Institut für Formale Methoden der Informatik

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Iterierte Substitutionen bei regulären Baumsprachen

Marcial Gaißert

Studiengang:	Informatik
Prüfer/in:	PD Dr. Manfred Kufleitner
Betreuer/in:	PD Dr. Manfred Kufleitner
Beginn am:	3. Dezember 2021
Beendet am:	13. Juni 2022

Kurzfassung

Wir zeigen, dass das Beschränktheitsproblem für eine Verallgemeinerung der *automata with coloring* nach Bala bzw. *desert automata* nach Kirsten mit mehreren priorisierten Zählern und auf Bäumen in PSPACE ist. Dieses Ergebnis nutzen wir, um zu zeigen, dass in doppelt exponentieller Zeit auf einer nichtdeterministischen Turingmaschine entschieden werden kann, ob eine endliche lineare Substitution σ mit $\sigma(L) = R$ für gegebene reguläre Baumsprachen L und R existiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Algorithmen

1 Einleitung

Eine Erweiterung von formalen Sprachen über Wörtern sind die Baumsprachen. Statt Wörtern, also Folgen von Zeichen, sind die Elemente hier Bäume mit beschrifteten Knoten. Dabei gibt es zwei Varianten: *Ranked*, d.h. die Beschriftung eines Knotens definiert die Anzahl seiner Kinder eindeutig, und *unkanked*, d.h. alle Beschriftungen an Knoten beliebigen Grades auftreten können. Auch bei den zur Auswahl stehenden Automatenmodellen gibt es nun Unterschiede nach der Richtung der Auswertung in *top-down* oder *bottom-up*.

Im Gegensatz zu Wortsprachen gibt es zudem mehrere Substitutionsbegriffe auf Baumsprachen, im wesentlichen oi und io [ES77; ES78]. Eine Schwierigkeit besteht darin, dass eine Substitution bei inneren Knoten auch definieren muss, wo deren Kinder im resultierenden Baum vorkommen sollen. Ist hier eine Verdopplung möglich, ist die entstehende Sprache schon für Homomorphismen nicht mehr zwingend regulär. Das HOM-Problem, also die Frage, ob das Bild einer regulären Baumsprache unter einem Homomorphismus regulär ist, ist enthscheidbar nach [GG13]. In dieser Arbeit werden wir uns auf *lineare* Substitutionen, also solche ohne Verdopplungen, konzentrieren. Die zentralen Entscheidungsprobleme sind hier zum einen, ob eine Substitution σ existiert mit $\sigma(L) \subseteq R \Leftrightarrow \sigma(L) \cap \overline{R} = \emptyset$, oder $\sigma(L) = R$, gegebene reguläre Sprachen L und R. Für das erste Problem im Fall von io-Substitutionen wurde in [CDD+22] Entscheidbarkeit gezeigt, auch für Baumsprachen über unendlichen Bäumen. Komplexitätsresultate für beide Probleme mit Substitutionen auf Forests die ausschließlich an Blättern ersetzen wurden gezeigt in [GK21], sowie Resultate bei sogenannten *unranked tree patterns* mit Ersetzung auch an inneren Knoten in [BNS19]. Wir zeigen hier die Entscheidbarkeit von $\sigma(L) = R$ in doppelt exponentieller Zeit auf nichtdeterministischen Turingmaschinen.

Für Wortsprachen wurde 2006 gleichzeitig von Bala [Bal06] und Kirsten [Kir04] gezeigt, dass die Frage, ob eine *endliche* Substitution mit $\sigma(L) = R$ existiert, entscheidbar ist, wobei Bala [Bal06] sogar die Entscheidbarkeit in polynomiellem Platz zeigt. Hierzu verwenden beide ein gewichtetes Automatenmodell, das von Bala als *automata with coloring* und von Kirsten als *desert automata* bezeichnet wird. Dieses erweitern wir in dieser Arbeit zunächst auf mehrere, priorisierte Zähler, und dann auf entsprechend gewichtete top-down-Baumautomaten. Hierfür können wir durch Streichen gewisser Zustände und Konstruktion eines Wortautomaten für Pfade die selben Komplexitätsresultate wie Bala zeigen. Dies erlaubt es, unsere Resultate für die Existenz von linearen Substitutionen auch auf die Existenz endlicher linearer Substitutionen auszuweiten.

2 Grundlagen

2.1 Wörter und Bäume

Ein Alphabet mit Stelligkeit Δ ist eine endliche Menge mit einer Funktion $\mathrm{rk}\colon \Delta \to \mathbb{N}$. Wir schreiben auch $\Delta_r = \{a \in \Delta \mid \mathrm{rk}(a) = r\}$.

Ein Wort w ist eine endliche Folge von Zeichen $a_1, \ldots, a_n$ aus einem endlichen Alphabet Σ, $w = a_1 \cdots a_n$. Dies entspricht einem Element des freien Monoids Σ^* über Σ. Wir bezeichnen das leere Wort mit ε. Für $w \in \Sigma^*, a \in \Sigma$ sei $|w|$ die Länge von w und $|w|_a$ die Anzahl der Vorkommen von a in w. Wenn auf Σ eine Ordnung definiert ist, schreiben wir für $w = a_1 \cdots a_{|w|}$ auch $\min(w) = \min\{a_1, \ldots, a_{|w|}\}$.

Ein Baum t ist eine Funktion auf ein endliches Alphabet Σ aus einer Menge $\mathrm{Pos}(t) \subseteq \mathbb{N}^*$ mit

$$\begin{aligned} \varepsilon &\in \mathrm{Pos}(t) \\ i.n \in \mathrm{Pos}(t), n \in \mathbb{N} &\implies i \in \mathrm{Pos}(t) \\ i.n \in \mathrm{Pos}(t), n \in \mathbb{N}, m < n &\implies i.m \in \mathrm{Pos}(t). \end{aligned}$$

Wir schreiben hier "." als Trennzeichen zwischen die Elemente von $\mathbb{N}$. Ein Baum t heißt *endlich*, wenn $\mathrm{Pos}(t)$ endlich ist, sonst unendlich. Für einen Baum t über eine Alphabet mit Stelligkeit Δ verlangen wir zudem, dass $\mathrm{Pos}(t) \cap \{i.n \mid n \in \mathbb{N}\} = \{1, \ldots, \mathrm{rk}(t(i))\}$ für alle $i \in \mathrm{Pos}(t)$ gilt. Die Menge aller Bäume über einem Alphabet mit Stelligkeit Δ bezeichnen wir mit $T(\Delta)$. Für $i \in \mathrm{Pos}(t)$ sei $t|_i$ der Baum mit $t|_i(j) = t(i.j)$ für alle $i.j \in \mathrm{Pos}(t)$. Wir schreiben Bäume auch als Terme, d.h. für $a \in \Delta_r$, $t_1, \ldots, t_r \in T(\Delta)$ ist

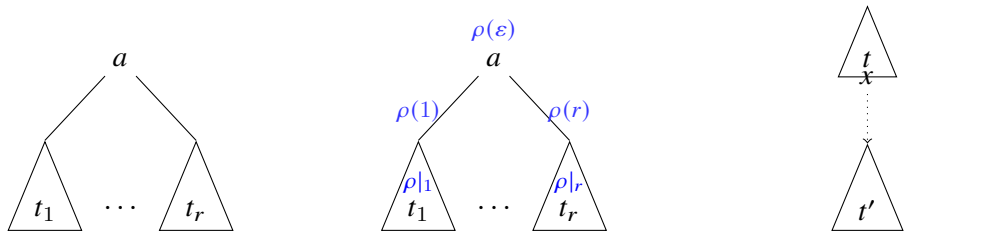
$$\begin{aligned} a(t_1, \ldots, t_r)\colon \{\varepsilon\} \cup \{i.j \mid j \in \mathrm{Pos}(t_i), 1 \le i \le r\} &\to \Delta \\ \varepsilon &\mapsto a \\ i.j &\mapsto t_i(j) \text{ für } 1 \le i \le r \text{ und } j \in \mathrm{Pos}(t_i). \end{aligned}$$

Eine grafische Darstellung von $a(t_1, \ldots, t_r)$ findet sich in Abbildung 2.1a. Wir definieren $\mathrm{Pos}_a(t) = \{i \in \mathrm{Pos}(t) \mid t(i) = a\}$. Es ist $\mathrm{height}(t) = \max\{|i| \mid i \in \mathrm{Pos}(t)\}+1$. Weiter definieren wir für $I \subseteq \mathbb{N}^*$:

$$\mathrm{Paths}(I) = \{(\varepsilon, i_1, \ldots, i_1 \ldots i_n) \mid i_1, i_1.i_2, \ldots, i_n \in \mathbb{N}, \forall 0 \le k \le n\colon i_1 \ldots i_k \in I\}$$

und für $t \in T(\Delta)$:

$$\mathrm{Paths}(t) = \{t(i_1) \cdots t(i_n) \mid (i_1, \ldots, i_n) \in \mathrm{Paths}(\mathrm{Pos}(t))\}.$$



(a) Baum $a(t_1, \ldots, t_r)$ **(b)** Lauf ρ auf $a(t_1, \ldots, t_r)$ **(c)** Ersetzung $t[x \mapsto t']$

Abbildung 2.1: Grafische Darstellungen

2.2 Baumautomaten

Es sind in der Literatur verschiedene Automatenmodelle auf Bäumen gebräuchlich. Diese unterscheiden sich unter anderem in der gedachten Richtung der Auswertung in *bottom-up* und *top-down*. In dieser Arbeit betrachten wir primär nichtdeterministische Top-Down-Baumautomaten, die wir im folgenden noch einmal explizit definieren. Eine Einführung in die unterschiedlichen Modelle findet sich zum Beispiel in [CDG+07].

Definition 2.2.1
Ein nichtdeterministischer Top-Down-Baumautomat (im folgenden abgekürzt als NTA) $\mathcal{A}$ *ist ein 4-Tupel* $\mathcal{A} = (Q, \Delta, \delta, s)$ *wobei*

- Q *eine endliche Menge von Zuständen,*
- Δ *ein Alphabet mit Stelligkeit,*
- $\delta \subseteq \bigcup_{r \in \mathbb{N}} Q \times \Delta_r \times Q^r$ *die Übergangsrelation, und*
- $s \in Q$ *ein ausgezeichneter Startzustand sind.*

Ein Lauf ρ *eines Automaten* $\mathcal{A}$ *auf einem Baum* t *ist eine Funktion* $\rho \colon \mathrm{Pos}(t) \to Q$ *wobei für alle* $i \in \mathrm{Pos}(t)$, $r = \mathrm{rk}(t(i))$ *gilt* $(\rho(i), t(i), \rho(i.0), \ldots, \rho(i.r)) \in \delta$. *Wir stellen einen Lauf grafisch wie in Abbildung 2.1b dar.*

Die von einem Zustand $q \in Q$ *erkannte* Sprache *ist*

$$L(\mathcal{A}, q) \coloneqq \{t \in T(\Delta) \mid \exists \rho \text{ Lauf von } \mathcal{A} \text{ auf } t \colon \rho(\varepsilon) = q\},$$

wobei $L(\mathcal{A}) \coloneqq L(\mathcal{A}, s)$.

Weiter definieren wir für $t \in T(\Delta)$

$$Q(\mathcal{A}, t) \coloneqq \{q \in Q \mid t \in L(\mathcal{A}, q)\}.$$

Für eine endliche Menge $\mathcal{X}$ und einen NTA $\mathcal{A} = (Q, \Delta, \delta, s)$ schreiben wir

$$\mathcal{A}[\mathcal{X}] \coloneqq (Q, \Delta \cup \mathcal{X}, \delta \cup Q \times \mathcal{X}, s),$$

wobei $\mathrm{rk}(\mathcal{X}) = \{0\}$.

2.3 Substitutionen auf Bäumen

Eine *Substitution* ist eine Funktion $\sigma\colon \mathcal{X} \to T(\Delta \cup \{1,\dots,\max(\mathrm{rk}(\mathcal{X}))\})$ wobei $\mathcal{X}$ ein endliches Alphabet mit Stelligkeiten ist und $\sigma(x) \subseteq T(\Delta \cup \{1,\dots,\mathrm{rk}(x)\})$ mit $\mathrm{rk}(1) = \cdots = \mathrm{rk}(\mathrm{rk}(x)) = 0$. Wir nennen σ *linear*, wenn für alle $x \in \mathcal{X}, i \in \{1,\dots,\mathrm{rk}(x)\}, t \in \sigma(x)$ gilt, dass $|\mathrm{Pos}_i(t)| \leq 1$.

Es gibt zwei übliche Möglichkeiten eine Substitution auf einen Baum anzuwenden, oi (by-name) und io (by-value), die von Engelfriet eingeführt wurden [ES77; ES78]. Für lineare Subsitutionen σ ist $\sigma_{oi} = \sigma_{io}$ und wir schreiben auch nur $\sigma = \sigma_{io}$. Da wir lediglich lineare Substitutionen betrachten, definieren wir hier nur $\sigma_{io}\colon T(\Delta \cup \mathcal{X}) \to 2^{T(\Delta)}$, wie folgt:

$$\sigma_{io}(a(t_1,\dots,t_r)) = \{a(t_1',\dots,t_r') \mid \forall i\colon t_i' \in \sigma_{io}(t_i)\} \qquad \forall a \in \Delta_r$$
$$\sigma_{io}(x(t_1,\dots,t_r)) = \{t[1 \mapsto t_1',\dots,r \mapsto t_r'] \mid t \in \sigma(x), \forall i\colon t_i' \in \sigma_{io}(t_i)\} \qquad \forall x \in \mathcal{X}_r$$

Hierbei ist $t[a_1 \mapsto t_1, \dots a_n \mapsto t_n]$ für $t \in T(\Delta), a_1,\dots,a_n \in \Delta_0, t_1,\dots,t_n \in T(\Delta)$ der Baum, den man durch ersetzen von a_i durch t_i in t erhält. Wir stellen diesen grafisch wie in Abbildung 2.1c dar.

Zudem sei $\sigma^{\leq n}(x) = \{t \in \sigma(x) \mid \mathrm{height}(t) \leq n\}$ für $n \in \mathbb{N}$, $x \in \mathcal{X}$.

2.4 Komplexitätstheorie

Wir verwenden die Komplexitätsklassen P, PSPACE = NPSPACE = coNPSPACE, EXPTIME = EXP, NEXPTIME = NEXP, EXPSPACE und

$$2 - \mathsf{NEXPTIME} = \bigcup_{c \in \mathbb{N}} \mathsf{NTIME}\left(2^{2^{n^c}}\right).$$

Für genauere Definitionen und eine Einführung in die Komplexitätstheorie verweisen wir auf [AB09].

3 k-Desert-Automaten

Dieses Kapitel betrachtet eine einfache Verallgemeinerung der Resultate zu *automata with coloring* nach Bala [Bal06], die den *desert automata* nach Kirsten [Kir04] entsprechen. Unser Vorgehen folgt dem von Bala [Bal06].

Definition 3.0.1
Ein k-Desert-Automat $\mathcal{A}$ *ist ein 5-Tupel* $(Q, \Sigma, \delta, s, F)$ *wobei*

- Q *eine endliche Menge von Zuständen,*
- Σ *ein endliches Alphabet*
- $\delta \subseteq Q \times \Sigma \times \{0, 1, \ldots, k\} \times Q$ *die Übergangsrelation,*
- s *der ausgezeichnete Startzustand, und*
- $F \subseteq Q$ *die Menge der Endzustände*

sind.

Ein Lauf ρ *von* $\mathcal{A}$ *auf* $a_0 \cdots a_{k-1}$ *ist eine Folge der Form*

$$p_0 \xrightarrow[c_0]{a_0} \cdots p_{k-1} \xrightarrow[c_{k-1}]{a_{k-1}} p_k$$

mit $(p_i, a_i, c_i, p_{i+1}) \in \delta$ *für alle* $i \in \{0, \ldots, k-1\}$. *Wir schreiben*

$$\mathrm{Runs}_{\mathcal{A}}\left(p_0 \xrightarrow[c_0 \cdots c_{k-1}]{a_0 \cdots a_{k-1}} p_k\right) = \left\{ p_0 \xrightarrow[c_0]{a_0} \cdots p_{k-1} \xrightarrow[c_{k-1}]{a_{k-1}} p_k \;\middle|\; p_1, \ldots, p_{k-1} \in Q \right\}$$

und

$$\mathrm{Runs}_{\mathcal{A}}\left(P \xrightarrow[C]{L} P'\right) = \bigcup \left\{ \mathrm{Runs}_{\mathcal{A}}\left(q \xrightarrow[c]{w} q'\right) \;\middle|\; w \in L, q \in P, q' \in P', c \in C \right\}.$$

In dieser Notation identifizieren wir einelementige Mengen mit ihren Elementen.

Nun definieren wir:

$$\begin{aligned}
\mathrm{Weight}_\ell(\rho) &= \max\left\{ |c_m|_\ell \;\middle|\; \exists p' \xrightarrow[c_p]{w_p} p \xrightarrow[c_m]{w_m} q \xrightarrow[c_s]{w_s} q' \ni \rho \colon c_m \in \{\ell, \ldots, k\}^* \right\} \\
\mathrm{Weight}_\ell(\mathcal{P}) &= \min\{\mathrm{Weight}_\ell(\rho) \mid \rho \in \mathcal{P}\} \\
\mathrm{Weight}(\mathcal{P}) &= \max\{\mathrm{Weight}_\ell(\mathcal{P}) \mid \ell > 0\} \\
\mathrm{Weight}(w) &= \mathrm{Weight}(\mathrm{Runs}_{\mathcal{A}}(s \xrightarrow[\{0,\ldots,k\}^*]{w} F))
\end{aligned}$$

$\mathcal{A}$ *heißt* beschränkt *mit Schranke* η, *wenn für jedes Wort* $w \in L(\mathcal{A})$ *gilt* $\mathrm{Weight}(w) \leq \eta$.

Problem 3.0.1 (Beschränktheit)
Gegeben *Ein k-Desert-Automat $\mathcal{A}$.*

Frage *Ist $\mathcal{A}$ beschränkt?*

Es kann hier davon ausgegangen werden, dass k immer unär kodiert vorliegt, denn nicht verwendete Gewichte ≥ 1 können gestrichen und alle größeren verkleinert werden. Der Rest des Kapitels wird einen Algorithmus zur Lösung dieses Problems entwickeln.

3.1 Semantik von k-Desert-Automaten

Für ein Wort $w \in \Sigma^*$ definieren wir:

$$\begin{aligned} \mathrm{Trans}_{\mathcal{A}}(w) &\subseteq Q \times \{0,\dots,k\} \times Q =: \mathrm{Sem}_{\mathcal{A}} \\ (p,c,q) \in \mathrm{Trans}_{\mathcal{A}}(w) &:\iff \mathrm{Runs}_{\mathcal{A}}\left(p \xrightarrow[\{c,\dots,k\}^*]{w} q\right) \neq \emptyset \\ &\qquad \text{und } c \text{ ist maximal mit dieser Eigenschaft} \end{aligned}$$

Statt $\mathrm{Sem}_{\mathcal{A}}$ und $\mathrm{Trans}_{\mathcal{A}}$ schreiben wir auch Sem und Trans, wenn der Bezug zu $\mathcal{A}$ aus dem Kontext ersichtlich ist.

Wir definieren zwei Operationen auf Sem:

$$\begin{aligned} &\odot, \otimes \colon \mathrm{Sem} \times \mathrm{Sem} \to \mathrm{Sem} \\ &(p,c,q) \in R_1 \odot R_2 :\iff c = \max\{\min\{c_1,c_2\} \mid (p,c_1,r) \in R_1, (r,c_2,q) \in R_2\} \\ &(p,c,q) \in R_1 \otimes R_2 :\iff c = \max\{\max\{c_1,c_2\} \mid (p,c_1,r) \in R_1, (r,c_2,q) \in R_2\} \end{aligned}$$

Hierbei sei $\max\{\}$ undefiniert, d.h. wenn keine $r \in Q$, $(p,c_1,r) \in R_1$ und $(r,c_2,q) \in R_2$ existieren, ist auch $(p,c,r) \notin R_1 \otimes R_2$ und $(p,c,r) \notin R_1 \odot R_2$ für $c \in \{0,\dots,k\}$.

Es gilt $\mathrm{Trans}(w_1) \odot \mathrm{Trans}(w_2) = \mathrm{Trans}(w_1 w_2)$ für beliebige $w_1, w_2 \in \Sigma^*$, und sowohl $\odot$ als auch $\otimes$ sind assoziativ.

Damit definieren wir für Tupel $(S_1,\dots,S_\ell)$

$$\bigotimes(S_1,\dots,S_\ell) = S_1 \otimes \cdots \otimes S_\ell$$

Für Tupel von Wörtern $(w_1,\dots,w_\ell,w_{\mathrm{tail}})$ definieren wir

$$\mathrm{Trans}(w_1,\dots,w_\ell,w_{\mathrm{tail}}) = (\mathrm{Trans}(w_1),\dots,\mathrm{Trans}(w_\ell),\{(p,0,q) \mid (p,c,q) \in \mathrm{Trans}(w_{\mathrm{tail}})\}).$$

Das letzte Element wird hier anders behandelt, was später wesentlich wird.

Für Mengen $P_1, P_2 \subseteq Q$ nennen wir ein Element $S \in \mathrm{Sem}$ P_1-P_2-*beschränkt*, wenn gilt, dass $S \cap P_1 \times \{0,\dots,k\} \times P_2 \subseteq P_1 \times \{0\} \times P_2$. Eine Menge $M \subseteq \mathrm{Sem}$ heißt P_1-P_2-beschränkt, wenn alle ihre Elemente P_1-P_2-beschränkt sind. Wir identifizieren auch hier einelementige Mengen mit ihren Elementen.

Beobachtung 3.1.1
Wenn $S \subseteq \mathrm{Sem}$ P_1-P_2-beschränkt ist, dann auch jede Teilmenge $S' \subseteq S$.

3.2 Faktorisierungen und Beschränktheit

Wir definieren zunächst

$$\text{Factorizations}_L(M,N) = \left\{ (w_1,\dots,w_\ell,w_{\text{tail}}) \;\middle|\; \begin{array}{l} w_1,\dots,w_\ell \in M, w_{\text{tail}} \in N, \\ w_1\cdots w_\ell w_{\text{tail}} \in L \end{array} \right\}.$$

Lemma 3.2.1
Wenn $\bigotimes \text{Trans}(\text{Factorizations}_{L(\mathcal{A})}(M,N))$ *s-F-beschränkt ist,* $|N|_{\max} < |M|_{\max}$ *und* $L(\mathcal{A}) \subseteq M^*N$*, dann ist* $\mathcal{A}$ *beschränkt mit Schranke* $2(|M|_{\max}-1)$*.*

Beweis. Sei also $\bigotimes \text{Trans}(\text{Factorizations}_{L(\mathcal{A})}(M,N))$ *s-F*-beschränkt mit $|N|_{\max} < |M|_{\max}$ und $L(\mathcal{A}) \subseteq M^*N$. Sei weiter $w \in L(\mathcal{A})$ mit

$$g \coloneqq \text{Weight}(w) = \text{Weight}_\ell\left(s \xrightarrow[\{0,\dots,k\}^*]{w} F\right) = \text{Weight}_\ell(\rho)$$

für $\ell > 0$ und $\rho \in \text{Runs}\left(s \xrightarrow[\{0,\dots,k\}^*]{w} F\right)$.

Dann gibt es $q_0 = s, q_1, \dots, q_m \in Q, q_{m+1} \in F, w_1, \dots, w_m \in M, w_{m+1} \in N$ sowie $c_1, \dots, c_{m+1} \in \{0,\dots,k\}^*$ mit

$$\text{Runs}\left(q_0 \xrightarrow[c_1]{w_1} q_1 \xrightarrow[c_2]{w_2} \cdots q_m \xrightarrow[c_{m+1}]{w_{m+1}} q_{m+1}\right) \ni \rho.$$

Dann sind alle $c_i \in \{0,\dots,k\}^*0\{0,\dots,k\}^*$. Sonst gäbe es ein w_i, für das $c > 0$ existiert mit $(q,c,q') \in \text{Trans}(w_i)$, also ein $(s,c',q'') \in \otimes\text{Trans}(w_1,\dots,w_{m+1})$ mit $c' \geq c, q'' \in F$, aber dann wäre die Voraussetzung verletzt. Jeder Faktor aus $\{\ell,\dots,k\}^*$ in $c_1\cdots c_{m+1}$ erstreckt sich höchstens über zwei c_i, da jedes c_i eine 0 enthält. Sei nun also $0 \leq i \leq m+1$ mit $\rho' \in \text{Runs}\left(q_{i-1} \xrightarrow[c_ic_{i+1}]{w_iw_{i+1}} q_{i+1}\right)$ und $\text{Weight}_\ell(\rho') = g$.

Betrachte nun $c'0c''0c''' = c_ic_{i+1}$ mit $c', c''' \in \{0,\dots,k\}^*, c'' \in \{\ell,\dots,k\}^*$ und $|c''|_\ell = g$. Dann ist $g = |c''|_\ell \leq |c''| \leq |c_ic_{i+1}| - 2 \leq 2|M|_{\max} - 2$.

Der Fall, in dem c_{m+1} Teil des optimalen Faktors ist, verläuft analog, da $|c_{m+1}| \leq |M|_{\max}-1$.□

Lemma 3.2.2
Seien $k,n \in \mathbb{N}$ *mit* $n \geq k^k$ *und* $c \in \{1,\dots k\}^n$*. Dann existiert ein Faktor* $c' \in \{\ell,\dots,k\}^*$ *von* c *mit* $|c'|_\ell \geq \left\lceil \sqrt[k]{n} - k \right\rceil$.

Beweis. Für $k = 1$ ist diese Eigenschaft trivialerweise erfüllt. Wenn $|c|_1 \geq \left\lceil \sqrt[k]{n} - k \right\rceil$, ist die Eigenschaft erfüllt mit $\ell = 1$ und $c' \coloneqq c$. Sei nun $|c|_1 < \left\lceil \sqrt[k]{n} - k \right\rceil$, also $|c|_1 \leq \sqrt[k]{n} - k$ und $c = c_0 1 c_1 \cdots 1 c_{|c|_1}$ für $c_0,\dots,c_{|c|_1} \in \{2,\dots,k\}^*$. Dann existiert ein $0 \leq i \leq |c|_1$ mit $|c_i|$ maximal und dafür ist

$$|c_i| \geq \frac{\sum_{j=0}^{|c|_1} |c_j|}{|c|_1+1} = \frac{|c|-|c|_1}{|c|_1+1} \geq \frac{n - \sqrt[k]{n} + k}{\sqrt[k]{n} - k + 1} > \frac{n-\sqrt[k]{n}}{\sqrt[k]{n}} = \frac{n}{\sqrt[k]{n}} - 1 = n^{\frac{k-1}{k}} - 1.$$

Durch zeichenweises Ersetzen von j durch $j-1$ in c_i für alle $2 \leq j \leq k$ erhalten wir ein Wort $c'' \in \{1,\dots,k-1\}^*$ mit $n' = |c''| = |c_i|$. Für dieses existiert induktiv ein Faktor $c''' \in \{\ell-1,\dots,k-1\}^*$ mit

$$|c'''|_{\ell-1} \geq \left\lceil \sqrt[k-1]{n'} - k + 1 \right\rceil \geq \sqrt[k-1]{n'} - 2 > \sqrt[k-1]{n^{\frac{k-1}{k}} - 1} - k + 1 > \sqrt[k-1]{n^{\frac{k-1}{k}}} - k = \sqrt[k]{n} - k.$$

Dieses c''' entspricht nun einem Faktor $c' \in \{\ell,\dots,k\}^*$ in c_i, und damit in c mit Gewicht $|c'|_\ell = |c'''|_{\ell-1} > \sqrt[k]{n} - k$, also $|c'|_\ell \geq \lceil \sqrt[k]{n} - k \rceil$. □

Lemma 3.2.3
Wenn $\bigotimes \mathrm{Trans}(\mathrm{Factorizations}_{L(\mathcal{A})}(M,N))$ *nicht s-F-beschränkt ist, dann existiert ein Wort* $w \in L(\mathcal{A})$*, so dass* $\mathrm{Weight}(w) \geq \left\lceil \sqrt[k]{|M|_{\min}} - k \right\rceil$

Beweis. Sei also $\bigotimes \mathrm{Trans}(\mathrm{Factorizations}_{L(\mathcal{A})}(M,N))$ nicht s-F-beschränkt.

Dann existieren ein $q \in F$ und $w \in L(\mathcal{A})$ mit $w = w_1 \cdots w_m w_{m+1}$, $w_1,\dots,w_m \in M, w_{m+1} \in N$ und $(s,c,q) \in \bigotimes \mathrm{Trans}(w_1,\dots,w_m,w_{m+1})$ für $c > 0$.

Dann gilt für jeden Lauf

$$\rho \in \mathrm{Runs}\left(\underbrace{q_0}_{=s} \xrightarrow[c_1]{w_1} q_1 \cdots \xrightarrow[c_{m+1}]{w_{m+1}} \underbrace{q_{m+1}}_{=q} \right),$$

dass $c = \max\{\min(c_1),\dots,\min(c_{m+1})\}$ und zudem $(q_{i-1},\min(c_i),q_i) \in \mathrm{Trans}(w_i)$ für jedes $1 \leq i \leq m+1$. Insbesondere existiert ein i mit $c = \min(c_i)$.

Es ist $|c_i| \geq |M|_{\min} =: n$ und $c_i \in \{1,\dots,k\}^*$. Nach Lemma 3.2.2 existiert dann ein $\ell > 0$ und ein Faktor $c_i' \in \{\ell,\dots,k\}^*$ in c_i mit $|c_i'|_\ell \geq \lceil \sqrt[k]{n} - k \rceil$. Also ist $\mathrm{Weight}_\ell(\rho) \geq \left\lceil \sqrt[k]{n} - k \right\rceil$ und damit $\mathrm{Weight}(w) \geq \left\lceil \sqrt[k]{|M|_{\min}} - k \right\rceil$ für ein $w \in L(\mathcal{A})$. □

Lemma 3.2.4
$\bigotimes \mathrm{Trans}(\mathrm{Factorizations}_{L(\mathcal{A})}(M,N))$ *ist s-F-beschränkt genau dann, wenn*

$\bigotimes \mathrm{Trans}(\mathrm{Factorizations}_{\Sigma^*}(M,N))$ *s-F-beschränkt ist.*

Beweis. Es ist $\mathrm{Factorizations}_{L(\mathcal{A})}(M,N) \subseteq \mathrm{Factorizations}_{\Sigma^*}(M,N)$. Also folgt die erste Richtung wegen Beobachtung 3.1.1 direkt.

Angenommen, es gibt ein $(w_1,\dots,w_i) \in \mathrm{Factorizations}_{\Sigma^*}(M,N) \setminus \mathrm{Factorizations}_{L(\mathcal{A})}(M,N)$, für das $\bigotimes \mathrm{Trans}(w_1,\dots,w_i)$ nicht s-F-beschränkt ist. Dann existiert auch ein $q \in F$ mit $(s,c,q) \in \bigotimes \mathrm{Trans}(w_1,\dots,w_i)$. Insbesondere existiert dann aber ein Lauf in

$$\mathrm{Runs}\left(s \xrightarrow[\{0,\dots,k\}^*]{w_1\cdots w_i} F \right)$$

also ist $w_1 \cdots w_i \in L(\mathcal{A})$ und damit $(w_1,\dots,w_i) \in \mathrm{Factorizations}_{L(\mathcal{A})}(M,N)$, im Widerspruch zur Annahme. □

3.3 Ein Algorithmus für das Beschränktheitsproblem

Satz 3.3.1
Sei $\mathcal{A}$ ein beschränkter k-Desert-Automat. Dann ist $\mathcal{A}$ beschränkt mit Schranke

$$\kappa = 2^{2(k+2)^{n^2}+1}.$$

Beweis. Sei $\mathcal{A}$ ein beschränkter k-Desert-Automat mit $|Q| =: n$ und Schranke η. Für jedes $S \in \mathrm{Sem}$ und alle $p, q \in P$ gilt nach Definition $|S \cap \{p\} \times \{0, \dots, k\} \times \{q\}| \leq 1$. Also gilt $|\mathrm{Sem}_{\mathcal{A}}| \leq (k+2)^{n^2}$.

Wenn $|M|_{\min} > (\eta + k)^k$, folgt, da $\mathcal{A}$ beschränkt ist, und wegen Lemma 3.2.3 und Lemma 3.2.4, dass $\bigotimes \mathrm{Trans}(\mathrm{Factorizations}(M, N))$ s-F-beschränkt ist.

Betrachte $T_i = \mathrm{Trans}(\{w \in \Sigma^* : i \leq |w| \leq 2i\}) \subseteq \mathrm{Sem}_{\mathcal{A}}$ für $i > 0$. Es ist

$$T_i = T_{i-1} \odot \mathrm{Trans}(\Sigma \cup \Sigma^2).$$

Da $T_i \subseteq \mathrm{Sem}_{\mathcal{A}}$, gibt es höchstens $2^{(k+2)^{n^2}}$ verschiedene T_i, sodass die Mengen $T_1, \dots, T_{2^{(k+2)^{n^2}}}$ alle möglichen Werte durchlaufen. Analog gilt dies für die $T'_i = \mathrm{Trans}(\Sigma^{\leq i}) = T'_{i-1} \odot \mathrm{Trans}(\Sigma)$.

Da $\bigotimes \mathrm{Trans}(\mathrm{Factorizations}(\{w \in \Sigma^* \mid i \leq |w| \leq 2i\}, \Sigma^{\leq i}))$ durch die beiden Mengen T_i und T'_i eindeutig definiert ist, und s-F-beschränkt ist für $i = (\eta + k)^k + 1$, ist es dies auch für ein $1 \leq i \leq 2^{2(k+2)^{n^2}}$.

Also ist $\mathcal{A}$ nach Lemma 3.2.1 auch beschränkt mit Schranke $2(2^{2(k+2)^{n^2}} - 1) \leq 2^{2(k+2)^{n^2}+1}$. □

Korollar 3.3.1
Sei $\mathcal{A}$ ein k-Desert-Automat. Dann ist $\mathcal{A}$ beschränkt genau dann, wenn $\bigotimes \mathrm{Trans}(\mathrm{Factorizations}_{\Sigma^*}(\{w \in \Sigma^* \mid \kappa \leq |w| \leq 2\kappa\}, \Sigma^{<\kappa}))$ s-F-beschränkt ist.

Satz 3.3.2
Problem 3.0.1 ist PSPACE*-vollständig.*

Beweis. Für $k = 1$ entsprechen die hier vorgestellten k-Desert-Automaten den *automata with coloring* bei [Bal06] bzw. den *desert automata* bei [Kir04]. Dort wird unter anderem PSPACE-Härte gezeigt, welche sich direkt auf dieses Problem überträgt.

Das Komplement von Problem 3.0.1 lässt sich zudem mit Algorithmus 3.2 in polynomiellem Platz auf einer nichtdeterministischen Turingmaschine lösen, also ist das Problem in coNPSPACE = NPSPACE = PSPACE.

Die Algorithmen benötigen jeweils nur polynomiellen Platz, da Elemente von Sem nur polynomiellen und Zähler nur logarithmischen Platz in ihrer Größe benötigen. Alle Zähler bleiben $\leq 2(\kappa + 1)$, was nur exponentiell groß ist. Weiter werden in einem Durchlauf zu jedem Zeitpunkt nur endlich viele solche Elemente und Zähler benötigt. □

Algorithmus 3.1 Nichtdeterministische Wahl eines Elements begrenzter Länge in einer Halbgruppe

function ChooseProduct(chooseS: $() \to S$, $\circ: S \times S \to S$, $\check{n} \in \mathbb{N}$, $\hat{n} \in \mathbb{N} \cup \{\infty\}$)
  fail if $\hat{n} < \check{n}$
  $s \leftarrow$ chooseS(), $k \leftarrow 1$
  while $k < \check{n}$ **do**
    $s' \leftarrow$ chooseS()
    $s \leftarrow s \circ s', k \leftarrow k + 1$
  end while
  while $k \leq \hat{n}$ **do**
    if choose{True, False} **then**
      return s
    end if
    $s' \leftarrow$ chooseS()
    $s \leftarrow s \circ s', k \leftarrow k + 1$
  end while
  fail
end function

Dabei ist chooseS ein Prozedurparameter, der bei jedem Aufruf neu ausgewertet wird und ggf. eine neue nichtdeterministische Wahl trifft.

Algorithmus 3.2 Entscheidungsalgorithmus für das Komplement von Problem 3.0.1

function ChooseTransN($\check{n} \in \mathbb{N}$, $\hat{n} \in \mathbb{N} \cup \{\infty\}$)
  // Wählt ein Element von Trans($\Sigma^{\geq \check{n}} \cap \Sigma^{\leq \hat{n}}$)
  return ChooseProduct($\lambda()$.choose(Trans(Σ)), $\odot, \check{n}, \hat{n}$)
end function
function ChooseTransFactorization($\mathcal{A} = (Q, \Sigma, \delta, s, F)$, $\check{n} \in \mathbb{N}$, $\hat{n} \in \mathbb{N}$)
  // Wählt ein Element von $\bigotimes$ Trans(Factorizations($\Sigma^{\leq \hat{n}} \cap \Sigma^{\geq \check{n}}, \Sigma^{<\check{n}}$))
  $s \leftarrow$ ChooseProduct($\lambda()$.ChooseTransN($\check{n}, \hat{n}$), $\otimes, 0, \infty$)
  $s' \leftarrow$ ChooseProduct($\lambda()$.choose{$\{(p, 0, q) \mid (p, a, c, q) \in \delta\} \mid a \in \Sigma\}, 0, \check{n} - 1$)
  return $s \otimes s'$
end function
function IsUnlimited($\mathcal{A} = (Q, \Sigma, \delta, s, F)$)
  $S \leftarrow$ ChooseTransFactorization($\mathcal{A}, \kappa + 1, 2(\kappa + 1)$)
  if $\exists q \in F \exists c > 0$: $(s, c, q) \in S$ **then**
    accept // S ist nicht s-F-beschränkt
  else
    fail
  end if
end function

4 k-Desert-Baum-Automaten

In diesem Kapitel betrachten wir eine Verallgemeinerung der im vorherigen Kapitel vorgestellten k-Desert-Automaten auf Bäume. Dabei betrachten wir ein Modell, das nichtdeterministische Top-Down-Baumautomaten erweitert:

Definition 4.0.1
Ein k-Desert-Baum-Automat $\mathcal{A}$ *ist ein 5-Tupel* (Q, Δ, δ, s) *wobei*

- Q *eine endliche Menge von Zuständen,*
- $\Delta = \bigcup_\ell \Delta_\ell$ *ein endliches Alphabet mit Stelligkeit,*
- $\delta \subseteq \prod_\ell Q \times \Delta_\ell \times \{0, \dots, k\} \times Q^\ell$ *die Übergangsrelation,*
- $s \in Q$ *der ausgezeichnete Startzustand*

sind.

Wenn $|\{(q, a, i, p_1, \dots, p_k) \in \delta \mid i \in \{0, 1\}, \forall \ell \colon p_\ell \in Q\}| = 1$ *für alle* $q \in Q, a \in \Delta$ *heißt er* deterministisch.

Ein Lauf ρ *auf einem (endlichen oder unendlichen) Baum* t *ist ein Paar von Funktionen* $(\tau_\rho \colon \mathrm{Pos}(t) \to Q, \gamma_\rho \colon \mathrm{Pos}(t) \to \{0, \dots, k\})$ *mit* $(\tau_\rho(i), t(i), \gamma_\rho(i), \tau_\rho(i.1), \dots, \tau_\rho(i.k)) \in \delta$ *für alle* $i \in \mathrm{Pos}(t)$. *Wir schreiben auch nur* ρ *statt* τ_ρ. ρ *heißt* initial, *wenn* $\tau_\rho(\varepsilon) = s$.

Hierauf definieren wir:

$$
\begin{aligned}
\mathrm{Weight}_\ell(\rho) &= \max \left\{ |c|_\ell \;\middle|\; \begin{array}{l} \exists (i_1, \dots, i_m) \in \mathrm{Paths}(\mathrm{Pos}(t)) \colon \\ c = \gamma_\rho(i_1) \cdots \gamma_\rho(i_m) \in \{\ell, \dots, k\}^* \end{array} \right\} \\
\mathrm{Weight}_\ell(\mathcal{P}) &= \min\{\mathrm{Weight}_\ell(\rho) \mid \rho \in \mathcal{P}\} \\
\mathrm{Weight}(\mathcal{P}) &= \max\{\mathrm{Weight}_\ell(\mathcal{P}) \mid \ell > 0\} \\
\mathrm{Weight}(t, q) &= \mathrm{Weight}(\{\rho \mid \rho \textit{ Lauf von } \mathcal{A} \textit{ auf } t, \rho(\varepsilon) = q\}) \\
\mathrm{Weight}(t) &= \mathrm{Weight}(t, s)
\end{aligned}
$$

$\mathcal{A}$ *heißt* beschränkt, *wenn es eine Zahl* η *gibt, sodass für jeden Baum* t *gilt* $\mathrm{Weight}(t) \leq \eta$.

Für jeden k*-Desert-Baumautomaten* $\mathcal{A}$ *ist*

$$\mathrm{NTA}(\mathcal{A}) = (Q, \Delta, \{(q, a, q_1, \dots, q_\ell) \mid (q, a, g, q_1, \dots, q_\ell) \in \delta\}, s)$$

ein NTA. Wir identifizieren diese und nennen auch den NTA $\mathcal{A}$.

Problem 4.0.1 (Beschränktheit)
Gegeben *Ein* k*-Desert-Baum-Automat* $\mathcal{A}$.

Frage *Ist* $\mathcal{A}$ *beschränkt?*

Wir definieren zudem

$$\text{Paths}(\rho) = \{(t(i_1), \gamma_\rho(i_1)) \cdots (t(i_n), \gamma_\rho(i_n)) \mid (i_1, \ldots, i_n) \in \text{Paths}(\text{Pos}(t))\}.$$

4.1 Reduzierte Automaten

Satz 4.1.1
Für einen k-Desert-Automaten $\mathcal{A} = (Q, \Delta, \delta, s)$ lässt sich in polynomieller Zeit die Menge $Q' \subseteq Q$ aller $q \in Q$ bestimmen mit $L(\mathcal{A}, q) \neq \emptyset$.

Weiter gilt $L(\mathcal{A}, q) = L(\mathcal{A}', q)$ für alle $q \in Q'$ und $\mathcal{A} = (Q', \Delta, \delta|_{Q'}, s)$, wenn $s \in Q'$. Wir nennen $\mathcal{A}'$ reduziert.

Dabei sei $\delta|_{Q'} = \delta \cap \prod_\ell Q' \times \Delta_\ell \times \{0, \ldots, k\} \times Q'^\ell$.

Algorithmus 4.1 Bestimmung der produktiven Zustände eines Top-Down-NTA

```
Q' ← ∅
changed ← True
while changed do
    changed ← False
    for (p, a, i, q_1, ..., q_r) ∈ δ do
        if q_1, ..., q_r ∈ Q' then
            Q' ← Q' ∪ {p}
            changed ← True
        end if
    end for
end while
```

Beweis. Algorithmus 4.1 durchläuft die Äußerste Schleife höchstens $|Q|$ mal, hat also insgesamt eine Laufzeit in $\mathcal{O}(|Q| \cdot |\delta|)$, was polynomiell in der Größe von $\mathcal{A}$ ist. Es bleibt die Korrektheit von Algorithmus 4.1 zu zeigen.

Sei $t \in L(\mathcal{A}, q)$ für ein $q \in Q$. Mit Induktion nach height(t). Wenn height$(t) = 1$ ist, dann ist $t = a \in \Delta_0$, also $(q, a, i) \in \delta$ für ein $i \in \{0, \ldots, k\}$, da sonst $a \notin L(\mathcal{A}, q)$, und q wird in der ersten Iteration zu Q' hinzugefügt. Ansonsten ist $t = a(t_1, \ldots, t_r)$ für ein $a \in \Delta_r$ und es gibt $q_1, \ldots, q_r \in Q$ und $i \in \{0, \ldots, k\}$ mit $(q, a, i, q_1, \ldots, q_r) \in \delta$. Nach Induktionsvoraussetzung werden $q_1, \ldots, q_r$ irgendwann zu Q' hinzugefügt, also ist in der darauffolgenden Iteration $(q, a, i, q_1, \ldots, q_r) \in \delta$ und $q_1, \ldots, q_r \in Q'$ und es wird q zu Q' hinzugefügt.

Wenn andererseits ein $q \in Q$ von Algorithmus 4.1 zu einem Zeitpunkt zu Q' hinzugefügt wird, existiert auch ein $t \in L(\mathcal{A}, q)$: Wenn q zu Q' hinzugefügt wird, muss $(q, a, g, q_1, \ldots, q_r) \in \delta$ mit $q \in \Delta_r$,$0 \leq g \leq k$ und $q_1, \ldots, q_r \in Q'$ gelten. Induktiv existieren bereits $t_i \in L(\mathcal{A}_i, q_i)$ für alle $1 \leq i \leq r$. Dann ist $a(t_1, \ldots, t_r) \in L(\mathcal{A}, q)$.

Im zweiten Teil der Aussage ist $L(\mathcal{A}, q) \supseteq L(\mathcal{A}', q)$ klar, da jeder Lauf von $\mathcal{A}'$ auch einer von $\mathcal{A}$ ist. Sei nun andererseits ρ ein Lauf auf $t \in T(\Delta)$ von $\mathcal{A}$, aber nicht von $\mathcal{A}'$. Dann gibt es ein i mit $\tau_\rho(i) \in Q \setminus Q'$ und $\rho|_i$ ist ein Lauf auf $t|_i$, also ist $t|_i \in L(\mathcal{A}, \tau_\rho(i))$, im Widerspruch zur Konstruktion von Q'. □

4.2 Ein Automat für Pfade

Definition 4.2.1
Für einen k-Desert-Baum-Automaten $\mathcal{A} = (Q, \Delta, \delta, s)$ definieren wir den k-Desert-Automaten $\mathrm{Paths}(\mathcal{A}) = (Q \cup \{q_f\}, \Delta, \delta', s, \{q_f\})$ *mit*

$$(q, a, g, q') \in \delta' :\Longleftrightarrow (q, a, g, p_1, \ldots, p_{j-1}, q', p_{j+1}, \ldots, p_m) \in \delta$$
$$(q, a, g, q_f) \in \delta' :\Longleftrightarrow (q, a, g) \in \delta$$

Lemma 4.2.1
Ein reduzierter k-Desert-Baum-Automat $\mathcal{A}$ ist beschränkt genau dann, wenn $\mathrm{Paths}(\mathcal{A})$ *beschränkt ist.*

Beweis. Wir zeigen, dass für alle $\eta \in \mathbb{N}$ ein Wort $w \in L(\mathrm{Paths}(\mathcal{A}))$ mit $\mathrm{Weight}(w) \geq \eta$ existiert genau dann, wenn ein Baum $t \in L(\mathcal{A})$ mit $\mathrm{Weight}(t) \geq \eta$ existiert.

Sei also ein solches $w = a_1 \cdots a_n$ und $\ell \in \{1, \ldots, k\}$, $\rho \in \mathrm{Runs}\left(s \xrightarrow[\{0,\ldots,k\}^*]{w} q_f\right)$ mit $\mathrm{Weight}(w) = \mathrm{Weight}_\ell\left(\mathrm{Runs}\left(s \xrightarrow[\{0,\ldots,k\}^*]{w} q_f\right)\right) = \mathrm{Weight}_\ell(\rho) \geq \eta$. Nun konstruiere t wie folgt: Es ist $t_n = a_n \in \Delta_0$ und $(\rho(n-1), a_n, g) \in \delta$ für ein $g \in \{0, \ldots, k\}$. In jedem Schritt existiert nun für $i \in \mathrm{Pos}(w)$ ein $(\rho(i-1), a_i, g, q_1, \ldots, q_{i_i-1}, \rho(i), q_{i_i+1}, \ldots, q_r) \in \delta$ für ein $g \in \{0, \ldots, k\}$. Induktiv existiert bereits $t_i \in L(\mathcal{A}, \rho(i))$. Da $\mathcal{A}$ reduziert ist, existieren $t'_m \in L(\mathcal{A}, q_m)$ für alle $m \in \{1, \ldots, i_i - 1, i_i + 1, \ldots, r\}$ und wir erhalten $t_{i-1} = a_i(t'_1, \ldots, t'_{i_i-1}, t_i, t'_{i_i+1}, \ldots, t'_r)$.

Sei nun ein Lauf ρ_t auf $t = t_0$. Es ist nun für jedes $c = \gamma_\rho(i_\alpha) \cdots \gamma_\rho(i_\beta) \in \{\ell, \ldots, k\}^*$ mit $(i_1, \ldots, i_\beta) \in \mathrm{Paths}(\mathrm{Pos}(t)), 1 \leq \alpha \leq \beta$ auch $\mathrm{Weight}_\ell(\rho_t) \geq |c|_\ell$ und wegen $\mathrm{Weight}_\ell(\rho) \geq \eta$ existiert ein solches c mit $|c|_\ell \geq \eta$, entsprechend den Positionen der a_i.

Sei nun ein Baum t mit $\mathrm{Weight}(t) \geq \eta$ und ρ ein Lauf von $\mathcal{A}$ auf t. mit $\mathrm{Weight}(\rho) \geq \eta$ Dann existieren $\ell \in \{1, \ldots, k\}$, $(i_1, \ldots, i_n) \in \mathrm{Paths}(\mathrm{Pos}(t))$ und $c = \gamma_\rho(i_\alpha) \cdots \gamma_\rho(i_\beta)$ für $1 \leq i_\alpha \leq i_\beta \leq n$ mit $|c|_\ell = \mathrm{Weight}(t) \geq \eta$. Dann ist $(\tau_\rho(i_j), t(i_j), g_j, \tau_\rho(i_j.1), \ldots, \tau_\rho(i_{j+1}), \ldots, \tau_\rho(i_j.\mathrm{rk}(t(i_j)))) \in \delta$ für $1 \leq j < n$ und $(\tau_\rho(i_n), t(i_n), g_n) \in \delta$, also auch $(\tau_\rho(i_j), t(i_j), g_j, \tau_\rho(i_{j+1})) \in \delta'$ und $(\tau_\rho(i_n), t(i_n), g_n, q_f) \in \delta'$. Zudem ist $|c| = |g_\alpha \cdots g_\beta|_\ell \geq \eta$, also ist auch $\mathrm{Weight}(t(i_1) \cdots t(i_n)) \geq \eta$ und damit $t(i_1) \cdots t(i_n) \in L(\mathrm{Paths}(\mathcal{A}))$. □

Satz 4.2.1
Problem 4.0.1 ist PSPACE*-vollständig.*

Beweis. Das Problem ist in PSPACE, da ein k-Desert-Baum-Automat nach Theorem 4.1.1 in Polynomialzeit in einen reduzierten k-Desert-Tree-Automaten und dieser nach Definition 4.2.1 in einen k-Desert-Automaten umgewandelt werden kann, der genau dann beschränkt ist, wenn dies der ursprüngliche Automat war, nach Lemma 4.2.1. Dieses Problem ist aber nach Theorem 3.3.2 in PSPACE.

Aus einem k-Desert-Automat $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ lässt sich zudem ein k-Desert-Baumautomaten $\mathcal{A}' = (Q, \Sigma \cup \{\$\}, \delta \cup \{(q, \$) \mid q \in F\}, s)$ mit $\mathrm{rk}(a) = 1$ für $a \in \Sigma$, $\mathrm{rk}(\$) = 0$ konstruieren. Dann ist $\mathcal{A}$ beschränkt genau dann, wenn $\mathcal{A}'$ beschränkt ist. Da ersteres aber nach Theorem 3.3.2 PSPACE-schwierig ist, gilt das auch hier. □

5 Lineare Substitutionen auf Baumsprachen

5.1 Ein NTA für $\sigma(L)$

Wir zeigen die Verallgemeinerung von [CDG+07, Theorem 1.4.3] auf lineare Substitutionen (statt Homomorphismen):

Satz 5.1.1
Für einen NTA $\mathcal{A} = (Q, \Delta \cup \mathcal{X}, \delta, s)$ und eine lineare, reguläre Substitution σ, gegeben durch NTAs $\mathcal{A}_{\sigma(x)} = (Q_x, \Delta \cup \{1, \dots, \mathrm{rk}(x)\}, \delta_x, s_x)$ für alle $x \in \mathcal{X}$, existiert ein 1-Desert-Baumautomat bzw. NTA $\sigma(\mathcal{A})$ mit

1. *$L(\sigma(\mathcal{A}), q) = \sigma(L(\mathcal{A}), q)$ für alle $q \in Q$.*
2. *Für jeden Baum $t \in L(\sigma(\mathcal{A}, q))$ und Lauf ρ von $\sigma(\mathcal{A})$ auf t mit $\rho(\varepsilon) = q$ existiert für $\eta = \mathrm{Weight}(\rho)$ ein $t' \in L(\mathcal{A}, q)$ mit $t \in \sigma^{\leq \eta+1}(t')$. Umgekehrt existiert für jedes $t \in \sigma^{\leq \eta+1}(t')$ mit $t' \in L(\mathcal{A}, q)$ ein Lauf ρ von $\sigma\mathcal{A}$ auf t mit $\mathrm{Weight}(\rho) = \eta$ und $\rho(\varepsilon) = q$.*
3. *$\sigma(\mathcal{A})$ hat polynomielle Größe und kann in logarithmischem Platz konstruiert werden.*

Definition 5.1.1
Wir definieren $\sigma(\mathcal{A}) = (Q', \Delta, \delta', s)$ mit $Q' = Q \cup \bigcup_{x \in \mathcal{X}} Q_x \times Q^{\mathrm{rk}(x)}$ und

- *$(q, a, 0, q_1, \dots, q_r) \in \delta'$ für alle $(q, a, q_1, \dots, q_r) \in \delta$ mit $a \in \Delta_r$.*
- *$((p, q_1, \dots, q_r), a, 1, (p_1, q_1, \dots, q_r), \dots (p_{r'}, q_1, \dots, q_r)) \in \delta'$ gilt, wenn ein $x \in \mathcal{X}_r$ existiert mit $(p, a, p_1, \dots, p_{r'}) \in \delta_x$.*
- *$(q, a, 0, (p_1, q_1, \dots, q_r), \dots, (p_{r'}, q_1, \dots, q_r)) \in \delta'$, wenn ein $x \in \mathcal{X}_r$ existiert mit $(q, x, q_1, \dots, q_r) \in \delta$, $(s_x, a, p_1, \dots, p_{r'}) \in \delta_x$.*
- *$((q, q_1, \dots, q_r), a, g, p_1, \dots, p_{r'}) \in \delta'$ wenn $i \in \{1, \dots, r\}$ existiert mit $(q, i) \in \delta_x$, $(q_i, a, g, p_1, \dots, p_{r'}) \in \delta'$. Dies ist wohldefiniert, da q_i nicht wieder ein $(q, q_1, \dots, q_r)$ sein kann.*

Beweis. Die Konstruktion von $\sigma(\mathcal{A})$ ist leicht in logarithmischem Platz durchführbar.

Sei $\eta \in \mathbb{N}$ und $t \in \sigma^{\leq \eta+1}(L(\mathcal{A}, q)) \subseteq \sigma(L(\mathcal{A}, q))$. Dann existiert ein $t' \in L(\mathcal{A}, q)$ mit $t \in \sigma^{\leq \eta+1}(t')$. Sei ρ' ein Lauf von $\mathcal{A}$ auf t' mit $\rho(\varepsilon) = q$. Wenn $t' = a(t'_1, \dots, t'_r)$ für ein $a \in \Delta_r$, dann ist $t = a(t_1, \dots, t_r)$ mit $\forall i \colon t_i \in \sigma(t'_i)$. Für alle $1 \leq i \leq r$ gilt wegen $t'_i \in L(\mathcal{A}, \rho'(i))$ und $t_i \in \sigma^{\leq \eta+1}(t'_i)$ induktiv, dass $t_i \in L(\sigma(\mathcal{A}), \rho'(i))$ und $\mathrm{Weight}_1(t_i, \rho'(i)) \leq \eta$. Zudem ist $(q, a, \rho'(1), \dots, \rho'(r)) \in \delta \cap \delta'$, also ist $t \in L(\sigma(\mathcal{A}), q)$. Diese Situation ist in Abbildung 5.1 dargestellt.

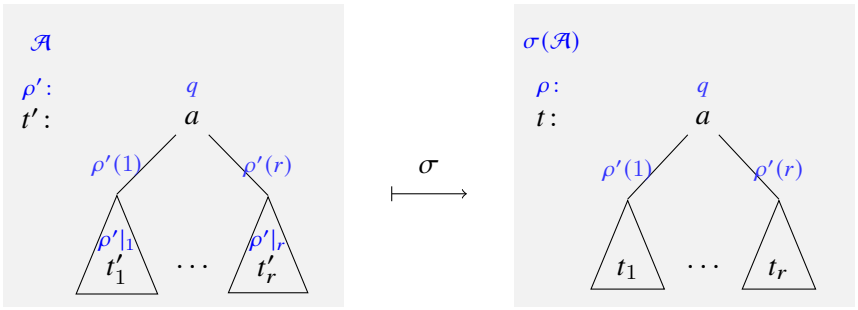


Abbildung 5.1: Fall $t = a(t_1, \ldots, t_r) \in \sigma(a(t'_1, \ldots, t'_r))$

Sei nun $t' = x(t'_1, \ldots, t'_r)$ für ein $x \in X_r$. Dann ist $t = t_x[1 \mapsto t_1, \ldots, r \mapsto t_r]$ für $t_x \in \sigma(x)$ und $\forall i\colon t_i \in \sigma^{\leq \eta+1}(t'_i)$ (vgl. Abbildung 5.2). Wegen $t'_i \in L(\mathcal{A}, \rho'(i))$ und $t_i \in \sigma^{\eta+1}(t'_i)$ ist induktiv $t_i \in L(\sigma(\mathcal{A}), \rho'(i))$ und ρ_i ein Lauf von $\sigma(\mathcal{A})$ mit $\mathrm{Weight}_1(\rho_i) \leq \eta$ auf t_i für $1 \leq i \leq r$. Sei ρ_x ein Lauf von $\mathcal{A}_{\sigma(x)}$ auf t_x. Definiere $\rho\colon \mathrm{Pos}(t) \to Q'$ mittels:

- $\rho(\varepsilon) = q$
- $\rho(j) = (\rho_x(j), \rho'(1), \ldots, \rho'(r))$ für alle $j \in \mathrm{Pos}(t_x) \setminus \{\varepsilon\}$.
- $\rho|_{\mathrm{Pos}_i(t_x).j} = \rho_i|_j$ für alle $1 \leq i \leq r$, $1 \leq j \leq \mathrm{rk}(t'(j))$.

Für die $j \in \mathrm{Pos}_\Delta(t_x) \setminus \{\varepsilon\}$ ist nach Definition $(\rho(j), t(j), \rho(j.1), \ldots, \rho(j.\mathrm{rk}(t(j)))) \in \delta'$ (da $(\rho_x(j), t_x(j), \rho_x(j.1), \ldots, \rho_x(j.\mathrm{rk}(t_x(j)))) \in \delta_x$ und $t(j) = t_x(j)$). Für die $j \in \mathrm{Pos}(t) \setminus \mathrm{Pos}(t_x)$ sind die Eigenschaften eines Laufs für ρ bekanntermaßen erfüllt. Es ist $(q, t(\varepsilon), \rho(1), \ldots, \rho(\mathrm{rk}(t(\varepsilon)))) \in \delta'$, da $(q, x, \rho'(1), \ldots, \rho'(r)) \in \delta$ und $(q, t(\varepsilon), \rho_x(1), \ldots, \rho_x(\mathrm{rk}(t(\varepsilon)))) \in \delta_x$. Zudem ist für $i_j \in \mathrm{Pos}_j(t_x)$ auch $(\rho(i_j), t(i_j), \rho(i_j.1), \ldots, \rho(i_j.\mathrm{rk}(t(i_j)))) \in \delta'$, da $\rho(i_j) = (\rho_x(i_j), \rho'(1), \ldots, \rho'(r))$, $(\rho_x(i_j), j) \in \delta_x$ und $(\rho'(j), t(i_j), \rho(i_j.1), \ldots, \rho(i_j.\mathrm{rk}(t(i_j)))) \in \delta'$ wegen $\rho_j(\varepsilon) = \rho'(j)$. Zusammen ist ρ ein Lauf von $\sigma(\mathcal{A})$ auf t und damit $t \in L(\sigma(\mathcal{A}), q)$. Die Erweiterung auf einen Lauf des 1-Desert-Automaten ist offensichtlich. Hier ist für alle $1 \leq i \leq r$, $j \in \mathrm{Pos}_i(t_x)$: $\tau_\rho(j) \in Q$, also $\gamma_\rho(j) = 0$ und $\mathrm{Weight}_1(\rho) \leq \max\{\eta, \mathrm{height}(t_x) - 1\} \leq \eta$.

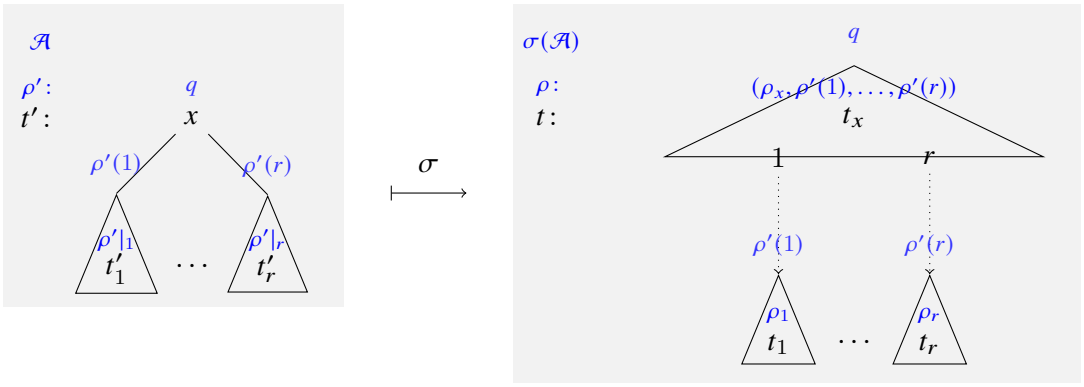


Abbildung 5.2: Fall $t = t_x[1 \mapsto t_1, \ldots, r \mapsto t_r] \in \sigma(x(t'_1, \ldots, t'_r))$

Sei $t \in L(\sigma(\mathcal{A}), q)$ für ein $q \in Q'$ und ρ ein Lauf von $\sigma(\mathcal{A})$ auf $t = a(t_1, \ldots, t_r)$ mit $\rho(\varepsilon) = q$, $q_i = \rho(i)$ für $1 \leq i \leq r$ und $\eta \coloneqq \text{Weight}(\rho) = \text{Weight}(t, q)$. Es ist $(q, a, g, q_1, \ldots, q_r) \in \delta'$.

- Wenn $q, q_1, \ldots, q_r \in Q$ (vgl. Abbildung 5.3), ist $g = 0$ und $(q, a, q_1, \ldots, q_r) \in \delta$. Also ist $t_i \in L(\sigma(\mathcal{A}), q_i)$ und $\text{Weight}(t_i, q_i) = \eta$ für $1 \leq i \leq r$. Induktiv erhalten wir $t_i \in \sigma^{\leq \eta+1}(t_i')$ für ein $t_i' \in L(\mathcal{A}, q_i)$. Damit ist auch $t = a(t_1, \ldots, t_r) \in \sigma^{\leq \eta+1}(t')$ für $t' = a(t_1', \ldots, t_r')$.

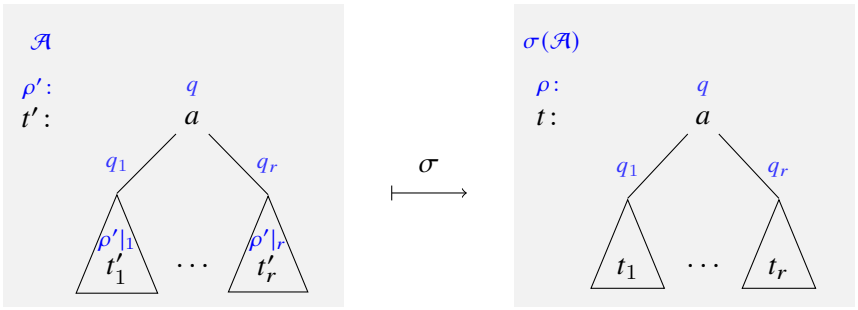


Abbildung 5.3: Fall $q, q_1, \ldots, q_r \in Q$

- Wenn $q, q_1, \ldots, q_r \in Q_x \times Q^{r_x}, g = 1$ für ein $x \in \mathcal{X}_{r_x}$ (vgl. Abbildung 5.4), dann seien $q = (p, q_1', \ldots, q_{r_x}')$ und $q_i = (p_i, q_1', \ldots, q_{r_x}')$, und es ist $(p, p_1, \ldots, p_r) \in \delta_x$. Dann existieren $u_i \in L(\mathcal{A}_{\sigma(x)}, p_i)$, $v_j \in L(\sigma(\mathcal{A}), q_j') \; \forall 1 \leq j \leq r$, und $\eta \coloneqq \text{Weight}(t, q)$ mit $t_i = u_i[1 \mapsto v_1, \ldots, r_x \mapsto v_{r_x}]$, $u_i \in L(\mathcal{A}_{\sigma(x)}, p_i)$ und

$$\begin{aligned}\eta &= \max\{\text{Weight}(v_1, q_1'), \ldots, \text{Weight}(v_{r_x}, q_{r_x}'), \text{height}(u_1) + 1, \ldots, \text{height}(u_r) + 1\} \\ &= \max\{\text{Weight}(v_1, q_1'), \ldots, \text{Weight}(v_{r_x}, q_{r_x}'), \text{height}(a(u_1, \ldots, u_r))\}.\end{aligned}$$

  Kompatible v_j existieren, da j jeweils nur in einem der u_i vorkommt. Zudem ist induktiv $v_j \in \sigma^{\leq \eta+1}(v_j')$ für $v_j' \in L(\mathcal{A}, q_j')$ für $1 \leq j \leq r_x$. Weiter ist $t = a(t_1, \ldots, t_r) = a(u_1, \ldots, u_r)[1 \mapsto v_1, \ldots, r_x \mapsto v_{r_x}]$. Kommt ein j nicht in $a(u_1, \ldots, u_r)$ vor, kann das $j \mapsto u_j'$ aus den Substitutionen ausgelassen werden und $q_j' = \perp$ ist möglich.

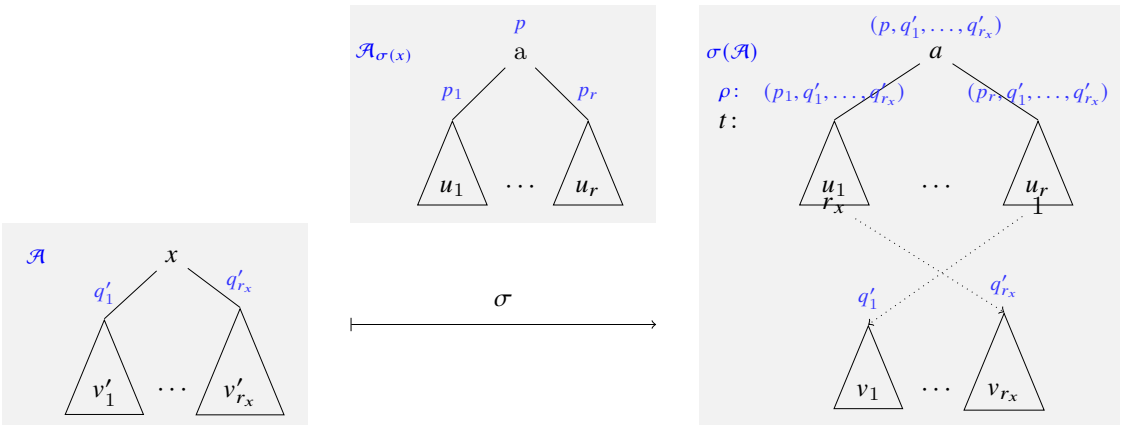


Abbildung 5.4: Fall $q, q_1, \ldots, q_r \in Q_x \times Q^{r_x}$

- Wenn $q \in Q$ und $q_1, \ldots, q_r \in Q_x \times Q^{r_x}$ für ein $x \in X_{r_x}$ (vgl. Abbildung 5.5), dann sei $q_i = (p_i, q'_1, \ldots, q'_{r_x})$ für $1 \leq i \leq r_x$ und es ist $g = 0$, und $(s_x, a, p_1, \ldots, p_r) \in \delta_x$ sowie $(q, x, q'_1, \ldots, q'_{r_x}) \in \delta$. Dann ist auch $t \in L(\sigma(\mathcal{A}), (s_x, q'_1, \ldots, q'_{r_x}))$ und $\mathrm{Weight}(t, (s_x, q'_1, \ldots, q'_{r_x})) = \max\{\eta, h\}$, wobei h die Länge des längsten 1-Pfades beginnend in der Wurzel ist. Dann existiert ein $u \in L(\mathcal{A}_\sigma, s_x)$ und $v_i \in L(\sigma(\mathcal{A}), q'_i)$ für $1 \leq i \leq r_x$ mit $t = u[1 \mapsto v_1, \ldots, r_x \mapsto v_{r_x}]$ und $\mathrm{height}(u) = h$ sowie $\mathrm{Weight}(t, (s_x, q'_1, \ldots, q'_{r_x})) = \max\{h, \mathrm{Weight}(u_i, q'_i) \mid 1 \leq i \leq r_x\}$. Für $1 \leq i \leq r_x$ ist induktiv $v_i \in \sigma^{\leq \eta_i + 1}(v'_i)$ für $\eta_i = \mathrm{Weight}(v_i, q'_i)$, $v'_i \in L(\mathcal{A}, q'_i)$. Folglich ist $u \in \sigma^{\leq \eta+1}(t')$ für $t' = x(v'_1, \ldots, v'_{r_x})$.

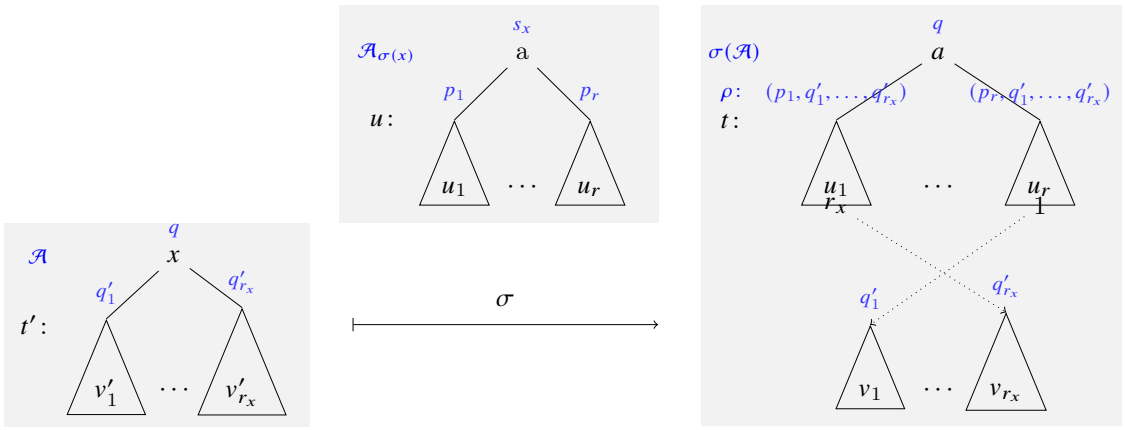


Abbildung 5.5: Fall $q \in Q, q_1, \ldots, q_r \in Q_x \times Q^{r_x}$

- Sonst ist $q = (p, q'_1, \ldots, q'_{r_x})$, $(p, i) \in \delta_x$ für ein $x \in X_{r_x}$ (vgl. Abbildung 5.6), $(q'_i, a, g, q_1, \ldots, q_r) \in \delta'$, und $g = 0$. Dann ist $i \in L(\mathcal{A}_{\sigma(x)}, q)$ und es gilt $t \in L(\sigma(\mathcal{A}), q'_i)$ mit $\mathrm{Weight}(t, q'_i) = \eta$. Also gilt wie gezeigt $t \in \sigma^{\leq \eta+1}(L(\mathcal{A}, q'_i))$ und damit ist $t = i[1 \mapsto v_1, \ldots, r_x \mapsto v_{r_x}]$, wobei $v_i = t \in \sigma^{\leq \eta+1}(v'_i)$ für einen Baum $v'_i \in L(\mathcal{A}, q'_i)$ und $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{r_x}$ beliebig. Zudem ist dann $\mathrm{Weight}(t, q) = \max\{\eta, 1\} = \max\{\eta, \mathrm{height}(i)\}$. □

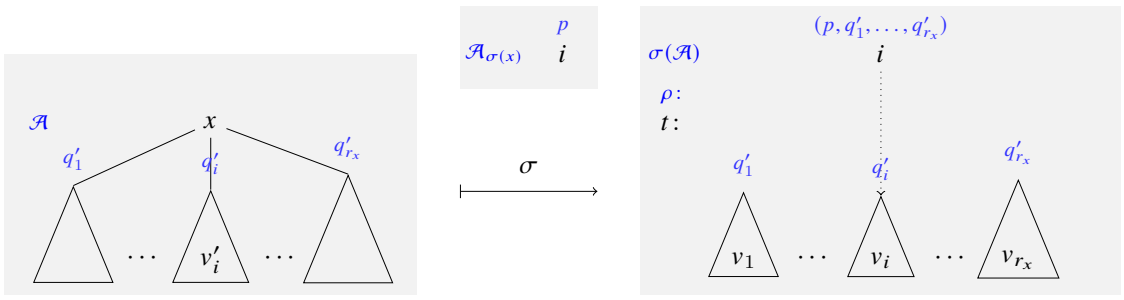


Abbildung 5.6: Fall $q = (p, q'_1, \ldots, q'_{r_x}) \in Q_x \times Q^{r_x}, (p, i) \in \delta_x$

5.2 Entscheidungsprobleme mit gegebenen Substitutionen

Mit der Konstruktion aus Definition 5.1.1 und mit Theorem 5.1.1 folgen direkt:

Problem 5.2.1
Gegeben *NTAs $\mathcal{A}$, $\mathcal{B}$, eine lineare reguläre Substitution σ (durch NTAs).*

Frage *$\sigma(L(\mathcal{A})) \cap L(\mathcal{B}) \neq \emptyset$?*

Korollar 5.2.1
Problem 5.2.1 ist in P.

Beweis. Es kann in logarithmischem Platz nach Theorem 5.1.1 ein NTA für $\sigma(L(\mathcal{A}))$ konstruiert werden. Dann können die beiden top-down-NTAs nach [CDG+07, Theorem 1.6.1] in bottom-up-NTAs umgewandelt werden, ein bottom-up-NTA für deren Schnitt nach [CDG+07, Theorem 1.3.1] bestimmt werden, und Leerheit entschieden werden wie in [CDG+07, Theorem 1.7.4]. □

Problem 5.2.2
Gegeben *NTAs $\mathcal{A}$, $\mathcal{B}$, eine lineare reguläre Substitution σ (durch NTAs).*

Frage *$\sigma(L(\mathcal{A})) = L(\mathcal{B})$?*

Korollar 5.2.2
Problem 5.2.2 ist in EXPTIME.

Beweis. Es kann auch hier nach Theorem 5.1.1 ein NTA für $\sigma(L(\mathcal{A}))$ konstruiert werden. Nach [CDG+07, Theorem 1.6.1] kann dieser in einen bottom-up-NTA konvertiert und nach [CDG+07, Theorem 1.7.8] darauf Äquivalenz in Exponentialzeit entschieden werden.□

5.3 Saturierung auf NTAs

Sei $\mathcal{A} = (Q, \Delta, \delta, s)$ ein NTA und $t \in T(\Delta \cup \{1, \ldots, r\})$ mit $\mathrm{rk}(i) = 0$ für alle $1 \leq i \leq r$. Dann definieren wir

$$\mu_{\mathcal{A},r}(t) = \left\{ (q, q_1, \ldots, q_r) \in Q \times (Q \cup \{\bot\})^r \;\middle|\; \begin{array}{c} \exists \text{ Lauf } \rho \text{ von } \mathcal{A}[\{1,\ldots,r\}] \forall i \in \{1,\ldots,r\}: \\ \rho(\mathrm{Pos}_i) \subseteq \{q_i\} \vee (\rho(\mathrm{Pos}_i) = \emptyset \wedge q_i = \bot) \end{array} \right\}.$$

Damit definieren wir für $L \subseteq T(\Delta \cup \{1, \ldots, r\})$ die Saturierung

$$\widehat{L}_{\mathcal{A},r} = \{t \in T(\Delta) \mid \exists t' \in L \colon \mu_{\mathcal{A},r}(t) = \mu_{\mathcal{A},r}(t'), \forall 1 \leq i \leq r \colon |\mathrm{Pos}_i(t)| \leq 1\}$$

und damit die saturierte Substitution $\hat{\sigma}_{\mathcal{B}}$ mit

$$\hat{\sigma}_{\mathcal{B}}(x) = \widehat{\sigma(x)}_{\mathcal{B},r} \; \forall x \in X_r, r \in \mathbb{N}.$$

Lemma 5.3.1
Sei ein NTA $\mathcal{A} = (Q, \Delta, \delta, s)$, σ eine lineare Substitution, $P \subseteq Q$, $t' \in T(\Delta \cup X)$ und $t \in \hat{\sigma}_{\mathcal{A}}(t')$. Dann existiert $\check{t} \in \sigma(t')$ mit $Q(\mathcal{A}, t) = Q(\mathcal{A}, \check{t})$.

Beweis. Ohne Beschränkung der Allgemeinheit sei $t' \in T(\mathcal{X})$, durch Einführen von $x_a \in \mathcal{X}_r$ mit $\sigma(x_a) = \{a(1, \ldots, r)\}$ für jedes $a \in \Delta_r$.

Seien nun $t' = x(t'_1, \ldots, t'_r)$ für $r \in \mathbb{N}$, $x \in \mathcal{X}_r$, $t'_1, \ldots, t'_r \in T(\mathcal{X}_r)$. Dann existieren $t_i \in \hat{\sigma}_{\mathcal{A}}(t'_i)$ für $1 \leq i \leq r$ und ein $t_x \in \hat{\sigma}_{\mathcal{A}}(x)$ mit $t = t_x[1 \mapsto t_1, \ldots, r \mapsto t_r]$. Induktiv existiert für jedes $1 \leq i \leq r$ ein $\check{t}_i \in \sigma(x)$ mit $Q(\mathcal{A}, t_i) = Q(\mathcal{A}, \check{t}_i)$. Weiter existiert nach Definition von $\hat{\sigma}_{\mathcal{A}}$ ein $\check{t}_x$ mit $\mu_{\mathcal{A},r}(t_x) = \mu_{\mathcal{A},r}(\check{t}_x)$. Setze $\check{t} = \check{t}_x[1 \mapsto \check{t}_1, \ldots, r \mapsto \check{t}_r]$.

Sei nun $q \in Q$ und ρ ein Lauf von $\mathcal{A}$ auf t mit $\rho(\varepsilon) = q$. Sei ρ_x die Einschränkung von ρ auf $\mathrm{Pos}(t_x)$. Dann ist ρ_x ein Lauf von $\mathcal{A}[\{1, \ldots, r\}]$ auf t_x. Dann existiert ein Lauf $\check{\rho}_x$ von $\mathcal{A}[\{1, \ldots, r\}]$ auf $\check{t}_x$ mit $\rho_x(\varepsilon) = \check{\rho}_x(\varepsilon) = q$ und für alle $1 \leq i \leq r$ gilt $\rho_x(\mathrm{Pos}_i(t_x)) = \check{\rho}_x(\mathrm{Pos}_i(\check{t}_x)) = \{q_i\}$ oder $\mathrm{Pos}_i(t_x) = \mathrm{Pos}_i(\check{t}_x) = \emptyset$. Es ist $q_i \in Q(\mathcal{A}, t_i)$, also auch $q_i \in Q(\mathcal{A}, \check{t}_i)$ und es existiert ein Lauf $\check{\rho}_i$ von $\mathcal{A}$ auf $\check{t}_i$. Damit existiert ein Lauf $\check{\rho}$ von $\mathcal{A}$ auf $\check{t}$:

$$\check{\rho}(i) = \begin{cases} \check{\rho}_x(i) & i \in \mathrm{Pos}(\check{t}_x) \\ \check{\rho}_h(i.j) & 1 \leq h \leq r, i \in \mathrm{Pos}_h(\check{t}_x), j \in \mathrm{Pos}(\check{t}_h) \end{cases}$$

Sei andererseits $\hat{\rho}$ ein Lauf von $\mathcal{A}$ auf $\check{t}$ mit $\check{\rho}(\varepsilon) = q$. Sei $\check{\rho}_x$ die Einschränkung von ρ auf $\mathrm{Pos}(\check{t}_x)$. Dann ist $\check{\rho}_x$ ein Lauf von $\mathcal{A}[\{1, \ldots, r\}]$ auf $\check{t}_x$. Dann existiert ein Lauf ρ_x von $\mathcal{A}[\{1, \ldots, r\}]$ auf t_x mit $\rho_x(\varepsilon) = \check{\rho}_x(\varepsilon) = q$ und $\rho_x(\mathrm{Pos}_i(t_x)) = \check{\rho}_x(\mathrm{Pos}_i(\check{t}_x)) = \{q_i\}$ oder $\mathrm{Pos}_i(t_x) = \mathrm{Pos}_i(\check{t}_x) = \emptyset$ für alle $1 \leq i \leq r$, da $\mu_{\mathcal{A},r}(t_x) = \mu_{\mathcal{A},r}(\check{t}_x)$. Es ist $q_i \in Q(\mathcal{A}, \check{t}_i)$, also auch $q_i \in Q(\mathcal{A}, t_i)$ und es existiert ein Lauf ρ_i von $\mathcal{A}$ auf t_i. Damit existiert auch ein Lauf ρ von $\mathcal{A}$ auf t:

$$\rho(i) = \begin{cases} \rho_x(i) & i \in \mathrm{Pos}(t_x) \\ \rho_h(i.j) & 1 \leq h \leq r, i \in \mathrm{Pos}_h(t_x), j \in \mathrm{Pos}(t_h) \end{cases}$$ □

Lemma 5.3.2
Seien $t' \in T(\Delta \cup \mathcal{X})$ und $\mathcal{A} = (Q, \Delta, \delta, s)$ ein NTA mit $q \in \mathcal{A}$. Es ist $\sigma(t') \cap L(\mathcal{A}, q) \neq \emptyset$ genau dann, wenn $\hat{\sigma}_{\mathcal{A}}(t') \cap L(\mathcal{A}, q) \neq \emptyset$.

Beweis. Die Richtung von links nach rechts ist klar wegen $\sigma(t') \subseteq \hat{\sigma}_{\mathcal{A}}(t')$.

Sei nun $t \in \hat{\sigma}_{\mathcal{A}}(t') \cap L(\mathcal{A}, q)$. Dann ist $q \in Q(\mathcal{A}, t)$ und nach Lemma 5.3.1 existiert $\check{t} \in \sigma(t')$ mit $Q(\mathcal{A}, \check{t}) = Q(\mathcal{A}, t) \ni q$, also ist $\check{t} \in \sigma(t') \cap L(\mathcal{A}, q)$. □

Lemma 5.3.3
Wenn $\sigma(L) = L(\mathcal{A}, q)$, dann auch $\hat{\sigma}_{\mathcal{A}}(L) = L(\mathcal{A}, q)$.

Beweis. Es ist nach Definition $\sigma(L) \subseteq \hat{\sigma}_{\mathcal{A}}(L)$. Sei $\sigma(L) \subseteq L(\mathcal{A}, q)$ und $t \in \hat{\sigma}(t')$, $t' \in L$. Dann existiert $\check{t} \in \sigma(t')$ mit $Q(\mathcal{A}, t) = Q(\mathcal{A}, \check{t}) \ni q$, also $t \in L(\mathcal{A}, q)$. Damit folgt für $\sigma(L) = L(\mathcal{A}, q)$ direkt $\hat{\sigma}_{\mathcal{A}}(L) = L(\mathcal{A}, q)$. □

Lemma 5.3.4
Für einen NTA $\mathcal{A} = (Q, \Delta, \delta, s)$, $r \in \mathbb{N}$ und ein Element $\tau \in Q \times (Q \cup \{\bot\})^r$ ist

$$\mu^{-1}_{\mathcal{A}}(\tau) \coloneqq \{t \in T(\Delta \cup \{1, \ldots, r\}) \mid \tau \in \mu_{\mathcal{A}}(t)\}$$

regulär und es kann in logarithmischem Platz ein NTA für $\mu^{-1}_{\mathcal{A}}(\tau)$ mit gleicher Größe wie $\mathcal{A}$ konstruiert werden.

Beweis. Für $\tau = (q, q_1, \ldots, q_r)$ sei $\mathcal{A}_\tau = (Q, \Delta, \delta', q)$ mit

- $\delta \subseteq \delta'$.
- $(q_i, i) \in \delta'$ für $1 \leq i \leq r$ und $q_i \neq \bot$.

Diese Konstruktion ist leicht in logarithmischem Platz möglich.

Sei $t \in \mu^{-1}_{\mathcal{A}}(\tau)$. Dann existiert ein Lauf ρ von $\mathcal{A}[\{1, \ldots, r\}]$ auf t mit $\rho(\mathrm{Pos}_i(t)) \subseteq \{q_i\}$. ρ erfüllt an allen $\mathrm{Pos}_\Delta(t)$ die Eigenschaften eines Laufs von $\mathcal{A}_\tau$ auf t. Zudem ist für alle $1 \leq i \leq r$ auch $(q_i, i) \in \delta'$, weshalb die Eigenschaft auch an $\mathrm{Pos}_i(t)$ erfüllt ist.

Sei andererseits $t \in L(\mathcal{A}_\tau, q)$. Dann existiert ein Lauf ρ von $\mathcal{A}_\tau$ auf t. Da alle Übergänge von $\mathcal{A}_\tau$ auch in $\mathcal{A}[\{1, \ldots, r\}]$ möglich sind, ist ρ auch ein Lauf von $\mathcal{A}[\{1, \ldots, r\}]$ auf t. Zudem ist $\rho(\varepsilon) = q$ und, da nur dann $(p, i) \in \delta'$ enthalten ist, ist $\rho(\mathrm{Pos}_i(t)) \subseteq \{q_i\}$. Also ist $t \in \mu^{-1}_{\mathcal{A}}(\tau)$. □

Lemma 5.3.5
Für $r \in \mathbb{N}$, $L \subseteq T(\Delta \cup \{1, \ldots, r\})$ mit $\forall t \in L, 1 \leq i \leq r \colon |\mathrm{Pos}_i(t)| \leq 1$ und $\mathcal{B} = (Q, \Delta, \delta, s)$ existiert ein NTA $\mathcal{A}$ mit $L(\mathcal{A}) = \widehat{L}_{\mathcal{B},r}$ und $\mathcal{A}$ hat exponentielle Größe in in der Größe von $\mathcal{B}$.

Beweis. Sei $\mathcal{R} = \{\mu_{\mathcal{B},r}(t) \mid t \in L\}$.

Für jedes $R \in \mathcal{R}$ ist

$$\mu^{-1}_{\mathcal{B},r}(R) = \bigcap_{\tau \in R} \mu^{-1}_{\mathcal{B},r}(\tau) \cap \bigcap_{\tau \in Q^{r+1} \setminus R} T(\Delta) \setminus \mu^{-1}_{\mathcal{B},r}(\tau).$$

Hierfür lässt sich mit Schnitt und Komplement[CDG+07, Theorem 1.3.1] aus den NTAs für die $\mu^{-1}_{\mathcal{B},r}(\tau)$ ein NTA mit höchstens $\left(2^{|Q|}\right)^{|R|} = 2^{|Q| \cdot |R|}$ Zuständen in polynomiellem Platz konstruieren. Es ist $|R| \leq |Q| \cdot (|Q| + 1)^r$.

Es ist nun $\widehat{L}_{\mathcal{B},r} = \bigcup_{R \in \mathcal{R}} \mu^{-1}_{\mathcal{B},r}(R)$ und hierfür existiert mit der üblichen Konstruktion[CDG+07, Theorem 1.3.1] ein NTA der Größe $\leq |\mathcal{R}| \cdot 2^{|Q| \cdot (|Q|+1)^r} \leq 2^{2|Q| \cdot (|Q|+1)^r}$. □

5.4 Existenz von linearen Substitutionen

Problem 5.4.1
Gegeben *NTAs $\mathcal{A}$, $\mathcal{B}$.*

Frage *Existiert eine lineare Substitution σ mit $\sigma(L(\mathcal{A})) \cap L(\mathcal{B}) \neq \emptyset$?*

Für die eng verwandten Problem mit $\sigma(L(\mathcal{A})) \subseteq L(\mathcal{B})$ bzw. $\sigma(L(\mathcal{A})) \cap L(\mathcal{B}) \neq \emptyset$ mit *compressed tree patterns* statt Substitutionen folgt EXPTIME-Vollständigkeit bzw. P-Vollständigkeit auch direkt aus [BNS19, Lemma 23, Theorem 27]. Trotzdem betrachten wir dieses Problem hier mit Hilfe der obigen Konstruktion als Hinführung auf unsere weiteren Resultate und erhalten folgendes Ergebnis:

Lemma 5.4.1
Problem 5.4.1 ist in NEXPTIME.

Beweis. Rate nichtdeterministisch für jedes $r \in \mathbb{N}$ und $x \in \mathcal{X}_r$ einen NTA $\mathcal{A}_{\sigma(x)}$ wie in Lemma 5.3.5 durch Raten von $\mathcal{R}$ und Durchführen der dort beschriebenen Konstruktion mit $\mathcal{B}$. Es kann mit einer Vergrößerung um Faktor 2^r für jeden Automaten sichergestellt werden, dass die so bestimmte Substitution linear ist.

Entscheide dann in Polynomialzeit hierin, also in Exponentialzeit in der ursprünglichen Eingabe, ob $\sigma(L(\mathcal{A})) \cap L(\mathcal{B}) \neq \emptyset$.

Wenn ja, ist eine lineare Substitution mit den gewünschten Eigenschaften gefunden. Wenn nein, dann existiert nach Konstruktion der $\mathcal{A}_{\sigma(x)}$ keine lineare Substitution σ mit $\hat{\sigma}_{\mathcal{B}}(L(\mathcal{A})) \cap L(\mathcal{B}) \neq \emptyset$, nach Lemma 5.3.2 also auch keine lineare Substitution mit $\sigma_{\mathcal{B}}(L(\mathcal{A})) \cap L(\mathcal{B}) \neq \emptyset$. □

Auf ähnliche Weise betrachten wir nun:

Problem 5.4.2
Gegeben *NTAs $\mathcal{A}$, $\mathcal{B}$.*

Frage *Existiert eine lineare Substitution σ mit $\sigma(L(\mathcal{A})) = L(\mathcal{B})$?*

Satz 5.4.1
Problem 5.4.2 ist in 2 – NEXPTIME.

Beweis. Es kann hier wie im Beweis von Lemma 5.4.1, vorgegangen werden mit dem Unterschied, dass am Ende auf Äquivalenz geprüft wird, was nach [CDG+07, Theorem 1.7.8] in exponentieller Zeit, also in doppelt exponentieller Zeit in der Eingabe möglich ist.

Wenn so ein passendes σ gefunden wird, ist die Antwort zweifelsohne korrekt. Wenn nicht, existiert kein σ mit $\hat{\sigma}_{\mathcal{B}}(L(\mathcal{A})) = L(\mathcal{B})$ und nach Lemma 5.3.3 auch kein σ mit $\sigma(L(\mathcal{A})) = L(\mathcal{B})$. □

5.5 Existenz von endlichen linearen Substitutionen

Endliche Substitutionen sind für das Teilmengenproblem direkt gegeben, im Sinne der Vollständigkeit betrachten wir trotzdem:

Problem 5.5.1
Gegeben *NTAs $\mathcal{A}$, $\mathcal{B}$.*

Frage *Existiert eine endliche lineare Substitution σ mit $\sigma(L(\mathcal{A})) \cap L(\mathcal{B}) \neq \emptyset$?*

Satz 5.5.1
Problem 5.5.1 ist in NEXPTIME.

Beweis. Sei $t \in \sigma(L(\mathcal{A})) \cap L(\mathcal{B})$. Mit $\check{\sigma}(x) = \{t_x \in \sigma(x) \mid \text{size}(t_x) \leq \text{size}(t)\}$ ist dann auch $t \in \check{\sigma}(L(\mathcal{A})) \cap L(\mathcal{B})$. Also ist dieses Problem äquivalent zu Problem 5.4.1. □

Schließlich kommen wir zum Hauptresultat dieses Kapitels:

Problem 5.5.2
Gegeben *NTAs $\mathcal{A}$, $\mathcal{B}$.*

Frage *Existiert eine endliche lineare Substitution σ mit $\sigma(L(\mathcal{A})) = L(\mathcal{B})$?*

Satz 5.5.2
Problem 5.5.2 ist in 2 – NEXPTIME.

Beweis. Wir gehen vor wie in Theorem 5.4.1, prüfen aber für den k-Desert-Automaten für $\hat{\sigma}_{\mathcal{A}}(L(\mathcal{A}))$ zusätzlich mittels Theorem 4.2.1, dass dieser beschränkt ist. Dies ist in polynomiellem Platz in seiner Größe, also in exponentiellem Platz in der Eingabe, möglich, und damit auch in doppelt exponentieller Zeit.

Sei σ eine lineare Substitution, und $\hat{\sigma}_{\mathcal{B}}(\mathcal{A})$ beschränkt mit Schranke η. Sei weiter $\hat{\sigma}_{\mathcal{B}}(L(\mathcal{A})) = L(\mathcal{B})$. Dann ist $\hat{\sigma}_{\mathcal{B}}^{\leq \eta+1}(x) = \{t \in \hat{\sigma}_{\mathcal{B}}(x) \mid \text{height}(t) \leq \eta + 1\}$ eine endliche lineare Substitution und $\sigma^{\leq \eta+1}(L(\mathcal{A})) \subseteq \hat{\sigma}_{\mathcal{B}}(L(\mathcal{A}))$. Sei nun $t \in L(\hat{\sigma}_{\mathcal{B}}(\mathcal{A}))$. Dann ist $g = \text{Weight}_1(t) \leq \eta$, also nach Theorem 5.1.1 $t \in \hat{\sigma}_{\mathcal{B}}^{\leq g+1}(L(\mathcal{A})) \subseteq \hat{\sigma}_{\mathcal{B}}^{\leq \eta+1}(L(\mathcal{A}))$. Also ist auch $\hat{\sigma}_{\mathcal{B}}^{\leq \eta+1}(L(\mathcal{A})) = L(\mathcal{B})$.

Sei σ eine endliche lineare Substitution mit der Eigenschaft $\sigma(L(\mathcal{A})) = L(\mathcal{B})$. Dann gilt für $\eta \coloneqq \max\{\text{height}(t) \mid t \in \sigma(x), x \in X\} - 1$, dass $\sigma^{\leq \eta+1} = \sigma$. Sei ein Baum $t \in L(\hat{\sigma}_{\mathcal{B}}(\mathcal{A})) = L(\mathcal{B}) = \sigma(L(\mathcal{A}))$. Dann ist $t \in \sigma^{\leq \eta+1}(L(\mathcal{A})) \subseteq \hat{\sigma}_{\mathcal{B}}^{\leq \eta+1}(L(\mathcal{A}))$. Also ist $\text{Weight}_1(t) \leq \eta$ nach Theorem 5.1.1. Folglich ist $\hat{\sigma}_{\mathcal{B}}(\mathcal{A})$ beschränkt. □

6 Fazit und Ausblick

In dieser Arbeit wurde zunächst das Modell der *automata with coloring* [Bal06] bzw. *desert automata* [Kir04] auf k priorisierte Zähler erweitert, und gezeigt, dass auch hier das Beschränktheitsproblem PSPACE-vollständig ist. In einem weiteren Schritt konnte dies auf ein Modell erweitert werden, dass statt auf endlichen (Wort-)Automaten auf nichtdeterministischen Top-Down-Baumautomaten basiert. Durch eine Reduktion über einen Wortautomat für die möglichen Pfade in erkannten Bäumen ist auch hier Beschränktheit in PSPACE.

Eine Betrachtung ähnlicher Fragestellungen aufbauend auf weiteren Sprachmodellen, zum Beispiel auf Grammatiken mit gefärbten Terminalen oder algebraischen Modellen, scheint vielversprechend für die Betrachtung ähnlicher Probleme.

Weiter haben wir, als Verallgemeinerung von [CDG+07, Theorem 1.4.3] eine explizite Konstruktion eines Baumautomaten für $\sigma(L(\mathcal{A}))$ angegeben, gegeben einen Top-Down-Baumautomaten $\mathcal{A}$ und eine reguläre Substitution. Damit konnten wir zeigen, dass die Fragestellung "$\exists\sigma\colon \sigma(L(\mathcal{A})) = L(\mathcal{B})$?" nichtdeterministisch in doppelt exponentieller Zeit entschieden werden können. Unter Anwendung der eingeführten k-Desert-Baumautomaten erhalten wir hier die selben Resultate auch für die Existenz *endlicher* linearer Substitutionen.

Für "$\exists\sigma\colon \sigma_{io}(L(\mathcal{A})) \cap L(\mathcal{B}) \neq \emptyset$" ist Entscheidbarkeit, auch auf unendlichen Baumsprachen, bekannt [CDD+22]. Hier wären folglich noch die Frage für endliche Substitutionen offen, sowie Komplexitätsresultate. Auch die Entscheidbarkeit von "$\exists\sigma\colon \sigma_{io}(L(\mathcal{A})) = L(\mathcal{B})$" ist offen. Für oi sind alle diese Fragestellungen offen. Zudem sind uns zu den meisten der Resultate keine Schwierigkeitsresultate bekannt. Das Problem der Existenz endlicher Substitutionen ist in allen dieser Fälle noch offen.

Literaturverzeichnis

[AB09] S. Arora, B. Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009 (zitiert auf S. 15).

[Bal06] S. Bala. „Complexity of regular language matching and other decidable cases of the satisfiability problem for constraints between regular open terms“. In: *Theory of Computing Systems* 39.1 (2006), S. 137–163 (zitiert auf S. 11, 17, 21, 37).

[BNS19] I. Boneva, J. Niehren, M. Sakho. „Regular Matching and Inclusion on Compressed Tree Patterns with Context Variables“. In: *Language and Automata Theory and Applications*. Springer International Publishing, 2019, S. 343–355. ISBN: 978-3-030-13435-8 (zitiert auf S. 11, 33).

[CDD+22] C. Camino, V. Diekert, B. Dundua, M. Marin, G. Sénizergues. „Regular matching problems for infinite trees“. In: *Logical Methods in Computer Science* Volume 18, Issue 1 (Feb. 2022). DOI: 10.46298/lmcs-18(1:25)2022 (zitiert auf S. 11, 37).

[CDG+07] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, M. Tommasi. *Tree Automata Techniques and Applications*. 2007. URL: https://people.cs.aau.dk/~srba/courses/FS-07/tata.pdf (zitiert auf S. 14, 27, 31, 33, 34, 37).

[ES77] J. Engelfriet, E. M. Schmidt. „IO and OI. I“. In: *J. Comput. Syst. Sci.* 15 (1977), S. 328–353. DOI: 10.1016/S0022-0000(77)80034-2 (zitiert auf S. 11, 15).

[ES78] J. Engelfriet, E. M. Schmidt. „IO and OI. II“. In: *J. Comput. Syst. Sci.* 16 (1978), S. 67–99. DOI: 10.1016/0022-0000(78)90051-X (zitiert auf S. 11, 15).

[GG13] G. Godoy, O. Giménez. „The HOM Problem is Decidable“. In: *J. ACM* 60.4 (Sep. 2013). ISSN: 0004-5411. DOI: 10.1145/2508028.2501600 (zitiert auf S. 11).

[GK21] M. Gaißert, M. Kufleitner. „The Inclusion Problem for Forest Languages under Substitutions“. In: *CoRR* abs/2106.02571 (2021). URL: https://arxiv.org/abs/2106.02571 (zitiert auf S. 11).

[Kir04] D. Kirsten. „Desert Automata and the Finite Substitution Problem“. In: *STACS 2004*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, S. 305–316. ISBN: 978-3-540-24749-4 (zitiert auf S. 11, 17, 21, 37).

Alle URLs wurden zuletzt am 09. 06. 2022 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift