist aber wie vorher W_P leer.

Es bleibt zu zeigen, dass sowohl auf dem Attraktor, als auch auf W_A das Gesamtspiel von A positional gewonnen wird. Das geht wie folgt: Auf W_A spielt A gemäß seiner positionalen Gewinnstrategie für das Teilspiel; auf dem

```
SolveGame(G) =
    let (Pos_A, Pos_P, δ, Ω) := G
    n := max{Ω(v) | v ∈ Pos}

    if  n = 0 then return (W_A := Pos, W_P := ∅)

    if n gerade then σ := A else σ := P

    N := {v ∈ Pos | Ω(v) = n}
    N' := Attr_σ(N)

    (W'_σ, W'_σ̄) := SolveGame(G \ N')
    if W'_σ̄ = ∅ then return (W_σ := W'_σ ∪ N', W_σ̄ := ∅)

    N'' := Attr_σ̄(W'_σ̄)
    (W''_σ, W''_σ̄) := SolveGame(G \ N'')
    return (W_σ := W''_σ, W_σ̄ := N'' ∪ W''_σ̄)
```

Abb. 17.4. Ein rekursiver Algorithmus zum Lösen von Paritätsspielen.

Attraktor hingegen erzwingt A einen Besuch von N. Diese Strategie stellt insbesondere sicher, dass der Spielverlauf $\text{Pos} \setminus U$ nie verlässt. Führt der Spielverlauf immer wieder nach N hinein, so gewinnt A nach Definition, ansonsten gemäß der Induktionshypothese.

Die positionale Determiniertheit der Paritätsspiele ist somit erwiesen.

17.1.1 Algorithmische Behandlung der Paritätsspiele

Ist die Menge der Positionen eines Paritätsspiel endlich, so kann man nach Algorithmen fragen, die entscheiden, ob eine bestimmte Position in W_A oder in W_P ist. Ein einfacher Algorithmus von McNaughton und Zielonka ist in Abbildung 17.4 dargestellt und funktioniert in etwa wie folgt:

Sei N die Menge der Positionen mit der höchsten (o.B.d.A. geraden Priorität). Berechne iterativ den Attraktor $Attr_A(N)$, also die Menge der Positionen von denen aus A das Spiel nach N zwingen kann. Berechne nunmehr rekursiv (die Zahl der Prioritäten ist ja kleiner geworden) die Menge der Positionen X, von denen aus P das Teilspiel $\text{Pos} \setminus Attr_A(N)$ gewinnt. Im Beispiel besteht der Attraktor aus $\{a, b, e, f, g, h\}$ und die Gewinnmenge X ist $\{c, d, i\}$. Siehe auch Figur 17.5. Ist diese leer, so gewinnt A überall. Ansonsten gewinnt P auch das gesamte Spiel von allen Positionen in X, da es ja keine Züge für A in den Attraktor gibt. Sei nun Y die Menge der Positionen im Gesamtspiel, von denen aus P den Spielverlauf nach X zwingen kann (auch ein Attraktor). Im Beispiel ist $Y = X \cup \{e\}$. Es ist klar, dass P auch auf allen Positionen in Y gewinnt. Nun bestimmen wir rekursiv W'_A und W'_P für das Teilspiel $\text{Pos} \setminus Y$ (weniger Positionen!). Im Beispiel besteht W'_A aus ganz $\text{Pos} \setminus Y$ und $W'_P = \emptyset$.

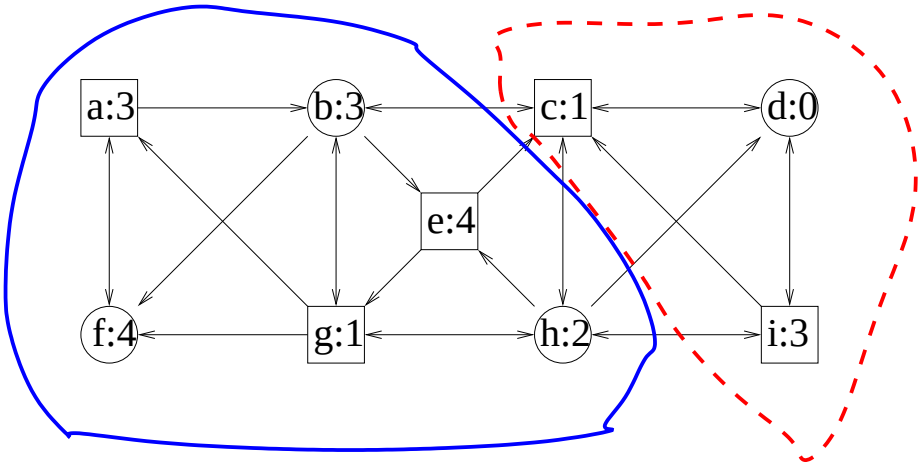


Abb. 17.5. Paritätsspiel mit Attraktor und Gewinnbereich für Teilspiel

Wir behaupten, dass $W_A = W'_A$ und $W_P = Y \cup W'_P$. In der Tat gewinnt A von W'_A aus, da ja P nicht in den Attraktor Y entkommen kann. Auf der anderen Seite gewinnt P von W'_P aus, da ein mögliches Entkommen von A in die Menge Y sofort mit der dort vorhandenen Gewinnstrategie beantwortet wird.

Abb. 17.4 zeigt diesen Algorithmus in Pseudocode. Dabei ist $G \setminus N$ für ein Spiel G und eine Menge N von Positionen darin das Spiel, welches aus G entsteht, wenn alle Positionen in N mit allen ein- und ausgehenden Kanten entfernt werden.

Leider werden in diesem Algorithmus zwei rekursive Aufrufe getätigt, sodass die Komplexität exponentiell ist. Es ist ein offenes Problem, ob sich Paritätsspiele in polynomialer Zeit entscheiden lassen.

Theorem 39. *Das folgende Problem liegt sowohl in NP, als auch in co-NP. Gegeben ein Paritätsspiel $G = (Pos_A, Pos_P, \delta, \Omega)$ und ein $v \in Pos$. Entscheide, ob $v \in W_A$ ist.*

Beweis. Die Inklusion in NP wird wie folgt gezeigt. Rate eine positionale Strategie für Spieler A, die den Knoten v enthält. Das ist offensichtlich in Zeit $O(|\delta|)$ möglich. Betrachte nun den von dieser Strategie s induzierten Graphen G'. Dieser enthält nur noch die Kanten

$$\{(w, s(w)) \mid w \in Pos_A\} \cup (\delta \cap Pos_P \times Pos)$$

Auch diese Konstruktion ist in Zeit $O(|\delta|)$ möglich.

Nun gilt: s ist eine Gewinnstrategie gdw. die höchste Priorität auf jedem in G' vorkommenden Zykel gerade ist. Diese Bedingung ist aber ebenfalls in polynomieller Zeit prüfbar. Die Inklusion in co-NP folgt aus der Determiniertheit. Beachte: Ein Problem ist in co-NP, falls sein Komplement in NP ist. D.h.

für die Inklusion in co-NP muss man einen NP-Algorithmus angeben, der entscheidet, ob $v \notin W_A$ ist. Dies ist zuerst nicht offensichtlich, denn dazu müsste man ja eigentlich jede Strategie testen. Es gilt aber: $v \notin W_A$ gdw. $v \in W_P$. Damit lässt sich obiger NP-Algorithmus genauso verwenden, indem lediglich nach einer Strategie für Spieler P gesucht wird.

17.2 Komplementierung der Paritätsbaumautomaten

Die schwierigste logische Abschlusseigenschaft ist der Abschluss der PBA unter Komplement. Wie schon bei endlichen TDBA fassen wir die Akzeptanz durch den Automaten als Zweipersonenspiel auf zwischen Spieler A (Automat) und P (Pfadfinder). Sei also ein PBA $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ und ein Baum t gegeben. Das assoziierte Paritätsspiel $G(\mathcal{A}, t)$ ist wie folgt erklärt:

1. Positionen des Spiels sind Paare (w, q) und Paare $(w, \mathbf{q})$, wobei $w \in D^*$ eine Position im Baum ist ($D = \{0, \ldots, n-1\}$, q ein Zustand und $\mathbf{q} \in Q^{<n}$ ein Tupel von Zuständen ist (n ist die maximale Stelligkeit des Alphabets.) Bei Positionen der Form (w, q) ist A am Zug; bei Positionen der Form $(w, \mathbf{q})$ ist P am Zug.
2. In der Position (w, q) wählt A ein Tupel $\mathbf{q} \in \delta_a(q)$, wobei $a = t(w)$ und erreicht die Position $(w, \mathbf{q})$).
3. In der Position $(w, \mathbf{q})$ wählt P eine Richtung $i < \sigma(t(w))$ und es wird die Position (wi, q_i), wobei q_i die i-te Komponente von $\mathbf{q}$ ist.
4. Die Priorisierung ist gegeben durch $\Omega(w, q) = \Omega(q)$ und $\Omega(w, \mathbf{q}) = 0$.

Lemma 38. *Spieler A gewinnt das Spiel $G(\mathcal{A}, t)$ von der Position (ϵ, q_0) aus, genau dann, wenn $t \in L(\mathcal{A})$.*

Beweis. Übung.

Mit der positionalen Determiniertheit der Paritätsspiele folgt hieraus, dass $t \notin L(\mathcal{A})$ genau dann, wenn P das Spiel von (ϵ, q_0) aus positional gewinnt. Dies nutzen wir nun, um einen PBA für das Komplement von $L(\mathcal{A})$ zu konstruieren.

Proposition 36 (Komplementierung von PBA). *Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ PBA. Man kann effektiv einen PBA $\mathcal{A}'$ konstruieren, mit $L(\mathcal{A}') = \overline{L}(\mathcal{A})$.*

Beweis. Wenn $t \notin L(\mathcal{A})$ ist, so besitzt P im Spiel $G(\mathcal{A}, t)$ eine positionale Gewinnstrategie. Diese hat die Form einer Funktion

$$s : \mathrm{dom}(t) \times \Sigma \times Q^n \to D$$

wobei n die maximale Stelligkeit von Σ ist und $D = \{0, \ldots, n-1\}$.

Wir setzen $S = \Sigma \times Q^n \to D$, sodass die Funktion s als mit S beschrifteter Baum aufgefasst werden kann.

Sei $\Delta = \Sigma \times S, p_1(a, s) = a, p_2(a, s) = s$.

Es ist also $t \notin L(\mathcal{A})$ genau dann, wenn ein Baum t' (über Δ) existiert mit $p_1(t') = t$ und so, dass $s := p_2(t')$ eine Gewinnstrategie in $G(\mathcal{A}, t)$ für P definiert.

Sei L' die Menge aller solchen Bäume t'.

Ein Baum t' ist in L' genau dann, wenn für alle Zugfolgen $z = \mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \ldots$ für A folgendes gilt: Die durch w und $s = p_2(t')$ definierte Partie ist entweder illegal und der Fehler dafür liegt bei w, also A, oder sie ist ein Gewinn für P.

Da jede Partie einen Pfad durch t definiert, können wir die Partien auch nach diesen Pfaden aufteilen:

Ein Baum t' ist in L' genau dann, wenn für alle Pfade π und alle Zugfolgen $z = \mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \ldots$ für A folgendes gilt:

Die durch w und $s = p_2(t')$ definierte Partie ist entweder illegal und der Fehler dafür liegt bei w, also A und falls die Partie den Pfad π definiert, so ist sie ein Gewinn für P. Dies aber lässt sich als MSO definierbare und damit ω-reguläre Eigenschaft F von Pfaden in t' formalisieren; somit ist $L' = F^t$ und nach Satz 35 durch einen PBA erkennbar. Den gewünschten Komplementautomaten erhält man dann durch nichtdeterministisches Raten der Strategiebeschriftung gemäß Satz 34.

17.3 Leerheitstest

Definition 50. *Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ ein PBA, und $m := \max\{\sigma(a) \mid a \in \Sigma\}$. Sei $\Sigma^- := \{a\}$ mit $\sigma(a) := m$.*

Definiere einen PBA $\mathcal{A}^- := (Q, \Sigma^-, q_0, \delta^-, \Omega)$ durch

$$\delta^-(q,a) \;:=\; \{(q_1, \ldots, \underbrace{q_i, \ldots, q_i}_{m-i+1 mal}) \mid \exists b \in \Sigma, \sigma(b) = i \text{ und } (q_1, \ldots, q_i) \in \delta(q,b)\}$$

Beachte: $\mathcal{A}^-$ erkennt intuitiv all diejenigen Bäume, die $\mathcal{A}$ erkennen würde, ohne dabei auf das Alphabet zu achten.

Lemma 39. *Sei $\mathcal{A}$ ein PBA über Σ. Es gilt $L(\mathcal{A}) = \emptyset$ gdw. $L(\mathcal{A}^-) = \emptyset$.*

Beweis. "$\Leftarrow$" Angenommen, $t \in L(\mathcal{A})$. Sei t' derjenige Baum, der aus t entsteht, indem jede Beschriftung durch a ersetzt wird, und der jeweils rechteste Sohn eines jeden Knoten so oft kopiert wird, bis jeder Knoten die Stelligkeit $\max\{\sigma(b) \mid b \in \Sigma\}$ hat. Man sieht leicht, dass $t' \in L(\mathcal{A}^-)$ gilt.

"$\Rightarrow$" Sei $m := \max\{\sigma(b) \mid b \in \Sigma\}$. Angenommen, $t' \in L(\mathcal{A}^-)$. Dann existiert ein erfolgreicher Lauf r von $\mathcal{A}^-$ auf t'. Sei $w \in dom(t')$ und $r(w) = q$ für ein $q \in Q$. Also existieren $q_1, \ldots, q_m$, so dass $r(w(i-1)) = q_i$ für $i = 1, \ldots, m$ gilt. Laut Definition von δ^- existiert dann auch ein $b \in \Sigma$, so dass $(q_1, \ldots, q_{\sigma(b)}) \in \delta(q,b)$ ist. In dem die $m - \sigma(b)$ rechten Söhne von jedem solchen w eliminiert werden, erhält man einen Baum t, auf dem r einen Lauf des Automaten $\mathcal{A}$ bildet. Da nur Pfade verworfen wurden, gilt die Paritätsbedingung weiterhin auf allen Pfaden, womit $t \in L(\mathcal{A})$ gezeigt ist. $\square$

Theorem 40. *Das Leerheitsproblem für PBAs mit n Zuständen und p Prioritäten über einem Alphabet Σ kann in Zeit $n^{O(p)}$ entschieden werden.*

Beweis. Sei $m := \max\{\sigma(b) \mid b \in \Sigma\}$ und $\mathcal{A}$ ein PBA mit n Zuständen und p Prioritäten. Offensichtlich ist dann auch $\mathcal{A}^-$ ein PBA mit denselben Größenbeschränkungen. Der Leerheitstest auf $\mathcal{A}$ reduziert sich laut Lemma 39 auf den Leerheitstest auf $\mathcal{A}^-$. Es gilt jedoch: $L(\mathcal{A}^-) \neq \emptyset$ gdw. $t_a \in L(\mathcal{A}^-)$, wobei t_a der Baum ist, dessen Knoten alle Beschriftung a und denselben Verzweigungsgrad m haben.

Laut Lemma 38 gilt $t_a \in L(\mathcal{A}^-)$ gdw. Spieler A das Paritätsspiel $G(\mathcal{A}^-, t_a)$ gewinnt. Da t_a fest ist, kann dieses Spiel als endlicher Graph mit $n + n^m$ Knoten und $O(n^{2m})$ vielen Kanten repräsentiert werden. Außerdem gibt es in $G(\mathcal{A}^-, t_a)$ nur p viele Prioritäten. Der in Abschnitt 17.1 skizzierte rekursive Algorithmus löst endliche Paritätsspiele in $O(e \cdot v^d)$ vielen Schritten, wobei e die Anzahl der Kanten, v die Anzahl der Knoten und d die maximale Priotität ist. Dies liefert einen Leerheitstest für PBA, der in Zeit $n^{O(p)}$ läuft, wenn die Stelligkeit des Alphabets fest ist. □

Definition 51. *Sei Σ ein Alphabet mit Stelligkeiten $\{1, \ldots, m\}$. Ein Baum über Σ heißt* endlich repräsentierbar*, falls er nur endlich viele nicht-isomorphe Teilbäume hat.*

Beachte, dass die endlich repräsentierbaren ω-Bäume genau diejenigen sind, die sich durch Gleichungssysteme der Form

$$\begin{aligned} t_1 &= a_1(t_{i_{1,1}}, \ldots, t_{i_{1,\sigma(a_1)}}) \\ &\vdots \\ t_n &= a_n(t_{i_{n,1}}, \ldots, t_{i_{n,\sigma(a_n)}}) \end{aligned}$$

darstellen lassen.

Korollar 10. *Sei $\mathcal{A}$ ein PBA. Es gilt $L(\mathcal{A}) \neq \emptyset$ gdw. es einen endlich repräsentierbaren Baum t gibt, so dass $t \in L(\mathcal{A})$ gilt.*

Beweis. Die Richtung "$\Leftarrow$" ist offensichtlich. Die Richtung "$\Rightarrow$" folgt aus Satz 40. Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$. Angenommen $L(\mathcal{A}) \neq \emptyset$. Dann hat Spieler A eine Gewinnstrategie s für das Spiel $G(\mathcal{A}^-, t_a)$. Da es sich um ein Paritätsspiel handelt, kann diese sogar als positional angenommen werden. Beachte: $s : Q \to Q^m$, wobei m die maximale Stelligkeit des Baumalphabets ist. Wir benutzen nun s, um einen endlich repräsentierbaren Baum zu konstruieren. Dieser ergibt sich als maximale Lösung für t_{q_0} im Gleichungssystem, welches für jedes $q \in Q$ die folgende Gleichung enthält.

$$t_q = b(t_{q_{i_1}}, \ldots, t_{q_{i_n}})$$

falls $\sigma(b) = n$ und $s(q) = (q_{i_1}, \ldots, q_{i_n}) \in \delta(q, b)$.

Zuerst zeigen wir, dass dieses Gleichungssystem existiert. Nach Voraussetzung ist $s(q_0)$ definiert. Also existiert ein $b \in \Sigma$ mit $s(q) = (q_{i_1}, \ldots, q_{i_n}) \in \delta(q_0, b)$. Da im Spiel $G(\mathcal{A}^-, t_a)$ nun Spieler P mit der Wahl eines q_{i_j} antwortet, muss $s(q_{i_j})$ wiederum für alle $j = 1, \ldots, n$ definiert sein. Dieser Prozess terminiert, da es nur endlich viele q gibt.

Es bleibt noch zu zeigen, dass $t_{q_0} \in L(\mathcal{A})$ gilt. Man sieht leicht, dass es einen Lauf von $\mathcal{A}$ auf t_{q_0} gibt – dieser beschriftet die Wurzel eines Unterbaums t_q mit dem Zustand q. Dass dieser Lauf erfolgreich ist, folgt aus der Tatsache, dass s eine Gewinnstrategie für Spieler A in $G(\mathcal{A}^-, t_a)$ ist. Beachte, dass jeder Pfad in diesem Lauf einer Partie in $G(\mathcal{A}^-, t_a)$ entspricht, in der Spieler A gemäß der Strategie s gewählt hat. Also ist die größte, unendlich auftretende Priorität in dieser Partie gerade. Die Prioritäten im Spiel sind jedoch die der Automatenzustände. Also ist auch die größte, unendlich oft auftretende Priorität auf jedem Pfad des Laufs gerade. □

18

MSO auf Bäumen

Wir betrachten jetzt eine zweitstufige Logik MSOT, die sich zu Baumautomaten so verhält, wie die MSO zu Büchiautomaten. So wie die MSO hat auch MSOT erst- und zweitstufige Variablen; die erststufigen Variablen rangieren hier aber über Positionen in einem unendlichen Baum, also *endlichen Wörtern* über einem Alphabet Δ von *Richtungen*. Oft ist $\Delta = \{1, 2\}$; man hat es dann unendlichen binären Bäumen zu tun.

Die zweitstufigen Variablen rangieren über Mengen von solchen Positionen, also *Sprachen* über Δ.

Definition 52. *Seien zwei abzählbar unendliche Mengen von erststufigen Variablen $V_1 = \{x, y, \ldots\}$ und zweitstufigen Variablen $V_2 = \{X, Y, \ldots\}$ gegeben. Formeln der* monadischen Logik zweiter Stufe auf Bäumen*, kurz MSOT, über V_1, V_2 sind gegeben durch folgende Grammatik.*

$$\varphi \ ::= x{=}y \mid x{=}\varepsilon \mid x{=}ya \mid X(x) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \exists x.\varphi \mid \exists X.\varphi$$

wobei $a \in \Delta$ und $x, y \in V_1$ und $X \in V_2$.

Ein Belegung $\mathcal{I}$ *ordnet den erststufigen Variablen Wörter über Δ und den zweitstufigen Variablen Sprachen über Δ zu. Die Semantik der MSOT ist definiert als die folgende Relation zwischen Belegungen und Formeln:*

$$\begin{aligned}
I \models x{=}y \quad & \textit{gdw.} \quad I(x) = I(y) \\
I \models x{=}\varepsilon \quad & \textit{gdw.} \quad I(x) = \varepsilon \\
I \models x{=}ya \quad & \textit{gdw.} \quad I(x) = I(y)a \\
I \models X(x) \quad & \textit{gdw.} \quad I(x) \in I(X) \\
I \models \varphi \vee \psi \quad & \textit{gdw.} \quad I \models \varphi \textit{ oder } I \models \psi \\
I \models \neg\varphi \quad & \textit{gdw.} \quad I \not\models \varphi \\
I \models \exists x.\varphi \quad & \textit{gdw.} \quad \textit{es gibt ein } w \in \Delta^*\textit{, sodass } I[x \mapsto w] \models \varphi \\
I \models \exists X.\varphi \quad & \textit{gdw.} \quad \textit{es gibt} L \subseteq \Delta^*\textit{, sodass } I[X \mapsto L] \models \varphi
\end{aligned}$$

Hat man eine endliche Menge $\mathcal{V}$ von Variablen (erst- und zweitstufig), so kann man Belegungen, die sich höchstens auf diesen Variablen von den Defaultwerten ε und $\emptyset$ unterscheiden, durch Bäume über dem Baumalphabet $\Sigma := \mathcal{P}(\mathcal{V})$ und $\sigma(x) = |\Delta|$ kodieren.

Hierzu fixiert man eine Ordnung auf Δ, o.B.d.A. setzen wir voraus $\Delta = \{1, 2, \ldots, n\}$, also $\sigma(x) = n$ für alle $x \in \mathcal{V}$.

Eine Belegung $\mathcal{I}$ definiert dann den Baum

$$t_{\mathcal{I}}(w) = \{X \mid X \in \mathcal{V} \wedge w \in \mathcal{I}(X)\} \cup \{x \mid x \in \mathcal{V} \wedge w = \mathcal{I}(X)\}$$

Durch Induktion über den Formelaufbau kann man nun zu jeder Formel ϕ einen PBA $\mathcal{A}_\phi$ konstruieren, sodass gilt:

$$t_{\mathcal{I}} \in L(\mathcal{A}_\phi) \iff \mathcal{I} \models \mathcal{A}_\phi$$

Existentielle Quantifikation und Disjunktion werden mit Nichtdeterminismus (Satz 34) behandelt, für die Negation verwenden wir den Komplementabschluss; die atomaren Formeln werden durch explizite Automatenkonstruktion codiert.

Mit Hilfe von erhalten wir dann:

Korollar 11. *Es ist entscheidbar, ob eine vorgelegte MSOT-Formel erfüllbar ist.*