Institut für Formale Methoden der Informatik

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Das uniforme Wortproblem für Automatengruppen beschränkter Aktivität

Maximilian Kotowsky

Studiengang:	Informatik
Prüfer:	Prof. Dr. Volker Diekert
Betreuer:	Jan Philipp Wächter
begonnen am:	28. April 2020
beendet am:	16. Dezember 2020

Kurzzusammenfassung

Endliche Automaten sind für viele Bereiche der Informatik von enormer Bedeutung. Neben den praktischen Anwendungen dieser Maschinenmodelle sind vor allem die, von ihnen erzeugten Automaten(halb)gruppen und ihre algebraischen sowie algorithmischen Eigenschaften von Interesse für die theoretische Informatik. Die Klassifizierung von Automatengruppen nach ihrer Aktivität durch Said N. Sidki und später durch Bartholdi et al. ermöglicht zusätzlich eine spezifischere Untersuchung dieser Eigenschaften. Die Aktivität ist hierbei das Wachstum der Sprache der Wörter, auf denen ein Automat nichttrivial operiert. In dieser Arbeit gehen wir vor allem auf die Komplexität des (uniformen) Wortproblems von Automatengruppen finitärer und Automatenmonoiden beschränkter Aktivität ein. Weiterhin ordnen wir eine Variante des uniformen Wortproblems, das komprimierte uniforme Wortproblem von Automatengruppen finitärer Aktivität, komplexitätstheoretisch ein. Dazu erweitern wir einen Beweis von Volodymyr V. Nekrashevych und Ievgen V. Bondarenko auf Automatenmonoide und zeigen, dass jedes Automatenmonoid beschränkter Aktivität auch kontrahierend ist. Basierend hierauf ist es möglich, das Wortproblem dieser Automatenmonoide in deterministisch logarithmischem Platz zu entscheiden. Für den Beweis der coNP-Vollständigkeit des uniformen Wortproblems finitärer Automatengruppen nutzen wir die Technik der balancierten Kommutatoren, die ursprünglich von Anatolij I. Mal'cev eingeführt und von David A. Mix Barrington verwendet wurde. Dieses Resultat kann direkt auf Automatengruppen beschränkter Aktivität übertragen werden und zeigt, dass das uniforme Wortproblem dieser Automatenklasse coNP-schwer ist. Schließlich betrachten wir das komprimierte uniforme Wortproblem finitärer Automatengruppen, bei dem das Zustandswort durch ein sogenanntes Straight-Line-Programm, also eine deterministisch kontextfreie Grammatik die nur ein Wort erzeugt, eingegeben wird. Analog zum vorangegangenen Beweis verwenden wir hier die Technik der balancierten Kommutatoren um zu zeigen, dass das genannte Problem PSPACE-vollständig ist.

Inhaltsverzeichnis

1 Einleitung

Rostislav Grigortschuk beschrieb 1984 in [Gri84] erstmal die nach ihm benannte Grigortschuk-Gruppe als spezielle Untergruppe der Automorphismengruppe des unendlichen Binärbaums. Sie ist neben ihrem intermediären, also superpolynomiellem aber subexponentiellem, Wachstum [GP08] eine Burnside-Gruppe, in der die Ordnung eines jeden Elements eine Potenz von Zwei ist [Gri80]. In der ursprünglichen rekursiven Beschreibung erscheint sie zunächst kompliziert, kann jedoch als Automatengruppe vergleichsweise einfach mittels eines endlichen Transduktors repräsentiert werden. Dies sind endliche, vollständige Automaten, bei denen neben der Eingabe in ihrer Übergangsrelation eine Ausgabe definiert wird. Während des Einlesens wird Buchstabe für Buchstabe übersetzt und so ein Ausgabewort erzeugt. Die Zustände eines solchen Automaten induzieren also präfixkompatible und längenerhaltende Funktionen über den Wörtern eines Alphabets und der Abschluss dieser Funktionen unter Komposition bildet die Automatenhalbgruppe beziehungsweise -gruppe.

Neben ihren vielfältigen algebraischen Eigenschaften bieten Automatengruppen einige algorithmische Vorteile. Einerseits können die zugrundeliegenden Automaten als Eingabe von Algorithmen dienen, andererseits weisen sie bezüglich ihrer Entscheidbarkeit und Komplexität hinsichtlich vieler wichtiger Probleme bessere Komplexitätsschranken als Gruppen im Allgemeinen auf. Traditionell werden Gruppen durch eine endliche Menge an Erzeugenden und eine endliche Menge an Relationen über diesen Erzeugenden dargestellt und speziell für die Verwertung durch Algorithmen ist eine solche endliche Darstellung als Eingabe für den Algorithmus notwendig. Viele Gruppen lassen sich jedoch nicht endlich repräsentieren, können also nicht durch eine endliche Menge an Relationen dargestellt werden, wobei sich allerdings manche dieser nicht endlich repräsentierbaren Gruppen durch einen endlichen Automaten beschreiben lassen, also Automatengruppen sind. Soll eine nicht repräsentierbare Gruppe nun durch einen Algorithmus verarbeitet werden, so kann der endliche Automat als Eingabe dienen. Ein Beispiel einer solchen Gruppe ist die in [GŻ01] vorgestellte Lamplighter-Group. Bezüglich der von Max Dehn im Jahr 1910 in [Deh11] formulierten fundamentalen algorithmischen Probleme weisen die Automatengruppen ebenfalls besondere Eigenschaften auf. Sowohl das Wortproblem [Boo59], das Konjugationsproblem [Nov54] als auch das Isomorphieproblem [Adi57] sind für Gruppen im Allgemeinen unentscheidbar, was nach Šunić und Ventura auch für das Konjugations- und das Isomorphieproblem von Automatengruppen gilt [ŠV12]. Für das Wortproblem von Automatengruppen konnte Steinberg in [Ste15] allerdings zeigen, dass es im Allgemeinen in polynomiellem Platz entscheidbar ist. Während also eine obere Platzschranke für die Entscheidbarkeit des Wortproblems von Automatengruppen im Allgemeinen existiert, so existieren doch viele Klassen von Automatengruppen, wie beispielsweise Hanoi Tower

Gruppen [Bon14] oder Automatengruppen mit polynomieller Aktivität [Bon11], die eine bessere obere Platzschranke besitzen. Wächter und Weiß konnten in [WW20] schließlich zeigen, dass eine Automatengruppe mit PSPACE-vollständigem Wortproblem existiert und so eine Behauptung Steinbergs [Ste15, Question 5] beweisen. Hier wurde vor allem die Technik der balancierten Kommutatoren zur Kodierung logischer Funktionen in Gruppen verwendet, die von Barrington in [Bar89] und bereits früher in [Mal62] Verwendung fand. Makanin erwähnt in [Mak85], dass Gurevich diese Technik schon früher verwendete. Mit Hilfe der balancierten Kommutatoren werden wir zeigen, dass das uniforme Wortproblem für Automatengruppen finitärer Aktivität für die Klasse coNP vollständig ist.

Said Sidki führte in [Sid00] die Hierarchie von Automatengruppen bezüglich ihrer Aktivität ein und fand damit eine Möglichkeit, diese Klasse sinnvoll zu strukturieren. Bartholdi, Godin, Klimann und Picantin erweiterten die Hierarchie von Sidki in [BGKP18] auf Automatenhalbgruppen, indem sie den Begriff der Aktivität neu definierten. Sidki ging in seiner Arbeit unter anderem auf die geometrische Charakterisierung der Aktivität ein, die in dieser Arbeit ebenfalls zu Rate gezogen wird. Viele algorithmische Probleme der Gruppentheorie sind für Klassen der Aktivitätshierarchie nach Sidki entscheidbar. So konnte beispielsweise Bondarenko in [Bon11] zeigen, dass das Wortproblem von Automatengruppen mit polynomiell beschränkter Aktivität in polylogarithmischem Platz entschieden werden kann. Für die Klasse der Automatengruppen mit beschränkter Aktivität, zu denen auch die Grigortschuk-Gruppe gehört, sind außerdem viele weitere algorithmische Probleme entscheidbar, wie beispielsweise für das Endlichkeitsproblem in [BW19] gezeigt werden konnte. Nekrashevych und Bondarenko bewiesen in [Nek05], dass jede Automatengruppe beschränkter Aktivität kontrahierend ist und Bartholdi, Figelius, Lohrey und Weiß konnten darauf aufbauend in [BFLW20] zeigen, dass das Wortproblem solcher Automatengruppen in deterministisch logarithmischem Platz entschieden werden kann. Wir werden dieses Konzept in Kapitel 3 auf Automatenmonoide erweitern und zeigen, dass das Wortproblem von Automatenmonoiden beschränkter Aktivität ebenfalls in deterministisch logarithmischem Platz entscheidbar ist. Bondarenko, Bondarenko, Sidki und Zapata zeigten in [BBSZ13] schließlich, dass eine verwandte Fragestellung des Konjugationsproblems für Automatengruppen beschränkter Aktivität entscheidbar ist. Eine weitere algorithmische Fragestellung ist das komprimierte Wortproblem, bei dem das Eingabewort durch ein sogenanntes Straight-Line-Programm eingegeben wird, also eine deterministisch kontextfreie Grammatik, die nur ein Wort erzeugt. Wie Markus Lohrey in [Loh14] ausführt, kann das Wortproblem endlich erzeugter Untergruppen der Automorphismengruppe $\mathrm{Aut}(G)$ einer endlich erzeugten Gruppe G effizient durch das komprimierte Wortproblem eben jener Gruppe G gelöst werden. In dieser Arbeit wird das uniforme komprimierte Wortproblem finitärer Automatengruppen speziell hinsichtlich seiner Komplexität untersucht und es kann gezeigt werden, dass es PSPACE-vollständig ist.

2 Grundlagen

Die Erörterung der eigentlichen Fragestellung nach den Wortproblemen von Automatengruppen beschränkter Aktivität, finitärer Aktivität und von Automatenmonoiden erfordert die Kenntnis einiger Grundlagen. Während einfache Grundlagen der Automatentheorie vorausgesetzt werden, werden im folgenden Kapitel einige Begriffe definiert und erläutert, sowie einige Lemmata bewiesen, die das Fundament dieser Arbeit legen.

In dieser Arbeit gilt $0 \in \mathbb{N}$. Für zwei Mengen A, B schreiben wir B^A für die Menge aller Funktionen von A nach B. Eine Funktion $f : A \to B$ eingeschränkt auf eine einen Teildefinitionsbereich $A' \subseteq A$ wird mit $f_{|A'} : A' \to B$ notiert. Die Vereinigung disjunkter Mengen A und B notieren wir mit $A \uplus B$. Das neutrale Element von Monoiden und Gruppen notieren wir mit $\mathbb{1}$. Für eine Halbgruppe H und eine Menge $E \subseteq H$ gilt außerdem $\langle E \rangle = \{e_1 \cdots e_n \mid e_i \in E, n \in \mathbb{N}\}$. Für eine Gruppe G mit $E \subseteq G$ gilt analog $\langle E \rangle = \{e_1 \cdots e_n \mid e_i \in E \cup E^{-1}, n \in \mathbb{N}\}$ wobei $E^{-1} = \{e^{-1} \mid e \in E, e^{-1}e = ee^{-1} = \mathbb{1}\}$ die Menge der inversen Elemente zu Elementen aus E ist.

2.1 Alphabet und Sternabschluss

Ein *Alphabet* ist eine endliche, nicht leere Menge, deren Elemente wir *Buchstaben* nennen. Mit Σ^* bezeichnen wir den *Sternabschluss* des Alphabets Σ, also die Menge aller Sequenzen beliebiger, endlicher Länge von Buchstaben aus Σ. Insbesondere ist auch die leere Sequenz ε Teil dieser Menge. Weitere nützliche Notationen sind $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ für die Menge aller Sequenzen mit Länge mindestens 1, $\Sigma^{\geq n} = \bigcup_{i=n}^{\infty} \Sigma^i$ für die Menge aller Sequenzen mit Länge mindestens n und $\Sigma^{\leq n} = \bigcup_{i=0}^{n} \Sigma^i$ für die Menge aller Sequenzen mit Länge maximal n.

2.2 Transduktoren und deren Darstellung

Endliche *Transduktoren* sind spezielle endliche Automaten, die auf eben diese Sternabschlüsse Σ^* wirken.

Definition 2.1. Ein endlicher Transduktor $\mathcal{T} = (Q, \Sigma, \delta)$ ist ein Tripel, bestehend aus einer endlichen, nicht leeren Zustandsmenge Q, einem Alphabet Σ sowie einer Übergangsrelation $\delta \subseteq Q \times \Sigma \times \Sigma \times Q$.

Elemente aus Σ^* bezeichnen wir im Allgemeinen als *Wörter*, Elemente aus Q^* als *Zustandswörter*. Weiterhin schreiben wir $p \xrightarrow{a/b} q$ für eine Übergangsrelation (p, a, b, q). Automaten werden aufgrund bestimmter grundlegender Eigenschaften klassifiziert, die für den späteren Verlauf der Arbeit wichtig sind.

Determinismus Für alle Zustände gilt, dass pro gelesenem Buchstaben höchstens ein möglicher Folgezustand existiert.

$$\forall p \in Q, a \in \Sigma : |\{(b, q) \mid p \xrightarrow{a/b} q \in \delta\}| \leq 1$$

Vollständigkeit Für alle Zustände gilt, dass für jeden Buchstaben mindestens ein möglicher Folgezustand existiert.

$$\forall p \in Q, a \in \Sigma : |\{q \mid p \xrightarrow{a/b} q \in \delta\}| \geq 1$$

Invertierbarkeit Vertauscht man Eingabe- und Ausgabebuchstaben, also die beiden mittleren Komponenten der Übergangsrelationen, so muss der Transduktor weiterhin deterministisch und vollständig sein.

$$\forall p \in Q, b \in \Sigma : |\{(b, q) \mid p \xrightarrow{a/b} q \in \delta\}| \leq 1$$

Als Konvention sprechen wir im weiteren Verlauf der Arbeit von einem *$\mathcal{S}$-Automaten*, wenn wir einen endlichen, vollständigen, deterministischen Transduktor meinen.

Definition 2.2. Für ein $\mathcal{S}$-Automaten $\mathcal{T} = (Q, \Sigma, \delta)$ ist $\mathcal{T}^{-1} = (Q^{-1}, \Sigma, \delta^{-1})$ der inverse Automate zu $\mathcal{T}$. Dabei ist

$$Q^{-1} = \{q^{-1} \mid q \in Q\} \text{ eine disjunkte Kopie von } Q$$
$$\delta^{-1} \subseteq Q^{-1} \times \Sigma \times \Sigma \times Q^{-1} \text{ die Übergangsrelation mit}$$
$$\delta^{-1} = \{p^{-1} \xrightarrow{b/a} q^{-1} \mid p \xrightarrow{a/b} q \in \delta\}$$

Wir definieren außerdem $Q^{\pm} = Q \cup Q^{-1}$. Ist der inverse $\mathcal{S}$-Automat $\mathcal{T}^{-1}$ deterministisch und vollständig, so ist $\mathcal{T}$ invertierbar und wir nennen $\mathcal{T}$ einen $\mathcal{G}$-Automat. Die Automaten lassen sich wie allgemein üblich mit Hilfe sogenannter Moore-Diagramme grafisch darstellen, wie in Abbildung 2.1 veranschaulicht wird.

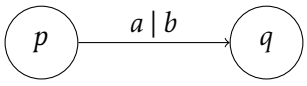
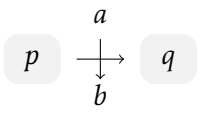


Abbildung 2.1: Graphische Darstellung von $p \xrightarrow{a/b} q$

2.3 Kreuzdiagramme und Zustandsfunktionen

Die Übergangsfunktion eines solchen Automaten kann durch sogenannte Kreuzdiagramme dargestellt werden. Für $p \xrightarrow{a/b} q$ schreiben wir



Durch die Komposition solcher Kreuzdiagramme können wir einerseits das Wirken eines Zustands auf ein Wort aus Σ^*, andererseits auch das Wirken eines Zustandswortes aus Q^* auf Buchstaben aus Σ darstellen (und als direkte Schlussfolgerung dann natürlich auf Wörter beziehungsweise Zustandswörter). Dies wird in Abbildung 2.2 veranschaulicht, wobei die einzelnen Teildiagramme wiederum Übergangsrelationen aus δ repräsentieren.

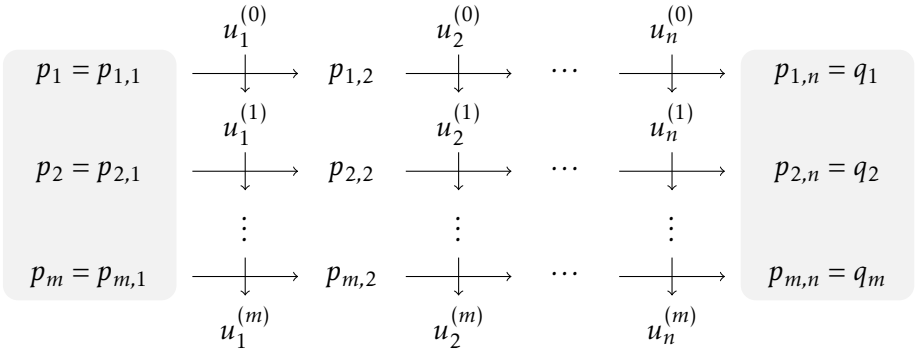


Abbildung 2.2: Komposition mehrerer Kreuzdiagramme

Sind die „internen“ Zustände im Kreuzdiagramm nicht von Interesse, so können diese weggelassen werden. Um Kreuzdiagramme trotz der Komposition kompakt darstellen zu können, können diese zusammengefasst werden. Setzen wir beispielsweise $u = u_1 \cdots u_n$ und $p = p_m \cdots p_1$, dann lässt sich das Beispiel in Abbildung 2.2 wie in Abbildung 2.3 gezeigt darstellen.

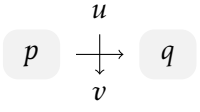
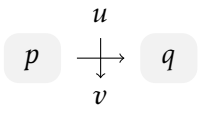
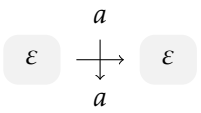


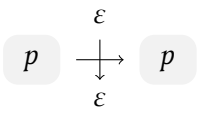
Abbildung 2.3: Komptakschreibweise eines zusammengesetzten Kreuzdiagramms

Die, durch die vorangegangenen Kreuzdiagramme dargestellte Wirkung von Zustandswörtern auf Wörter (senkrechte Pfeile) und die Dualwirkung von Wörtern auf Zustandswörter (senkrechte Pfeile) lassen sich formal als Funktionen $\circ$ und $\cdot$ definieren. Da der $\mathcal{S}$-Automat deterministisch und vollständig ist, gibt es für ein $q \in Q^+$ und $u \in \Sigma^+$ genau ein Kreuzdiagramm mit



für ein q, v. Dann definieren wir $p \circ u = v$ und $p \cdot u = q$ für Funktionen $\circ : Q^* \times \Sigma^* \to \Sigma^*$ und $\cdot : Q^* \times \Sigma^* \to Q^*$. Für den Fall, dass einer der beiden Operanden das leere Wort ε ist, gilt





Für ε aus der Menge Q^* schreiben wir im weiteren Verlauf der Arbeit $\mathbb{1}$. Um die Zustände sinnvoll mit ihrer Wirkung auf Σ^* identifizieren zu können, definieren wir zunächst die folgende Abbildung.

$$\pi : Q^* \to (\Sigma^*)^{\Sigma^*}, p \mapsto (p\circ) \tag{2.1}$$

Es ist also $\pi(p) = (p\circ)$ eine Abbildung mit

$$(p\circ) : \Sigma^* \to \Sigma^*, u \mapsto p \circ u.$$

Eine Funktion $(p\circ)$ bildet also Wörter aus Σ^* auf andere Wörter aus Σ^* ab, wobei die Abbildung Wörter so transformiert, wie der zugehörige Transduktor gestartet im Zustand p. Es ist leicht zu sehen, dass π ein Homomorphismus ist. Im Verlauf der Arbeit werden wir nicht $\pi(p)$ für die, vom Zustand p induzierte Funktion, sondern direkt $(p\circ)$ schreiben. Die Komposition zweier solcher Funktionen ergibt sich direkt aus der Homomorphie-Eigenschaft von π und es gilt

$$(p\circ)(q\circ) = (pq\circ)$$
$$(pq\circ)(u) = (p\circ)(q \circ u)$$

als sequenzielle Wirkung der Zustände p und q auf das Wort u. Man sieht leicht, dass eine Funktion $\pi(p^{-1})$ mit p^{-1} aus Q^{-1}, dem Gegenstück zu p aus Q, die Umkehrfunktion zu $(p\circ)$ ist.

2.4 Automatenstrukturen: Automatengruppen und -monoide

Transduktoren beschreiben, wie bereits gesehen und definiert, die Wirkung von Zuständen auf Wörter und die Dualwirkung von Wörtern auf Zustände. Dies ermöglicht es, algebraische Strukturen auf endliche Transduktoren zurückführen zu können. Im Folgenden werden wir solche Strukturen betrachten und sehen, wie diese aus endlichen Transduktoren erwachsen.

Definition 2.3. Sei $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{S}$-Automat. Sei $\equiv_{\mathcal{T}} \subseteq Q^* \times Q^*$ die Relation mit

$$p \equiv_{\mathcal{T}} q \iff \forall u \in \Sigma^* : p \circ u = q \circ u \text{ für } q, p \in Q^*$$

Da die Relation $\equiv_{\mathcal{T}}$, wie man leicht sieht, die Eigenschaften

$$\begin{aligned} \forall p \in Q^* &: p \equiv_{\mathcal{T}} p \\ \forall p, q, t \in Q^* &: (p \equiv_{\mathcal{T}} q \wedge q \equiv_{\mathcal{T}} r) \implies p \equiv_{\mathcal{T}} r \\ \forall p, q \in Q^* &: p \equiv_{\mathcal{T}} q \implies q \equiv_{\mathcal{T}} p \end{aligned}$$

erfüllt, ist sie eine Äquivalenzrelation. Im Folgenden werden wir außerdem zeigen, dass $\equiv_{\mathcal{T}}$ eine Kongruenzrelation ist. Seien p, p', q, q' aus Q^* beliebig mit $p \equiv_{\mathcal{T}} p'$ und $q \equiv_{\mathcal{T}} q'$. Sei weiterhin für ein beliebiges w aus Σ^* $q \circ w = q' \circ w = v$ die Ausgabe von q beziehungsweise q' mit Eingabe w. Dann gilt

$$(pq) \circ w = p \circ (q \circ w) = p \circ v = p' \circ v = p' \circ (q' \circ w) = (p'q') \circ w$$

also $pq \equiv_{\mathcal{T}} p'q'$ und $\equiv_{\mathcal{T}}$ ist eine Kongruenzrelation.

Fakt 2.4. Sei $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{S}$-Automat. Dann ist $\mathscr{M}(\mathcal{T}) = Q^*/\equiv_{\mathcal{T}}$ ein Monoid. Ein Automatenmonoid $\mathscr{M}(\mathcal{T})$ ist ein Monoid, das auf diese Weise entsteht.

Das neutrale Element $[\varepsilon]_{\equiv_{\mathcal{T}}}$ eines solchen Monoids bezeichnen wir im Folgenden mit $\mathbb{1}$. Die Verknüpfung von $Q^*/\equiv_{\mathcal{T}}$ ist wie üblich definiert als

$$[p]_{\equiv_{\mathcal{T}}}[q]_{\equiv_{\mathcal{T}}} = [pq]_{\equiv_{\mathcal{T}}}$$

Eine andere Art der Herangehensweise ist die Komposition der Endomorphismen von Σ^*, auf die wir im vorherigen Abschnitt die Zustände von $\mathcal{T}$ mit Hilfe der Funktion in Gleichung (2.1) abgebildet haben.

Definition 2.5. Sei $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{S}$-Automat. Wir definieren

$$Q^*\circ = \{(p\circ) \mid p \in Q^*\}.$$

Dies ist der Abschluss aller Funktionen $(p\circ)$ mit $p \in Q$ unter Komposition.

Proposition 2.6. Sei $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{S}$-Automat. Dann ist $Q^*\circ$ ein Monoid und die Monoide $Q^*\circ$ und $Q*/\equiv_{\mathcal{T}}$ sind zueinander isomorph.

Beweis. Sei $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{S}$-Automat und

$$\begin{aligned}\phi :& Q^*\circ \xrightarrow{\sim} Q^*/\equiv_{\mathcal{T}}\\ & (p\circ) \mapsto [p]_{\equiv_{\mathcal{T}}}\end{aligned}$$

eine Abbildung zwischen den Monoiden. Dann gilt für alle p, q aus Q^*:

$$\begin{aligned}
&(p\circ) = (q\circ) \implies \forall u \in \Sigma^* : p \circ u = q \circ u \implies [p]_{\equiv_{\mathcal{T}}} = [q]_{\equiv_{\mathcal{T}}}. && \text{(Wohldefiniertheit)}\\
&\phi((\varepsilon\circ)) = [\varepsilon]_{\equiv_{\mathcal{T}}} = \{q \in Q^* \mid \forall u \in \Sigma^* : q \circ u = u\}. && \text{(Homomorphie)}\\
&\phi((p\circ)(q\circ)) = \phi(pq\circ) = [pq]_{\equiv_{\mathcal{T}}} = [p]_{\equiv_{\mathcal{T}}}[q]_{\equiv_{\mathcal{T}}}. && \text{(Homomorphie)}\\
&[p]_{\equiv_{\mathcal{T}}} = [q]_{\equiv_{\mathcal{T}}} \implies \forall u \in \Sigma^* : p \circ u = q \circ u \implies (p\circ) = (q\circ). && \text{(Injektivität)}\\
&[p]_{\equiv_{\mathcal{T}}} \ni Q^*/\equiv_{\mathcal{T}} \implies \phi((q\circ)) = [p]_{\equiv_{\mathcal{T}}}. && \text{(Surjektivität)}
\end{aligned}$$

Also ist ϕ ein Isomorphismus und $Q^*\circ$ und $Q^*/\equiv_{\mathcal{T}}$ sind zueinander isomorph. □

Im weiteren Verlauf der Arbeit werden wir einerseits über Zustandswörter $p = p_1 \cdots p_n$ aus Q^*, andererseits über Elemente eines Automatenmonoids $\mathscr{M}(\mathcal{T})$ sprechen. Die Elemente von $\mathscr{M}(\mathcal{T})$ werden wir sowohl anhand ihrer Äquivalenzklassen $[p]_{\equiv_{\mathcal{T}}}$ als auch durch die Komposition der Funktionen $(p_1\circ)\cdots(p_n\circ) = (p\circ)$ darstellen. Es wird jeweils die sinnvollste Definition für den konkreten Fall genutzt. Neben den Automatenmonoiden sind außerdem noch die Automatengruppen von Interesse.

Proposition 2.7. Sei $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{G}$-Automat. Dann ist $(Q^{\pm})^*\circ$ eine Gruppe G. Eine Automatengruppe $\mathscr{G}(\mathcal{T})$ ist eine Gruppe, die auf diese Weise entsteht.

Beweis. Sei $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{G}$-Automat. Dann sind aufgrund der Invertierbarkeit alle Funktionen $(p\circ)$ Bijektionen und folglich $(Q^{\pm})^*\circ$ eine Gruppe. □

2.5 Aktivität und Wachstum

Die in dieser Arbeit untersuchten Automatengruppen und Automatenmonoide sind solche finitärer und beschränkter Aktivität. Der Begriff der Aktivität wurde urspünglich von Said N. Sidki in [Sid00] eingeführt und von Bartholdi, Godin, Klimann und Picatin in [BGKP18] auf Automatenhalbgruppen erweitert. Betrachtet man die Menge der Wörter einer bestimmten Länge, auf denen ein bestimmter Zustand des Automaten nichttrivial operiert, das Ausgabewort sich also an mindestens einer Stelle vom Eingabewort unterscheidet, so ist die Aktivität als Wachstum dieser Menge bei ansteigender Länge der Wörter zu verstehen.

Definition 2.8 (Aktivität)**.** Sei $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{S}$-Automat. Für $q \in Q^*$ ist

$$A_q(n) = \{w \mid \exists v \in \Sigma^n : q \circ v = w, q \cdot v \not\equiv_{\mathcal{T}} \mathbb{1}\}$$

die Menge der aktiven Wörter der Länge n und $\alpha_q(n) = \left|A_q(n)\right|$ die Aktivität von q in n. Die Aktivität $\alpha_T(n)$ eines Automaten $\mathcal{T}$ ist die maximale Aktivität aller seiner Zustände.

Bemerkung 2.9. Die Definition 2.8 ist für $\mathcal{G}$-Automaten und somit Automatengruppen äquivalent zur ursprünglichen Definition von Sidki in [Sid00].

Wie Bartholdi, Godin, Klimann und Picantin in [BGKP18, Lemma 3.1.] zeigten, gilt für die Aktivität zweier Zustände $p, q \in Q$, dass

$$\alpha_{pq}(n) \leq \alpha_p(n) + \alpha_q(n). \tag{2.2}$$

Definition 2.10 (Beschränkte Aktivität)**.** Sei $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{S}$-Automat mit Aktivität $\alpha_{\mathcal{T}}(n)$. Wenn eine Zahl K existiert, sodass $\alpha_{\mathcal{T}}(n)$ für alle n kleiner gleich K ist, dann nennen wir $\mathcal{T}$ beschränkt mit Aktivitätskonstante K.

Definition 2.11 (Finitäre Aktivität)**.** Sei $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{S}$-Automat mit Aktivität $\alpha_{\mathcal{T}}(n)$. Wenn für alle Zustände $q \in Q$ und alle n größer einer Zahl n_0 gilt, dass $\alpha_q(n) = 0$, dann ist q finitär. Gilt dies für alle Zustände des Automaten, so ist $\mathcal{T}$ ebenfalls finitär.

Definition 2.12. Ein *Pfad* der Länge n in einem $\mathcal{S}$-Automaten $\mathcal{T} = (Q, \Sigma, \delta)$ ist eine Folge $p_0, a_1, p_1, \ldots, p_{n-1}, a_n, p_n$ mit $p_i \in Q$ und $a_i \in \Sigma$, wobei für alle $j \in \{1, \ldots, n\}$ mindestens ein Zustandsübergang $p_{j-1} \xrightarrow{b/a_j} p_j$ für beliebiges $b \in \Sigma$ existieren muss. Wir bezeichnen p_0 als Startzustand und p_n als Endzustand von P. Jede Teilfolge von P ist wieder ein Pfad P' und wir sagen P enthält P'.

Man beachte, dass der Pfad nur von den besuchten Zuständen und der Ausgabe abhängt. Das bedeutet insbesondere, dass Zustandsübergänge mit verschiedener Eingabe und gleicher Ausgabe zu einem Pfad zusammengefasst werden.

Definition 2.13. Ein *Zyklus* C in $\mathcal{T}$ ist ein Pfad, sodass Startzustand und Endzustand des Pfades identisch sind, also $p_0 = p_n$.

Lemma 2.14 (Kreislemma)**.** Sei $\mathcal{T}$ ein $\mathcal{S}$-Automat beschränkter Aktivität. Dann existieren in $\mathcal{T}$ nur Pfade, die höchstens einen Zyklus enthalten (außer dem Zyklus des trivialen Zustands).

Beweis. Angenommen es existiert ein $\mathcal{S}$-Automat $\mathcal{T} = (Q, \Sigma, \delta)$, der beschränkt durch die Konstante K ist und einen Pfad mit Endzustand $\mathbb{1} \in Q$ besitzt, der zwei Zyklen enthält. Seien ℓ_1, ℓ_2 die Längen dieser Zyklen und z_1, z_2, z_3 die Länge der „Zwischenstücke“ auf dem Pfad. Wir setzen $z = z_1 + z_2 + z_3$ und $\ell = \mathrm{kgV}(\ell_1, \ell_2)$. Betrachte nun Wörter w aus Σ^* mit mindestens Länge $n \geq z$ und ℓ teilt $n - z$. Dann ist die Anzahl der aktiven Wörter auf diesem Pfad mindestens

$$\binom{c+1}{1} = c + 1 \text{ mit } c = \frac{n-z}{\ell}.$$

Dies gilt, da wir die Anzahl an Durchgängen durch eine Schleife beliebig wählen können, also c Schleifendurchgänge auf 2 Schleifen aufteilen müssen. Für $c = K$ gilt also $\alpha_q(n) \geq c+1 = K+1$, was größer als K ist. Somit kann $\mathcal{T}$ nicht durch die Konstante K beschränkt gewesen sein. □

Fakt 2.15. Es gilt ebenfalls, dass wenn in $\mathcal{T}$ nur Pfade existieren, die maximal einem Zyklus enthalten, dann ist $\mathcal{T}$ maximal von beschränkter Aktivität.

Fakt 2.16. Die Pfade in finitären Automaten dürfen keine Zyklen enthalten (außer den Zyklus des trivialen Zustands).

Da die Aktivität subadditiv ist, folgt aus Gleichung (2.2) direkt, dass wenn ein $\mathcal{S}$-Automat $\mathcal{T}$ beschränkt ist auch $\mathscr{M}(\mathcal{T})$ beschränkt ist. Das Gleiche gilt für finitäre $\mathcal{S}$-Automaten. Nun werden wir noch beweisen, dass dies auch für beschränkte (und finitäre) $\mathcal{G}$-Automaten und $\mathscr{G}(\mathcal{T})$ gilt.

Proposition 2.17. Sei $\mathcal{T}$ ein $\mathcal{G}$-Automat und beschränkt. Dann ist auch $\mathcal{T}^{-1}$ beschränkt.

Beweis. Sei $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{G}$-Automat und beschränkt. Nach Lemma 2.14 existiert auf jedem Pfad in $\mathcal{T}$ nur maximal ein Kreis. Da $\mathcal{T}$ ein $\mathcal{G}$-Automat und somit invertierbar ist, die Funktionen $(p\circ)$ für beliebige $p \in Q$ also Bijektionen sind, ändern sich durch das Invertieren von Ein- und Ausgabe die Pfade und somit auch die Anzahl der Zyklen auf einem Pfad nicht. Die Anzahl der Zyklen bleibt auf jedem Pfad gleich und nach Fakt 2.15 ist $\mathcal{T}^{-1}$ ebenfalls beschränkt. □

Fakt 2.18. Aus dem Beweis zu Proposition 2.17 lässt sich ableiten, dass für einen $\mathcal{G}$-Automaten $\mathcal{T}$ finitärer Aktivität ebenfalls gilt, dass $\mathcal{T}^{-1}$ finitär ist.

Proposition 2.19. Für einen $\mathcal{G}$-Automaten $\mathcal{T}$ gilt, dass $\mathscr{G}(\mathcal{T}) = \mathscr{M}(\mathcal{T} \cup \mathcal{T}^{-1})$.

Beweis. Sei $\mathcal{T}$ ein $\mathcal{G}$-Automat. Dann ist $\mathcal{G}(\mathcal{T})$ definiert als $(Q^{\pm})^*\circ = \{(p\circ) \mid p \in (Q \cup Q^{-1})^*\}$. Da $\mathcal{T}$ ein $\mathcal{G}$-Automat ist, sind die Funktionen $(p\circ)$ mit $p \in Q$ Bijektionen. Da die Funktionen $(p^{-1}\circ)$ mit $p^{-1} \in Q^{-1}$ die inversen Funktionen zu den $(p\circ)$ sind, gilt außerdem dass $\pi(pp^{-1}) = (pp^{-1}\circ) = (\epsilon\circ) = (p^{-1}p\circ) = \pi(p^{-1}p)$. Also folgt, dass für Zustandswörter $q \in (Q^{\pm})^*$ ein Zustandswort $q' \in Q^* \cup (Q^{-1})^*$ existiert, sodass $(q\circ) = (q'\circ)$ ist. Daraus folgt, dass $(Q^{\pm})^*\circ$ als $\{(p\circ) \mid p \in Q^* \cup (Q^{-1})^*\}$ geschrieben werden kann und es gilt $\mathcal{G}(\mathcal{T}) = \mathscr{M}(\mathcal{T} \cup \mathcal{T}^{-1})$. □

Aus Proposition 2.17 folgt mit Proposition 2.19 und Gleichung (2.2), dass wenn ein $\mathcal{G}$-Automat $\mathcal{T}$ beschränkt ist, $\mathcal{G}(\mathcal{T})$ ebenfalls beschränkt ist. Für finitäre $\mathcal{G}$-Automaten analog. In dieser Arbeit werden wir uns speziell mit finitärer und beschränkter Aktivität beschäftigen.

2.6 Die *Adding Machine* als Beispiel

Im Folgenden betrachtet wir ein Beispiel für eine Automatengruppe. Sei $\mathcal{T}$ ein $\mathcal{S}$-Automat (*Adding Machine* genannt):

$$\mathcal{T} = (\{\mathbb{1}, q\}, \{0,1\}, \delta) \text{ mit } \delta = \{(q,1,0,q), (q,0,1,\mathbb{1}), (\mathbb{1},0,0,\mathbb{1}), (\mathbb{1},1,1,\mathbb{1})\}.$$

Man sieht leicht, dass $\mathcal{T}$ ein $\mathcal{G}$-Automat ist. Wir stellen $\mathcal{T}$ sowie $\mathcal{T}^{-1}$ im Folgenden graphisch dar.

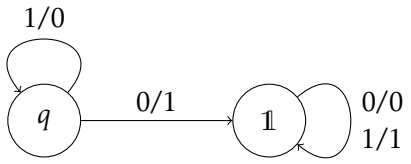


Abbildung 2.4: Adding Machine

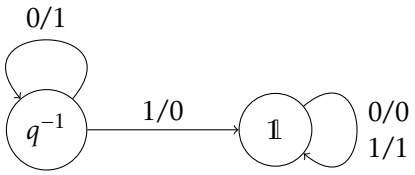


Abbildung 2.5: Inverse Adding Machine

Interpretiert man Wörter aus $\{0,1\}^*$ als Zahlen in Binärdarstellung, wobei die Zahlen von rechts nach links gelesen werden, so inkrementiert der Zustand q diese um den Wert 1. Wir verdeutlichen dies anhand zweier Beispiele.

$$q\circ 000 = (q\circ 0)00 = 1(\mathbb{1}\circ 0)0 = 10(\mathbb{1}\circ 0) = 100$$
$$q\circ 110 = (q\circ 1)10 = 0(q\circ 1)0 = 00(q\circ 0) = 001$$

Der Zustand $\mathbb{1}$ verändert die Eingabe offensichtlich nicht. Mögliche Verknüpfungen von Zuständen des Transduktors sind $\mathbb{1}\mathbb{1}$, $\mathbb{1}q$, $q\mathbb{1}$, qq, folgend als Kreuzdiagramme mit Eingabewort 110 dargestellt.

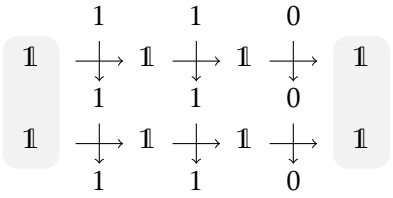


Abbildung 2.6: $\mathbb{1}\mathbb{1}\circ 110$

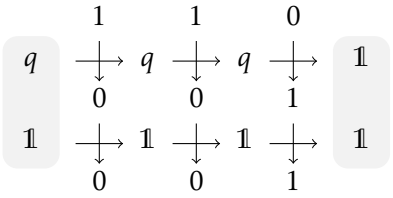


Abbildung 2.8: $\mathbb{1}q\circ 110$

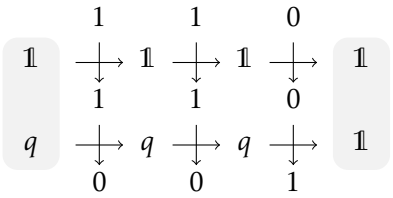


Abbildung 2.7: $q\mathbb{1}\circ 110$

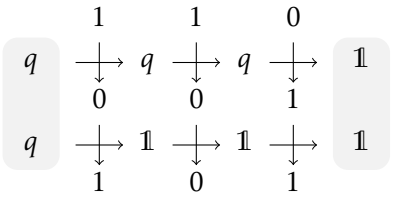


Abbildung 2.9: $qq\circ 110$

Aus den Kreuzdiagrammen in Abbildung 2.6 bis Abbildung 2.9 wird klar, dass

$$(q\circ)(\mathbb{1}\circ) = (\mathbb{1}\circ)(q\circ) = (q\circ)$$
$$(\mathbb{1}\circ)(\mathbb{1}\circ) = (\mathbb{1}\circ)$$

gilt. Es ist also lediglich die Verknüpfung $(q\circ)(q\circ) = (q^2\circ)$ interessant. Man sieht, dass das Gruppenelement $(qq\circ) \in \mathcal{G}(\mathcal{T})$ ein Eingabewort um eins und $(q^n\circ)$ ein Eingabewort um den Wert n inkrementiert. Die Funktionen $(q^{-m}\circ)$ (vgl. Abbildung 2.5) dekrementieren ein Eingabewort dementsprechend um den Wert m. Die Funktion $\phi : (Q^{\pm})^*\circ \to \mathbb{Z}, (q^n\circ) \mapsto n$ liefert uns einen Isomorphismus in die additive Gruppe der ganzen Zahlen $\mathbb{Z}$ und $(\mathbb{Z}, +)$ ist also eine Automatengruppe.

3 Das Wortproblem von Automatenmonoiden beschränkter Aktivität

Wie Nekrashevych und Bondarenko in [Nek05, Theorem 3.8.8] zeigen, sind selbstähnliche, endlich erzeugte Automatengruppen beschränkter Aktivität[1] kontrahierend unter der Dualwirkung von Σ^*. Dies erweist sich als nützliches Werkzeug bei der Lösung des Wortproblems solcher Gruppen. Im Folgenden wollen wir beweisen, dass Automatenmonoide beschränkter Aktivität ebenfalls kontrahierend sind.

3.1 Grundlagen und Begriffe

Im Folgenden werden zunächst die verwendeten Begriffe definiert und anhand von einführenden Beispielen erläutert. Anschließend wird der eben dargestellte Sachverhalt formalisiert und bewiesen.

Automatenmonoide und Aktivität

Volodymyr Nekrashevych definiert in seiner Monografie [Nek05] eine kontrahierende Wirkung einer Gruppe G auf eine Menge X. Dieser Begriff lässt sich ebenfalls Automatenmonoide definieren.

Definition 3.1. Sei $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{S}$-Automat. Wir nennen $\mathscr{M}(\mathcal{T})$ kontrahierend, wenn eine endliche Teilmenge $\mathcal{N} \subseteq \mathscr{M}(\mathcal{T})$ existiert, sodass es für alle $[p]_{\equiv_\mathcal{T}} \in \mathscr{M}(\mathcal{T})$ ein $k \in \mathbb{N}$ gibt, sodass für alle $w \in \Sigma^{\geq k}$ gilt, dass $[p]_{\equiv_\mathcal{T}} \cdot w \in \mathcal{N}$ ist. Wir nennen $\mathcal{N}$ den Nucleus.

Mit Hilfe dieser Definitionen können wir nun das folgende Theorem aufstellen, das wir im Verlauf von Abschnitt 3.2 beweisen werden.

Satz 3.2. *Sei $\mathscr{M}(\mathcal{T})$ ein Automatenmonoid mit beschränkter Aktivität. Dann ist $\mathscr{M}(\mathcal{T})$ kontrahierend.*

[1] Untergruppen der Gruppe $\mathcal{B}_0$, also der Gruppe der Automaten, deren Aktivität durch ein Polynom nullten Grades beschränkt ist, siehe Abschnitt 3.8.2 in [Nek05].

3.2 Automatenmonoide beschränkter Aktivität sind kontrahierend

Sei im Folgenden $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{S}$-Automat mit beschränkter Aktivität und $\mathscr{M}(\mathcal{T})$ das von $\mathcal{T}$ erzeugte Automatenmonoid. Man beachte, dass im Gegensatz zu den $\mathcal{G}$-Automaten die Automaten hier nicht invertierbar sein müssen. Es ist also für Zustandswörter $q \in Q^*$ möglich, dass verschiedene $w \in \Sigma^*$ zur selben Ausgabe v führen und deshalb verschiedene Zustandswörter unter der Dualwirkung bei einer Ausgabe v resultieren können. Um diese Menge der unter der Dualwirkung resultierenden Zustandswörter besser beschreiben zu können, definieren wir den folgenden Operator.

Definition 3.3. Seien $p \in Q^*$, $v \in \Sigma^*$ und $U \subseteq \Sigma^*$ mit $U = \{u \mid p \cdot u = v\}$. Dann ist

$$p \xrightarrow[v]{U} p\backslash v$$

$$p\backslash v = \{q \in Q^* \mid \exists u \in \Sigma^* : p \circ u = v, p \cdot u = q\}$$

die Menge aller $q \in Q^*$, in denen ein Lauf mit Ausgabe v, gestartet in p, enden kann.

Seien weiterhin $C, F, P \subseteq Q$ Zustandsmengen mit:

- $C = \{q \in Q \mid \exists w \in \Sigma^+ : q\backslash w \ni q\}$ die Menge aller Zustände, die auf Zyklen liegen.
- $F = \{p \in Q \mid \exists n_0 \in \mathbb{N} : \forall w \in \Sigma^{\geq n_0} : p \cdot w \equiv_{\mathcal{T}} \mathbb{1}\}$ die Menge der finitären Zustände.
- $P = Q \setminus (C \cup F)$ die Menge der Zustände, die vor einem Zyklus liegen.
- $(C_p)_{p \in Q} = \{q \in Q \mid \exists w \in \Sigma^* : p\backslash w \ni q,\ q\backslash w \ni p\}$ die Menge der Zustände, die auf einem Zyklus liegen.

Sei weiterhin n_1 ein Vielfaches aller Zyklenlängen in der graphischen Darstellung von $\mathcal{T}$, also der Mächtigkeit aller C_p, mit n_1 ist größer als die Mächtigkeit der Mengen P und F. Wir betrachten im Folgenden Elemente des Automatenmonoids $\mathscr{M}(\mathcal{T})$ sowie Zustandswörter aus Q^*, wobei wir je nach Zweck zwischen den Notationen wechseln und uns in Erinnerung rufen, dass sich das Monoidelement $[m]_{\equiv_{\mathcal{T}}}$ und das Zustandswort m unter der Dualwirkung gleich verhalten und identisch auf Σ^* operieren, weswegen der Wechsel problemlos möglich ist. Außerdem ist die Wahl eines Zustandswortes als Vertreter der jeweiligen Äquivalenzklasse beliebig, da die Vertreter zwar verschieden repräsentiert sein können, jedoch die gleiche Funktion induzieren und sich deshalb auch identisch verhalten. Sei zunächst

$$\mathcal{N}_1 = \{[m]_{\equiv_{\mathcal{T}}} \in \mathscr{M}(\mathcal{T}) \mid (\exists v \in \Sigma^{n_1} : m\backslash v \ni m) \wedge (\forall v' \in \Sigma^{n_1}, v' \neq v : m\backslash v' \subseteq \langle F \rangle)\}$$

die Menge der Monoidelemente m, die mit genau einer Ausgabe w wieder in sich selbst überführt werden und sonst durch finitäre Zustände erzeugt werden können.

Bemerkung 3.4. Offensichtlich gilt für beliebige $i \in \mathbb{N}$ und den trivialen Zustand $\mathbb{1} \in Q$, dass $[\mathbb{1}^i]_{\equiv_T} = [\varepsilon]_{\equiv_T} \in \mathcal{N}_1$ ist.

Um sinnvoll über die Anzahl der Buchstaben in einem $m \in Q^*$ reden zu können, die in C liegen, definieren wir zunächst noch die Länge eines Wortes in C.

Definition 3.5 (Länge in C)**.** Sei $q \in Q^*$. Dann ist

$$\|q\|_C = \min\{k \in \mathbb{N} \mid q \equiv_T b_0 a_1 b_1 \cdots b_{k-1} a_k b_k \text{ mit } a_i \in C,\, b_j \in (Q \setminus C)^*\} \tag{3.1}$$

die Länge eines Zustandswortes q in C.

Lemma 3.6. Es existiert ein $\ell \in \mathbb{N}$, sodass für alle $w \in \Sigma^{\ell n_1}$ und $q \in Q^*$ gilt, dass $[q]_{\equiv_T} \cdot w \in \mathcal{N}_1$ ist.

Beweis. Sei q aus Q^* beliebig. Zunächst lassen wir ein beliebiges Wort u der Länge n_1 auf q wirken. Da n_1 größer als $|P|$ ist, gilt für das resultierende Zustandswort $q' = q \cdot u$, dass es nur aus Buchstaben aus $C \cup F$ besteht, also $q' \in (C \cup F)^*$. Wir führen den Beweis für Lemma 3.6 im Folgenden als Induktion über die Länge von q' in C.

Induktionsanfang Im Fall, dass $\|q'\|_C = 0$ ist, ist q' ein Wort über F. Da die Zustände in F finitär sind, also jeder in ihnen gestartete Lauf nach höchstens einer konstanten Anzahl an Schritten in einem trivialen Zustand enden muss und n_1 größer als die Mächtigkeit von F ist, gilt für jedes Wort $v \in \Sigma^{n_1}$ der Länge n_1, dass $q' \cdot v = \mathbb{1}^{|q'|}$ ist. Also gilt

$$q' \cdot v = q \cdot uv = \mathbb{1}^{|q'|}$$

und es ist $[q]_{\equiv_T} \in \mathcal{N}_1$.

Induktionsschritt Für den Fall $\|q'\|_C > 0$ müssen wir zwei Fälle unterscheiden:

- Für alle Wörter $v \in \Sigma^{n_1}$ ist $\|q' \cdot v\|_C$ kleiner als $\|q'\|_C$, die Anzahl der Buchstaben aus C nimmt also ab.

Da die Länge von q' in C unter der Dualwirkung von v abgenommen hat, ist $[q' \cdot v]_{\equiv_T}$ nach Induktionsvoraussetzung in $\mathcal{N}_1$.

- Für mindestens ein Wort w der Länge n_1 ist $\|q' \cdot w\|_C$ gleich $\|q'\|_C$, die Anzahl der Buchstaben aus C bleibt also für mindestens eine Eingabe gleich.

Wir wissen, dass q' von der Form $b_0 a_1 b_1 \cdots b_{m-1} a_m b_m$ ist, wobei die a_i Buchstaben aus C und die b_i Wörter aus F^* sind, also insbesondere auch leer sein können. Zustände aus P können in q' nicht vorkommen, da wir diese durch die Dualwirkung des Wortes u zu Beginn des Beweises eliminiert haben. Da n_1 größer als die Anzahl aller finitären Zustände ist und finitäre Zustände nicht in einen Kreis übergehen können, werden die b_j alle in $\mathbb{1}^{|b_j|}$ überführt. Das Kreuzdiagramm in Abbildung 3.1 verdeutlicht dies. Weiterhin gilt, dass die Länge in C nicht kleiner geworden ist, also jeder Buchstabe $a_i \cdot v^{(2i-1)}$ weiterhin

aus C sein muss. Da die Wörter $v^{(2i-1)}$ Länge n_1, also ein Vielfaches aller Zyklenlängen in $\mathcal{T}$ lang sind und nach Lemma 2.14 auf jedem Pfad nur ein Kreis existieren kann, muss $a_i \cdot v^{(2i-1)} = a_i$ sein und $[q']_{\equiv_T}$ ist per Definition Element von $\mathcal{N}_1$.

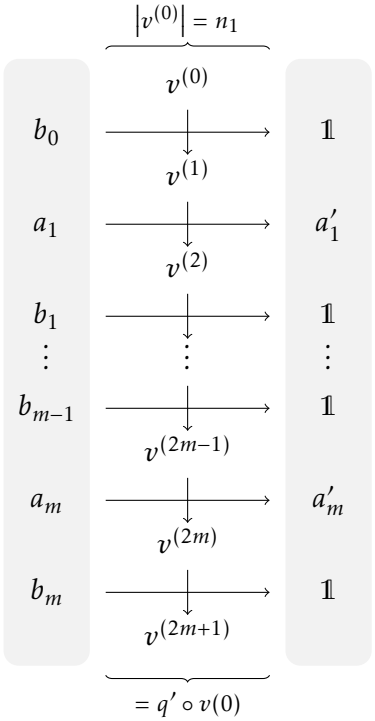


Abbildung 3.1: Dualwirkung eines Wortes v auf das Zustandswort q'

Damit ist die Behauptung aus Lemma 3.6 bewiesen. □

Dies bedeutet, dass für alle Wörter w der Länge mindestens $2n_1$ und für alle $q \in Q^*$ gilt, dass $[q \cdot w]_{\equiv_T} \in \mathcal{N}$ oder $[q \cdot w]_{\equiv_T} \in \langle F \rangle$ ist.

Proposition 3.7. Die Menge $\mathcal{N} = \mathcal{N}_1 \cup \langle F \rangle$ ist der Nucleus von $\mathscr{M}(\mathcal{T})$. Insbesondere ist $\mathcal{N}$ endlich.

Beweis. Da $\mathcal{T}$ von beschränkter Aktivität ist, operieren die finitären Zustände aus F nur auf einem endlichen Präfix von Wörtern $u \in \Sigma^*$ der Länge maximal k nichttrivial. Dies folgt direkt auch für Zustandswörter $f \in F^*$ und somit für die Funktionen $(f\circ) \in \langle F \rangle$. Damit also zwei Funktionen $(f\circ), (g\circ) \in \langle F \rangle$ nicht identisch sind, müssen sie bereits auf den ersten k Buchstaben eines Wortes u verschieden operieren und ein solches $(f\circ)$ kann folglich durch den endlichen Präfix auf dem es nichttrivial operiert eindeutig identifiziert werden. Damit ist auch $\langle F \rangle$ endlich.

Für die $[m]_{\equiv_T} \in \mathcal{N}_1$ gilt, dass genau ein Wort $u \in \Sigma^{n_1}$ existiert, sodass $m \in m\backslash v$ ist und für alle anderen Wörter $v \neq u, v \in \Sigma^{n_1}$ gilt, dass $m\backslash v \subseteq F^*$ ist. Die Elemente von $\mathcal{N}_1$ können also anhand ihrer Wirkung auf Σ^{n_1} und der Menge $m\backslash u$ charakterisiert werden. Das bedeutet, dass eine Abbildung

$$\iota : \mathcal{N}_1 \hookrightarrow ((\Sigma^{n_1})^{\Sigma^{n_1}}) \times ((F^* \cup C)^{\Sigma^{n_1}}), [m]_{\equiv_T} \mapsto (m|_{\Sigma^{n_1}}, u \mapsto m \cdot u)$$

existiert. Für beliebige $[s]_{\equiv_T}, [t]_{\equiv_T} \in \mathcal{N}_1$ mit $[s]_{\equiv_T} \neq [t]_{\equiv_T}$ gibt es zwei Fälle. Im ersten Fall gilt, dass

$$\exists u \in \Sigma^{n_1} : (s\circ)u \neq (t\circ)u.$$

Daraus folgt, dass $\iota(s) \neq \iota(t)$, da sie sich in der ersten Komponente unterscheiden. Der zweite Fall ist, dass

$$\exists v \in \Sigma^{n_1}, w \in \Sigma^+ : (s\circ)vw \neq (t\circ)vw \text{ mit } (s\circ)v = (t\circ)v.$$

Da $(s\circ)vw = (s \circ v)((s \cdot v)\circ)w$ und $(t\circ)vw = (t \circ v)((t \cdot v)\circ)w$ gelten, muss also $s \cdot v \neq t \cdot v$ sein, da sonst $(s\circ)vw = (t\circ)vw$ wäre. Daraus folgt, dass $\iota([s]_{\equiv_T}) \neq \iota([t]_{\equiv_T})$, da sie sich in der zweiten Komponente unterscheiden. Also ist die Abbildung injektiv. Da also $\mathcal{N}_1$ injektiv in ein Kreuzprodukt aus endlichen Mengen abgebildet werden kann, ist $\mathcal{N}_1$ ebenfalls endlich und die Behauptung aus Proposition 3.7 gilt. □

Also ist $\mathscr{M}(\mathcal{T})$ kontrahierend mit Nucleus $\mathcal{N}$ und die Behauptung aus Satz 3.2 ist bewiesen.

3.3 Wortproblem von Automatenmonoiden beschränkter Aktivität in deterministisch logarithmischem Platz

Im Folgenden werden wir zeigen, dass das Wortproblem von Automatenmonoiden mit beschränkter Aktivität in L liegt, also von einer deterministischen Turingmaschine mit einem logarithmisch beschränktem Arbeitsband entschieden werden kann.

Satz 3.8. *Sei $\mathcal{T} = (Q, \Sigma, \delta)$ von beschränkter Aktivität und $\mathscr{M}(\mathcal{T})$ ein Automatenmonoid von beschränkter Aktivität. Dann ist das Wortproblem von $\mathscr{M}(\mathcal{T})$ in L.*

Wortproblem von Automatenmonoiden

Konstant	*Automat $\mathcal{T} = (Q, \Sigma, \delta)$.*
Eingabe	*Zustandswörter $p, q \in Q^*$.*
Ausgabe	*Gilt $p \equiv_T q$?*

Wir werden das Wortproblem eines Automatenmonoids $\mathscr{M}(\mathcal{T})$ im weiteren Verlauf mit $WP(\mathscr{M}(\mathcal{T}))$ bezeichnen. Um das Wortproblem lösen zu können, werden wir zunächst das

Komplement des Wortproblems, also die Frage nach der Ungleichheit zweier Zustandswörter, in logarithmischem Platz entscheiden.

Beweis. Sei also $\mathcal{T} = (Q, \Sigma, \delta)$ ein Automat mit beschränkter Aktivität und $\mathscr{M}(\mathcal{T})$ das von ihm erzeugte Automatenmonoid beschränkter Aktivität. Nach Satz 3.2 ist $\mathscr{M}(\mathcal{T})$ kontrahierend mit Nucleus $\mathcal{N}$ aus Proposition 3.7. Nach [HRR17, Proposition 9.3.11] existieren also Konstanten $\lambda \geq 1, 0 < \rho < 1, \nu \geq 0$, wobei für alle $p \in Q^*$ gilt, dass $\left|[p]_{\equiv_\mathcal{T}}\right| \geq 1/\rho \left|[p]_{\equiv_\mathcal{T}} \cdot u\right| + \nu$ für alle $u \in \Sigma^\lambda$, wobei die Länge eines Monoidelements die kleinste Anzahl an Erzeugenden (also Zuständen) ist, die dieses Element bilden können. Daraus folgt, dass für ein $v \in \Sigma^{s\lambda}$ mit $s = -\log(n_0)/\log(\rho)$ die Äquivalenzklasse $[p \cdot v]_{\equiv_\mathcal{T}} \in \mathcal{N}$ sein muss, da dann $|p \cdot u| \leq 1$[2], also ist $p \cdot u = p'$ ein Zustand aus C und somit ist $[p']_{\equiv_\mathcal{T}} \in \mathcal{N}$ oder p war bereits im Nucleus und seine Länge reduziert sich nicht weiter. Da der Automat bereits vor Ausführung des Wortproblem-Algorithmus bekannt ist, können $\mathscr{M}(\mathcal{T})$ und $\mathcal{N}$ vorab berechnet werden. Da $\mathcal{N}$ endlich ist (siehe Proposition 3.7), kann für alle Elemente bestimmt und tabelliert werden, ob diese äquivalent sind oder nicht. Die Konstanten λ, ρ, ν und somit s mit $n_0 = \max\{|p|, |q|\}$ können ebenfalls vor der Ausführung bestimmt werden. Für alle $v \in \Sigma^{s\lambda}$ gilt außerdem $[q \cdot v]_{\equiv_\mathcal{T}} \in \mathcal{N}$ und $[p \cdot v]_{\equiv_\mathcal{T}} \in \mathcal{N}$. Der folgende Algorithmus löst $WP(\mathscr{M}(\mathcal{T}))$. Zähle alle Wörter $v \in \Sigma^{s\lambda}$ auf und überprüfe im tabellierten Nucleus, ob $[p \cdot v]_{\equiv_\mathcal{T}} \neq [q \cdot v]_{\equiv_\mathcal{T}}$ ist. Dies gilt genau dann, wenn $(p, q) \in WP(\mathscr{M}(\mathcal{T}))$. Da die Länge von v logarithmisch in n_0 und somit der Eingabe ist, ist $WP(\mathscr{M}(\mathcal{T}))$ in L. □

[2] Dies gilt, da $n_0 \cdot \rho^s \implies s = -\log(n_0)/\log(\rho)$ und da $\log(\rho) < 0$ ist s positiv.

4 Das uniforme Wortproblem finitärer Automatengruppen ist coNP-vollständig

In diesem Kapitel werden wir das uniforme Wortproblem finitärer Automatengruppen hinsichtlich seiner Komplexität untersuchen und zeigen, dass das Komplement des Problems in der Komplexitätsklasse NP liegt und so schwer wie alle Probleme in NP, also NP-vollständig ist.

4.1 Grundlagen und Begriffe

Zentrales Element der Fragestellung dieser Arbeit ist das Wortproblem von Automatengruppen, insbesondere dessen Komplexität. Das neutrale Element einer Gruppe wird in dieser Arbeit mit $\mathbb{1}$ notiert. Da wir Elemente von Automatengruppen zum Teil als Zustandswörter über der Zustandsmenge eines $\mathcal{G}$-Automaten darstellen, schreiben wir $p =_{\mathcal{G}(\mathcal{T})} q$, falls die Wörter p und q in der Gruppe $\mathcal{G}(\mathcal{T})$ das gleiche Element beschreiben. Das Wortproblem ist die Frage, ob für einen konstanten Automaten $\mathcal{T}$ ein gegebenes Zustandswort q über dessen Zustandsmenge trivial auf Σ^* operiert, die induzierte Funktion von q also die Identität der Gruppe $\mathcal{G}(\mathcal{T})$ ist. Für das Wortproblem existiert außerdem eine uniforme Variante, bei der der $\mathcal{G}$-Automat $\mathcal{T}$ nicht konstant, sondern Teil der Eingabe ist. Diese Variante ist ungleich schwerer, da Konstanten bezüglich des Automaten oder seiner erzeugten Gruppe nicht vorab berechnet werden können, wie es beispielsweise in Kapitel 3 für das Wortproblem der Automatenmonoide gemacht wurde. Für das nicht uniforme Wortproblem ist bekannt, dass es für beliebige Automatengruppen in PSPACE[1] liegt (siehe [Ste15]) und dass es Automatengruppen gibt, deren Wortproblem PSPACE-vollständig ist (siehe [WW20]). Wir betrachten zunächst das uniforme Wortproblem für finitäre Automatengruppen und wollen zeigen, dass dieses für die Zeitkomplexitätsklasse coNP vollständig ist, sich also jedes Problem innerhalb der Klasse coNP darauf reduzieren lässt und das uniforme Wortproblem selbst innerhalb der Klasse coNP liegt. Hierbei ist coNP die Klasse aller Probleme, deren Komplement von einer nichtdeterministischen Turingmaschine erkannt werden kann, deren Rechenzeit durch ein Polynom $p \in \mathcal{O}(n^c)$ mit beliebigem c und Eingabelänge n beschränkt ist.

Ein grundlegender Baustein des Beweises werden sogenannte balancierte Kommutatoren sein, mit deren Hilfe eine logische UND-Verknüpfung in eine Gruppe kodiert werden kann. Diese Konstruktion des balancierten Kommutators wurde erstmals von Mal'cev in [Mal62] und

[1] Die Klasse PSPACE wird in Kapitel 5 eingeführt.

später von Barrington in [Bar89] genutzt. Um diesen zu konstruieren, benötigen wir zunächst folgende Begriffe. Für g, h aus einer Gruppe G bezeichnen wir $g^h = h^{-1}gh$ als die *Konjugation* des Elements g mit h. Der *Kommutator* von g und h ist $[g,h] = g^{-1}h^{-1}gh$.

Definition 4.1 (Balancierter Kommutator)**.** Seien $g_1, \ldots, g_n, \alpha, \beta \in G$. Wir definieren den balancierten Kommutator induktiv als

$$B_{\alpha,\beta}[g_1] = g_1$$
$$B_{\alpha,\beta}[g_n, \ldots, g_1] = \left[B_{\alpha,\beta}[g_n, \ldots, g_{\lfloor \frac{n}{2} \rfloor + 1}]^\beta, B_{\alpha,\beta}[g_{\lfloor \frac{n}{2} \rfloor}, \ldots, g_1]^\alpha\right].$$

Der balancierte Kommutator $B_{\alpha,\beta}[g_n, \ldots, g_1]$ lässt sich als Binärbaum interpretieren, dessen Blätter die $g_n, \ldots, g_1$ sind, wobei der Wurzelknoten dem Funktionswert von $B_{\alpha,\beta}$ an der Stelle $(g_n, \ldots, g_1)$ entspricht. Jede Ebene des Baums entspricht hierbei einer Iteration des balancierten Kommutators.

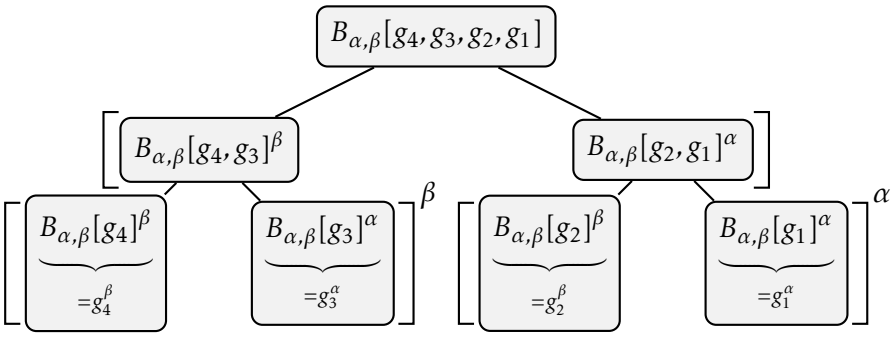


Abbildung 4.1: Interpretation von $B_{\alpha,\beta}[g_4, g_3, g_2, g_1]$

Mit Hilfe eines Kommutators ist es uns möglich, die logische Konjunktion in die alternierende Gruppe A_5, die Gruppe aller geraden Permutationen von fünf Elementen, zu kodieren. Hierzu identifizieren wir das neutrale Element $\mathbb{1} \in A_5$ mit dem logischen Wahrheitswert *falsch* und ein Element weiteres Element $\sigma \in A_5$ mit dem logischen Wahrheitswert *wahr*. Damit der Kommutator dieser zweier Element $g, h \in \{\sigma, \mathbb{1}\}$ sich wie die logische Konjunktion verhält, muss also gelten, dass

$$[g,h] = \begin{cases} \sigma, & \text{falls } g = h = \sigma \\ \mathbb{1} & \text{sonst.} \end{cases} \tag{4.1}$$

Lemma 4.2. Es existieren $\sigma, \alpha, \beta \in A_5$, sodass $[\sigma^\beta, \sigma^\alpha] = \sigma$ gilt.

Beweis. Siehe [WW20, Lemma 2]. □

Im weiteren Verlauf der Arbeit ist σ aus Lemma 4.2 eben jenes Element $\sigma \in A_5$, das wir mit *wahr* identifizieren. Falls nicht explizit genannt sind außerdem mit α und β die aus Lemma 4.2 gemeint.

Lemma 4.3. Seien $\mathbb{1}, \sigma, \alpha, \beta \in A_5$ und $g_1, \ldots, g_n \in \{\sigma, \mathbb{1}\}$. Dann gilt $B_{\alpha,\beta}[g_n, \ldots, g_1] = \sigma$ genau dann, wenn für alle $i \in \{1, \ldots, n\}$ gilt, dass $g_i = \sigma$ ist und ansonsten $B_{\alpha,\beta}[g_n, \ldots, g_1] = \mathbb{1}$.

Beweis. Wir werden dies per Induktion über die Anzahl der g_i beweisen.

Induktionsanfang Sei $n = 1$. Dann gilt offensichtlich

$$B_{\alpha,\beta}[g_1] = g_1 = \begin{cases} \sigma, & \text{falls } g_1 = \sigma \\ \mathbb{1} & \text{sonst.} \end{cases}$$

Induktionsschritt Sei nun $n > 1$.

Hier unterscheiden wir zwei Fälle. Im ersten Fall gilt $g_1 = \ldots = g_n = \sigma$ und folglich

$$\begin{aligned} B_{\alpha,\beta}[g_n, \ldots, g_1] &= \left[B_{\alpha,\beta}[g_n, \ldots, g_{\lfloor \frac{n}{2} \rfloor + 1}]^\beta, B_{\alpha,\beta}[g_{\lfloor \frac{n}{2} \rfloor}, \ldots, g_1]^\alpha\right] \\ &\overset{\text{I.V.}}{=} \left[\sigma^\beta, \sigma^\alpha\right] \\ &= \sigma \end{aligned}$$

nach Lemma 4.2. Im zweiten Fall ist mindestens eines der $g_i = \mathbb{1}$. Dies liefert uns weitere drei Fälle:

- Für alle $j \in \left\{1, \ldots, \lfloor \frac{n}{2} \rfloor\right\}$ gilt $g_j = \sigma$.
- Für alle $j \in \left\{\lfloor \frac{n}{2} \rfloor + 1, \ldots, n\right\}$ gilt $g_j = \sigma$.
- Keiner der beiden vorherigen Fälle gilt.

Für den ersten Fall gilt

$$\begin{aligned} B_{\alpha,\beta}[g_n, \ldots, g_1] &= \left[B_{\alpha,\beta}[g_n, \ldots, g_{\lfloor \frac{n}{2} \rfloor + 1}]^\beta, B_{\alpha,\beta}[g_{\lfloor \frac{n}{2} \rfloor}, \ldots, g_1]^\alpha\right] \\ &\overset{\text{I.V.}}{=} [\mathbb{1}^\beta, \sigma^\alpha] \\ &= (\mathbb{1}^\beta)^{-1} (\sigma^\alpha)^{-1} \mathbb{1}^\beta \sigma^\alpha \\ &= \beta^{-1} \mathbb{1}^{-1} \beta \alpha^{-1} \sigma^{-1} \alpha \beta^{-1} \mathbb{1} \beta \alpha^{-1} \sigma \alpha \\ &= \alpha^{-1} \sigma^{-1} \alpha \alpha^{-1} \sigma \alpha \\ &= \mathbb{1}. \end{aligned}$$

Der zweite und dritte Fall funktionieren analog.

Damit ist Lemma 4.3 bewiesen. □

Zusammengefasst gilt also

$$B_{\alpha,\beta}[g_n, \ldots, g_1] = \begin{cases} \sigma, & \text{falls } g_1 = \ldots = g_n = \sigma \\ \mathbb{1}, & \text{falls } g_1 = \mathbb{1} \vee \ldots \vee g_n = \mathbb{1} \end{cases} \tag{4.2}$$

und der balancierte Kommutator verhält sich äquivalent zu einer mehrwertigen aussagenlogischen Konjunktion. Da wir im Folgenden Zustandswörter über der Zustandsmenge eines Automaten $\mathcal{T} = (Q, \Sigma, \delta)$ betrachten, übertragen wir die Definition des balancierten Kommutators intuitiv auf diese Zustandswörter, beziehungsweise auf die Zustände des Automaten und notieren den balancierten Kommutator eines Zustandswortes $q \in Q^*$ mit $B_{\mathcal{T};\alpha,\beta}[q]$.

Mit Hilfe dieser Grundlagen ist es uns nun möglich, die Kernaussage dieses Kapitels aufzustellen und zu beweisen.

Satz 4.4. *Das uniforme Wortproblem finitärer Automatengruppen ist für die Klasse* coNP *vollständig.*

Wortproblem für Automatengruppen

Konstant	*Ein $\mathcal{G}$-Automat $\mathcal{T}$.*
Eingabe	*Ein Zustandswort q aus $(Q^{\pm})^*$.*
Ausgabe	*Gilt $(q\circ)u = u$ für alle $u \in \Sigma^*$?*

Wir werden das uniforme Wortproblem einer Gruppe G im weiteren Verlauf der Arbeit mit $\mathrm{UWP}(G)$ und die spezielle Variante für finitäre Automatengruppen mit UWPfA bezeichnen.

4.2 Beweis der Aussage

Um dieses Theorem zu beweisen werden wir einerseits zeigen, dass das Komplement des UWPfA innerhalb der Klasse coNP liegt und andererseits, dass es mindestens so schwer wie alle Probleme in coNP ist. Hierzu werden wir das Problem Exakt-3-KNF-SAT reduzieren.

4.2.1 Zugehörigkeit zu NP

Lemma 4.5. Das Komplement des uniformen Wortproblems für Automatengruppen beschränkter Aktivität liegt in NP.

Beweis. Sei $\mathcal{I} = (\mathcal{T} = (Q, \Sigma, \delta), q \in Q^*)$ eine Instanz des Komplements des UWPfA und sei M eine nichtdeterministische Turingmaschine. Wir können $\mathcal{I}$ mit Hilfe von M via *guess and check* lösen. Hierzu rät M ein Wort w, sodass $q\circ w \neq w$, falls $q \neq_{\mathcal{G}(\mathcal{T})} \mathbb{1}$ ist, und überprüft diese Lösung anschließend. Da $\mathcal{T}$ von finitärer Aktivität ist (andernfalls wäre $\mathcal{G}(\mathcal{T})$ ebenfalls nicht von finitärer Aktivität), endet jeder Lauf nach maximal $|Q|$ Schritten im trivialen Zustand, also dem Zustand, der wie das Gruppenelement $\mathbb{1}$ auf Σ^* wirkt. Das Überprüfen der Lösung w erfordert die Berechnung $q\circ w$. Da q als Wort über Q vorliegt, benötigt die Berechnung $|q| \cdot |w|$ viele Schritte.

Sowohl das Raten des Lösungswortes w, als auch die Überprüfung sind also in polynomieller Zeit bezüglich der Eingabe möglich und das Komplement des UWPfA liegt in NP. □

4.2.2 NP-Vollständigkeit

Lemma 4.6. Exakt-3-KNF-SAT lässt sich in logarithmischem Platz auf das Komplement des UWPfA reduzieren.

Exakt-3-KNF-SAT

Eingabe Eine aussagenlogische Formel $\varphi = \bigwedge_{i=1}^{k} \bigvee_{j=1}^{3} L_{i,j}$.
Ausgabe Ist φ erfüllbar?

Konstruktion der Reduktionsfunktion

Als Eingabe erhalten wir also eine aussagenlogische Formel φ, die in konjunktiver Normalform ist und genau drei Literale pro Klausel enthält. Wir werden aus dieser Formel nun in logarithmischem Platz einen Automaten $\mathcal{T} = (Q, \Sigma, \delta)$ sowie ein Zustandswort q generieren, das genau dann nicht trivial auf Σ^* operiert, wenn eine erfüllende Belegung für φ existiert.

Sei also $\varphi = \bigwedge_{i=1}^{k} \bigvee_{j=1}^{3} L_{i,j}$ gegeben, wobei k die Anzahl der Klauseln in φ ist und n die Anzahl der atomaren Formeln, die in φ vorkommen. Ohne Einschränkung können wir die atomaren Formeln A_l aufsteigend sortieren. Unser Automat $\mathcal{T} = (Q, \Sigma, \delta)$ wird nun aus verschiedenen, disjunkten Teilautomaten konstruiert. Sei $\Sigma = \Sigma_\top \uplus \Sigma_\bot$ mit $\Sigma_\top = \{a_1, a_2\}$, $\Sigma_\bot = \{a_3, a_4, a_5\}$ das Alphabet unseres Automaten. Wir werden Buchstaben aus $\Sigma_\top$ mit dem Symbol $\top$ und Buchstaben aus $\Sigma_\bot$ mit dem Symbol $\bot$ benennen.

Für jede Klausel $1 \leq i \leq k$ konstruieren wir im Folgenden einen Teilautomaten $\mathcal{T}_i = (Q_i, \Sigma, \delta_i)$. Seien $L_{i,1} = A_r^{\epsilon_r}$, $L_{i,2} = A_s^{\epsilon_s}$ und $L_{i,3} = A_t^{\epsilon_t}$ mit $1 \leq r < s < t < n$ und $\epsilon_i \in \{+1, -1\}$, wobei $A^{+1} = A$ ein positives Literal und $A^{-1} = \neg A$ ein negatives Literal im Sinne der Aussagenlogik ist. Sei $Q_i = \{q_{i,j} \mid j \in \{0, \ldots, n\}\} \cup \{\hat{q}_{i,j} \mid j \in \{r+1, \ldots, n\}\}$ und

$$\begin{aligned}
\delta_i = &\{q_{i,j} \xrightarrow{a/a} q_{i,j+1} \mid a \in \Sigma, j \in \{0, \ldots, n-1\} \setminus \{r, s, t\}\} \\
&\cup \{\hat{q}_{i,j} \xrightarrow{a/a} \hat{q}_{i,j+1} \mid a \in \Sigma, j \in \{r+1, \ldots, n-1\}\} \\
&\cup \{q_{i,j} \xrightarrow{\top/\top} q_{i,j+1}, q_{i,j} \xrightarrow{\bot/\bot} \hat{q}_{i,j+1} \mid j \in N\} \\
&\cup \{q_{i,j} \xrightarrow{\bot/\bot} q_{i,j+1}, q_{i,j} \xrightarrow{\top/\top} \hat{q}_{i,j+1} \mid j \in P\}
\end{aligned}$$

mit $P = \{j \mid j \in \{r, s, t\}, \epsilon_j > 0\}$ und $N = \{j \mid j \in \{r, s, t\}, \epsilon_j < 0\}$.

Die Idee der Klauselautomaten, wie beispielhaft in Abschnitt 4.2.2 dargestellt, ist zu testen, ob eine Belegung der drei Literale der Klausel, repräsentiert durch die Zustände $q_{i,r}, q_{i,s}, q_{i,t}$, existiert, sodass die Klausel wahr wird. Ist dies der Fall, so führt ein Übergang in die „untere Ebene“

des Automaten und der Lauf endet im Zustand σ des Automaten. Durch das Zusammenführen all dieser Klauselteilautomaten als Zustandswort kann also überprüft werden, ob ein Wort (= Belegung) existiert, sodass jede Klausel wahr wird. Genau dann gehen alle Teilautomaten in den Zustand σ über. Sonst endet mindestens einer der Klauselautomaten im trivialen Zustand $\mathbb{1}$ und nach Lemma 4.3 und Gleichung (4.2) kollabiert der balancierte Kommutator des Zustandswort zu $\mathbb{1}$ und das Zustandswort operiert trivial auf allen Eingaben.

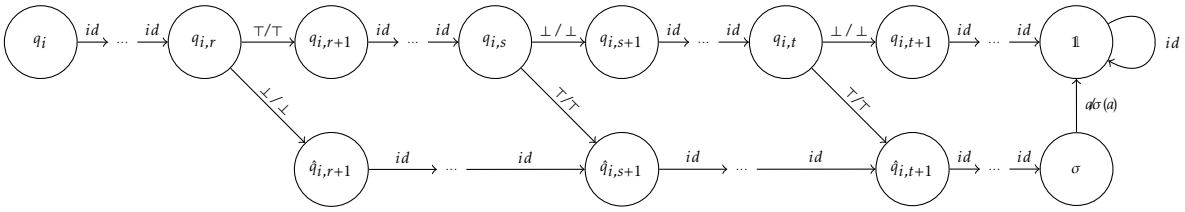


Abbildung 4.2: Beispiel eines Klauselautomaten für die Klausel $K_i = (A_r \vee \neg A_s \vee \neg A_t)$

Des Weiteren konstruieren wir Automaten $\mathcal{T}_\alpha$ und $\mathcal{T}_\beta$. Sei $\mathcal{T}_\alpha = (Q_\alpha, \Sigma, \delta_\alpha)$ mit

$$Q_\alpha = \{\alpha_i \mid i \in \{0, n-1\}\} \cup \{\alpha\}$$
$$\delta_\alpha = \{\alpha_i \xrightarrow{a/a} \alpha_{i+1} \mid a \in \Sigma, i \in \{0, \dots, n-2\}\} \cup \{\alpha_{n-1} \xrightarrow{a/a} \alpha \mid a \in \Sigma\}.$$

Der Automat $\mathcal{T}_\beta$ wird analog konstruiert. Hier gilt, dass die Zustände α und β so auf Σ^* operieren, wie die Elemente der A_5 aus Lemma 4.2, die Buchstaben in Σ also permutieren.

Der Gesamtautomat $\mathcal{T} = (Q, \Sigma, \delta)$ wird nun konstruiert als die Vereinigung aller zuvor konstruierten Teilautomaten. Außerdem fügen wir der Zustandsmenge von Q noch einen weiteren Zustand $\mathbb{1}$ sowie der Übergangsrelation δ weitere Kanten hinzu.

$$Q = \left(\biguplus_{i=1}^{k} Q_i\right) \cup Q_\alpha \cup Q_\beta \cup \{\mathbb{1}\}$$
$$\delta = \left(\biguplus_{i=1}^{k} \delta_i\right) \cup \delta_\alpha \cup \delta_\beta \cup \{q_{i,n} \xrightarrow{a/a} \mathbb{1}, \hat{q}_{i,n} \xrightarrow{a/\sigma(a)} \mathbb{1} \mid i \in \{1, \dots, k\}, a \in \Sigma\}$$
$$\cup \{\mathbb{1} \xrightarrow{a/a} \mathbb{1}, \alpha \xrightarrow{a/\alpha(a)} \mathbb{1}, \beta \xrightarrow{a/\beta(a)} \mathbb{1} \mid a \in \Sigma\}$$

Man bemerke, dass die Zustände $\hat{q}_{i,n}$ wie ein Zustand σ operieren, die Buchstaben aus Σ also wie das Element $\sigma \in A_5$ aus Lemma 4.2 permutieren. Der Automat $\mathcal{T}$ ist ein $\mathcal{G}$-Automat, da lediglich die Zustände $\hat{q}_{i,n}, \beta, \alpha$ nichttrivial Zuständsübergänge haben. Da diese die Buchstaben aus Σ lediglich permutieren, also Bijektionen induzieren, sind sie invertierbar. Außerdem ist $\mathcal{T}$ finitär, da keine Zyklen im Automaten existieren (siehe Fakt 2.16) und somit ist auch die Automatengruppe $\mathcal{G}(\mathcal{T})$ nach Gleichung (2.2) finitär.

Sei nun $q = B_{\mathcal{T};\alpha_0,\beta_0}[q_{k,0}, \dots, q_{1,0}]$. Damit ist unsere Konstruktion komplett und es gilt zu zeigen, dass diese in logarithmischem Platz möglich und korrekt ist.

Beweis der Platzschranke

Die Konstruktion des Automaten erfordert im wesentlichen zwei Arbeitsschritte. Zunächst die Konstruktion des Automaten $\mathcal{T}$ und anschließend die Konstruktion des balancierten Kommutators $B_{\mathcal{T};\alpha_0,\beta_0}[q]$.

$$\varphi \xrightarrow{(1)} \mathcal{T} \xrightarrow{(2)} B_{\mathcal{T};\alpha_0,\beta_0}[q]$$

(1) Insgesamt werden $k+2$ Teilautomaten konstruiert. Die $q_k, \ldots, q_1$ haben jeweils $n+1$ Zustände, α und β ebenfalls. Anschließend mündet jeder der Automaten in den Zustand σ beziehungsweise den Zustand $\mathbb{1}$. Die Konstruktion des Automaten ist also in $\mathcal{O}(k \cdot n)$ Schritten möglich.

(2) siehe [WW20, Lemma 6].

Beweis der Korrektheit der Konstruktion

Im Folgenden beweisen wir nun die Korrektheit der Reduktion.

Lemma 4.7. $\exists w \in \Sigma^* : q \circ w \neq w \iff \exists \mathcal{A} : \mathcal{A} \models \varphi$.

Zunächst werden wir beweisen, dass der balancierte Kommutator für die soeben konstruierten Teilautomaten kompatibel mit der Dualwirkung eines Wortes $w \in \Sigma^n$ ist.

Proposition 4.8. Für $w \in \Sigma^n$ und $q \in \{q_{k,0}, \ldots, q_{1,0}\}^+$ gilt, dass $B_{\mathcal{T};\alpha_0,\beta_0}[q] \cdot w = B_{\mathcal{T};\alpha,\beta}[q \cdot w]$.

Beweis. Wir beweisen dies per Induktion über die Länge des Zustandswortes q.

Induktionsanfang Sei zunächst $|q| = 1$. Dann gilt $B_{\mathcal{T};\alpha_0,\beta_0}[q] \cdot w = q \cdot w = B_{\mathcal{T};\alpha,\beta}[q \cdot w]$ nach Definition 4.1.

Induktionsschritt Sei $|q| > 1$ mit $q = q_n \cdot q_1$. Sei außerdem $\ell = \lfloor \frac{n}{2} \rfloor$. Dann gilt

$$\begin{aligned}
B_{\mathcal{T};\alpha_0,\beta_0}[q] \cdot w =& \left[B_{\mathcal{T};\alpha_0,\beta_0}[q_n,\dots,q_{\ell+1}]^{\beta_0}, B_{\mathcal{T};\alpha_0,\beta_0}[q_n,\dots,q_{\ell+1}]^{\alpha_0}\right] \cdot w \\
&\overset{(1)}{=} (b_1^{\beta_0})^{-1}(b_2^{\alpha_0})^{-1} b_1^{\beta_0} b_2^{\alpha_0} \cdot w \\
&= \beta_0^{-1} b_1^{-1} \beta_0 \alpha_0^{-1} b_2^{-1} \alpha_0 \beta_0^{-1} b_1 \beta_0 \alpha_0^{-1} b_2 \alpha_0 \cdot w \\
&\overset{(2)}{=} \beta^{-1}(b_1^{-1} \cdot w)\beta\alpha^{-1}(b_2^{-1} \cdot w)\alpha\beta^{-1}(b_1 \cdot w)\beta\alpha^{-1}(b_2 \cdot w)\alpha \\
&\overset{(3)}{=} \beta^{-1}(b_1 \cdot w)^{-1}\beta\alpha^{-1}(b_2 \cdot w)^{-1}\alpha\beta^{-1}(b_1 \cdot w)\beta\alpha^{-1}(b_2 \cdot w)\alpha \\
&= ((b_1 \cdot w)^{\beta})^{-1}((b_2 \cdot w)^{\alpha})^{-1}(b_1 \cdot w)^{\beta}(b_2 \cdot w)^{\alpha} \\
&= \left[(b_1 \cdot w)^{\beta}, (b_2 \cdot w)^{\alpha}\right] \\
&\overset{\text{I.V.}}{=} \left[B_{\mathcal{T};\alpha,\beta}[q_n,\dots,q_{\ell+1} \cdot w]^{\beta}, B_{\mathcal{T};\alpha,\beta}[q_\ell,\dots,q_1 \cdot w]^{\alpha}\right] \\
&= B_{\mathcal{T};\alpha,\beta}[q \cdot w]
\end{aligned}$$

(1) Wir führen aus Gründen der Lesbarkeit die folgenden Notationen ein:

- $B_{\mathcal{T};\alpha_0,\beta_0}[q_n,\dots,q_{\ell+1}] = b_1$,
- $B_{\mathcal{T};\alpha_0,\beta_0}[q_\ell,\dots,q_1] = b_2$.

(2) Dies gilt, da die Zustände α_0, β_0 und alle Zustände $q_{i,0}$ des Automaten $\mathcal{T}$ auf den ersten n Buchstaben trivial operieren, das Wort w also nicht verändern.

(3) Es ist leicht zu sehen, dass in einem $\mathcal{G}$-Automaten gilt, dass $q^{-1} \cdot w = (q \cdot w)^{-1}$. Wäre $q^{-1} \cdot w \neq (q \cdot w)^{-1}$, dann wäre $((q^{-1} \cdot w)(q \cdot w)) \circ \neq_{\mathcal{G}(\mathcal{T})} \mathbb{1}$ und somit auch $(q^{-1}q) \circ \neq_{\mathcal{G}(\mathcal{T})} \mathbb{1}$.

Somit ist Proposition 4.8 bewiesen. □

Sei außerdem $\mathcal{AV}_\varphi$ die Menge der aussagenlogischen Variablen in φ und $\nu : \Sigma^n \to \{0,1\}^{\mathcal{AV}_\varphi}$ eine Hilfsfunktion mit

$$\nu : a_1 \cdots a_n \mapsto \begin{cases} V_i \mapsto 1, & \text{falls } a_i \in \Sigma_\top \\ V_i \mapsto 0, & \text{falls } a_i \in \Sigma_\bot. \end{cases}$$

Mit Hilfe dieser Hilfsfunktion können wir nun Lemma 4.7 beweisen.

Beweis. Wir werden hierzu die beiden Implikationen der Äquivalenz gesondert beweisen.

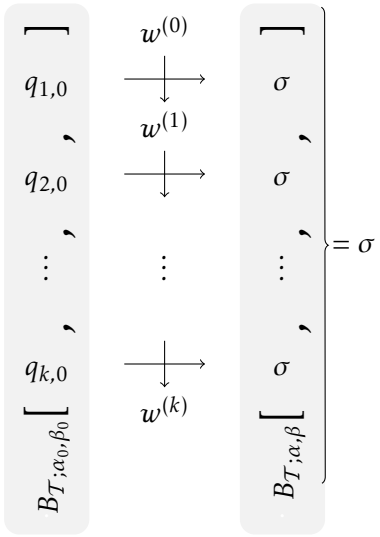
„$\Longrightarrow$“ Zunächst beweisen wir: $\exists w \in \Sigma^* : q \circ w \neq w \implies \exists \mathcal{A} : \mathcal{A} \models \varphi$.

$\exists w \in \Sigma^* : q \circ w \neq w$

$\implies \exists w \in \Sigma^* : B_{\mathcal{T};\alpha_0,\beta_0}[q_{k,0},\dots,q_{1,0}] \circ w \neq w$

$\implies \exists w \in \Sigma^* : B_{\mathcal{T};\alpha_0,\beta_0}[q_{k,0},\dots,q_{1,0}] \cdot w = B_{\mathcal{T};\alpha,\beta}[\underbrace{\sigma,\dots,\sigma}_{\text{k mal}}]$ nach Proposition 4.8

$\implies \forall i \in \{1,\dots,k\} : q_{i,0} \cdot w^{(i-1)} = \sigma$ nach Lemma 4.3



$\implies \forall i \in \{1,\dots,k\}$: Klausel i aus φ enthält mindestens ein Literal $L_{i,j}, j \in \{1,2,3\}$, sodass $\mathcal{A}(L_{i,j})$ die Klausel erfüllt.

$\implies \nu(w^{(0)}) \models \varphi$

„$\Longleftarrow$“ Nun beweisen wir: $\exists \mathcal{A} : \mathcal{A} \models \varphi \implies \exists w \in \Sigma^* : q \circ w \neq w$.

$\exists \mathcal{A} : \mathcal{A} \models \varphi$

$\implies \forall i \in \{1,\dots,k\} : \exists$ Klausel K_i in $\varphi : \exists L_{i,j} \in K_i : \mathcal{A}(L_{i,j}) = 1$

$\implies \forall i \in \{1,\dots,k\} : q_{i,0} \circ \nu^{-1}(\mathcal{A}) = \sigma$

$\implies B_{\mathcal{T};\alpha_0,\beta_0}[q_{k,0},\dots,q_{1,0}] \cdot \nu^{-1}(\mathcal{A}) = B_{\mathcal{T};\alpha,\beta}[\underbrace{\sigma,\dots,\sigma}_{\text{k mal}}]$ nach Proposition 4.8

$\implies \exists a \in \Sigma : B_{\mathcal{T};\alpha_0,\beta_0}[q_{k,0},\dots,q_{1,0}] \circ \nu^{-1}(\mathcal{A})a = \nu^{-1}(\mathcal{A})\sigma(a) \neq \nu^{-1}(\mathcal{A})a$

$\implies B_{\mathcal{T};\alpha_0,\beta_0}[q_{k,0},\dots,q_{1,0}] \neq_{\mathcal{G}(\mathcal{T})} \mathbb{1}$

□

Es gilt also $\overline{\texttt{UWPfA}} \in \text{NP}$ und $\texttt{Exakt-3-KNF-SAT} \leq_{log} \overline{\texttt{UWPfA}}$. Stephen A. Cook hat in [Coo71] gezeigt, dass die Menge der Tautologien in disjunktiver Normalform mit maximal drei Variablen pro Monom und somit auch das Problem 3-KNF-SAT für NP vollständig ist. Daraus lässt sich auf Grund folgenden Tricks leicht ableiten, dass auch Exakt-3-KNF-SAT NP-vollständig ist. Seien x, y, z aussagenlogische Variablen. Dann gilt

$$(x) = (x \vee y) \wedge (x \vee \neg y), \tag{4.3}$$

$$(x \vee y) = (x \vee y \vee z) \wedge (x \vee y \vee \neg z). \tag{4.4}$$

Das heißt wir können Klauseln mit weniger als drei Variablen durch das Hinzufügen neuer, nicht in der ursprünglichen Formel vorkommender Variablen in die richtige Form bringen und somit 3-KNF-SAT leicht in logarithmischem Platz auf Exakt-3-KNF-SAT reduzieren. Somit ist Satz 4.4 bewiesen ist. □

Außerdem lässt sich folgender Sachverhalt leicht aus Satz 4.4 folgern.

Korollar 4.9. Das uniformen Wortproblems für Automatengruppen beschränkter Aktivität ist coNP-Schwer.

5 Das uniforme komprimierte Wortproblem finitärer Automatengruppen ist PSPACE-vollständig

In diesem Kapitel werden wir das uniforme komprimierte Wortproblem finitärer Automatengruppen untersuchen und zeigen, dass es PSPACE-vollständig ist. Zunächst werden grundlegende Begriffe eingeführt, wobei Definitionen und Notationen aus den vorhergehenden Kapiteln übernommen und teilweise erweitert werden.

5.1 Grundlagen und Begriffe

Mit PSPACE bezeichnen wir die Klasse aller Probleme, die von einer deterministischen Turingmaschine entschieden werden können, wobei die Bänder der Turingmaschine bei Eingabelänge n durch ein Polynom $p \in \mathcal{O}(n^c)$ mit $c \in \mathbb{N}$ platzbeschränkt sind. Mit NPSPACE bezeichnen wir die Klasse der Probleme, die von einer nichtdeterministischen Turingmaschine mit gleicher Platzbeschränkung erkannt werden können. Man sieht leicht, dass PSPACE $\subseteq$ NPSPACE. Da Walter Savitch in [Sav70] gezeigt hat, dass eine nichtdeterministische Turingmaschine mit Platzbeschränkung $s(n)$ mit Hilfe einer deterministischen Turingmaschine mit Platzbeschränkung $s(n)^2$ simuliert werden kann, gilt also auch NPSPACE $\subseteq$ PSPACE. Soll also gezeigt werden, dass ein Problem $P \in$ PSPACE ist, so genügt es eine nichtdeterministische Turingmaschine mit polynomiell beschränktem Band anzugeben, die P erkennt.

Für das Verständnis des komprimierten Wortproblems ist der Begriff des Straight-Line-Programms oder auch SLP notwendig. Straight-Line-Programme sind kontextfreie Grammatiken, die nur ein Wort erzeugen. Wir bezeichnen die Auswertung eines solchen SLP S zum erzeugten Wort w im Folgenden mit $\mathrm{val}(S) = w$. Aufgrund der Strukturen solcher kontextfreien Grammatiken ist es möglich, dass die Länge von $\mathrm{val}(S)$ exponentiell groß in der Anzahl der Variablen von S ist. Dies wird für den späteren Beweis von Bedeutung sein. Da ein solches SLP hier zur Erzeugung eines Zustandswortes benötigt wird, sind seine Terminalsymbole Zustände aus Q.

Da wir in diesem Kapitel außerdem weitere aussagenlogische Begriffe benötigen, werden diese hier definiert und erklärt. Mit $\mathrm{dom}(\mathcal{A})$ bezeichnen wir die Domäne, also den Definitionsbereich einer Belegung $\mathcal{A} : V_\varphi \to \{0,1\}$. Dies sind im Grunde alle freien Variablen, die in φ vorkommen. Wenn $\mathcal{A}$ eine Belegung mit $\mathrm{dom}(\mathcal{A}) = \{x_1, \ldots, x_{d-1}\}$ ist, dann ist $\mathcal{A}[x_d \mid X_d]$ mit $X_d \in \{0,1\}$ die Belegung $\mathcal{A}'$ mit $\mathrm{dom}(\mathcal{A}') = \{x_1, \ldots, x_d\}$, wobei die Variable x_d mit dem Wert X_d belegt wird.

Satz 5.1. *Das komprimierte uniforme Wortproblem finitärer Automatengruppen ist PSPACE-Vollständig.*

Komprimiertes uniformes Wortproblem finitärer Automatengruppen

Eingabe *Finitärer $\mathcal{G}$-Automat $\mathcal{T} = (Q, \Sigma, \delta)$, Straight-Line-Programm S.*
Ausgabe *Gilt $(\mathrm{val}(S)\circ)u = u$ für alle $u \in \Sigma^*$?*

Wir werden das uniforme komprimierte Wortproblem einer Gruppe G im weiteren Verlauf der Arbeit mit UCWP(G) und die spezielle Variante für finitäre Automatengruppen mit UCWPfA bezeichnen.

5.2 Beweis der Aussage

Um Satz 5.1 zu beweisen, werden wir zunächst zeigen, dass das Komplement des uniformen komprimierten Wortproblems finitärer Automatengruppen $\overline{\text{UCWPfA}}$ von einer nichtdeterministischen Turingmaschine mit polynomiell beschränktem Band erkannt werden kann. Anschließend werden wir durch eine Reduktion die Vollständigkeit für PSPACE beweisen.

5.2.1 Zugehörigkeit zu PSPACE

Lemma 5.2. Das Komplement des uniformen komprimierten Wortproblems finitärer Automatengruppen liegt in PSPACE.

Beweis. Sei $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{G}$-Automat mit finitärer Aktivität. Dann ist $\mathcal{G}(\mathcal{T})$ ebenfalls finitär. Bei finitären Automaten wird jeder Zustand unter der Dualwirkung nach endlich vielen Schritten in einen trivialen Zustand überführt. Die maximale Anzahl der Schritte hängt direkt von der Anzahl an Zuständen des Automats ab, da keine Zyklen auf den Pfaden in $\mathcal{T}$ vorkommen dürfen weil $\mathcal{T}$ sonst mindestens beschränkte Aktivität hätte. Somit gilt insbesondere für jedes Zustandswort q aus $(Q^{\pm})^*$, dass q nach linear vielen Schritten unter der Dualwirkung zum Einselement $\mathbb{1}$ in $\mathcal{G}(\mathcal{T})$ transformiert wird. Um zu überprüfen, ob für Zustandswort q ungleich der $\mathbb{1}$ in $\mathcal{G}(\mathcal{T})$ ist, kann mit Hilfe von *guess and check* ein Wort $w \in \Sigma^{\leq |Q|}$ geraten werden, sodass $q \circ w \neq w$. Da w nur maximal linear lang in Q sein kann, kann die Korrektheit der Lösung in linearem Platz überprüft werden. Somit liegt $\overline{\text{UCWPfA}}$ in PSPACE. □

5.2.2 PSPACE-Vollständigkeit

Um Vollständigkeit für die Komplexitätsklasse PSPACE zu beweisen, werden wir nun zeigen, dass UCWPfA mindestens so schwer wie alle Probleme in PSPACE ist.

Lemma 5.3. QSAT[1] lässt sich in logarithmischem Platz auf das Komplement des uniformen komprimierten Wortproblems finitärer Automatengruppen reduzieren.

Quantifizierte Erfüllbarkeit (QSAT)

Eingabe	Eine quantifizierte boolsche Formel φ ohne freie Variablen.
Ausgabe	Ist φ wahr?

Dem Beweis der PSPACE-Vollständigkeit von QSAT in [Pap93] kann entnommen werden, dass φ in konjunktiver Normalform und Pränex-Normalform vorliegt. Um den Beweis von Lemma 5.3 einfacher führen zu können, werden wir zunächst einige Vorarbeit leisten müssen. Sei also φ eine Formel mit insgesamt D Variablen.

Umformen der Formel φ

Zunächst fordern wir, dass φ in konjunktiver Normalform mit exakt drei Literalen pro Klauseln vorliegt (vgl. `Exakt-3-KNF-SAT`). Dies funktioniert analog zum Trick aus Abschnitt 4.2 unter Verwendung der Gleichungen (4.3) und (4.4). Damit weiterhin nur gebundene Variablen in φ existieren, werden diese neuen Variablen allquantifiziert. Insgesamt ist dies in logarithmischem Platz möglich.

Die Formel liegt nun also in folgender Form vor, wobei die quantifizierten Variablen ohne Einschränkung von D bis 1 absteigend sortiert sind:

$$\varphi = Q_D x_D \cdots Q_1 x_1 : \bigwedge_{i=1}^{k} \bigvee_{j=1}^{3} L_{i,j}$$

mit $Q_i \in \{\exists, \forall\}$. Im Weiteren werden wir die Quantoren in eine, für uns geeignete Form bringen. Dazu werden wir mit Hilfe der semantischen Äquivalenz

$$\exists x : \phi(x) \equiv \neg \forall x : \neg \phi(x),$$

wobei $\phi(x)$ eine Aussage über die Variable x ist, die Existenzquantoren in Allquantoren umwandeln. Dies ist in logarithmischem Platz möglich, da wir von „vorne“ nach „hinten“, also von links nach rechts, durch die Quantoren iterieren und Existenzquantoren negieren, womit auch der nachfolgende Teil der Formel φ negiert wird. Doppelte Negationen heben sich nach den boolschen Axiomen auf, sodass φ nachher von folgender Form ist.

$$\varphi = (\neg)\forall x_D \cdots (\neg)\forall x_1 : (\neg) \bigwedge_{i=1}^{k} \bigvee_{j=1}^{3} L_{i,j}$$

[1] Wie auch Papadimitriou in [Pap93] benennen wir das Problem mit QSAT anstatt mit der verbreiteteren Namensvariante QBF (Erfüllbarkeitsproblem für quantifizierte boolesche Formeln), um den direkten Bezug zu SAT (Erfüllbarkeitsproblem aussagenlogischer Formeln) und die daraus resultierende Hierarchie der Problemstellungen hervorzuheben.

Hier stehen die $(\neg)$ für eventuelle Negationen, die bei der Transformation auftreten können. Außerdem definieren wir

$$\varphi_i = (\neg)\forall x_i \cdots (\neg)\forall x_1 : (\neg) \bigwedge_{i=1}^{k} \bigvee_{j=1}^{3} L_{i,j}$$

mit $i \in \{0,\dots,D\}$, wobei $\varphi_0 = (\neg) \bigwedge_{i=1}^{k} \bigvee_{j=1}^{3} L_{i,j}$. Nun ist φ in einer für uns geeigneteren Form.

Konstruktion des Automaten

Wie bereits in Kapitel 4 werden wir nun einen $\mathcal{G}$-Automat $\mathcal{T}$ sowie ein Zustandswort $q \in Q$ konstruieren, sodass q genau dann auf mindestens einem Wort $w \in \Sigma^*$ nichttrivial operiert, wenn φ wahr ist. Sei im Folgenden $\mathcal{T} = (Q, \Sigma, \delta)$ ein $\mathcal{S}$-Automat. Wie in Kapitel 4 werden wir mit Hilfe der balancierten Kommutatoren $B_{\mathcal{T};\alpha_0,\beta_0}$ eine logische UND-Verknüpfung in ein Zustandswort q kodieren. Hierzu ist Σ erneut eine fünfelementige Menge mit $\Sigma = \Sigma_\top \uplus \Sigma_\bot$ mit $\Sigma_\top = \{a_1, a_2\}$, $\Sigma_\bot = \{a_3, a_4, a_5\}$. Wie auch im vorangegangenen Kapitel bezeichnen wir die Buchstaben aus Σ mit dem Index der Teilmenge, in der sie liegen. Wir werden zunächst einzelne Teilautomaten konstruieren, die wir im Anschluss zum Automaten $\mathcal{T}$ vereinigen werden.

Für jede Klausel K_i in φ mit $k \in \{1,\dots,k\}$ konstruieren wir zunächst den Teilautomaten $\mathcal{T}_i = (Q_i, \Sigma, \delta_i)$ mit $Q_i = \{q_{i,j} \mid j \in \{0,\dots,D\}\} \cup \{\hat{q}_{i,j} \mid j \in \{r+1,\dots,D\}\}$ analog zum Beweis in Kapitel 4.

$$\begin{aligned}
\delta_i = &\{q_{i,j} \xrightarrow{a/a} q_{i,j+1} \mid a \in \Sigma,\, j \in \{0,\dots,D-1\} \setminus \{r,s,t\}\} \\
&\cup \{\hat{q}_{i,j} \xrightarrow{a/a} \hat{q}_{i,j+1} \mid a \in \Sigma,\, j \in \{r+1,\dots,D-1\}\} \\
&\cup \{q_{i,j} \xrightarrow{\top/\top} q_{i,j+1},\, q_{i,j} \xrightarrow{\bot/\bot} \hat{q}_{i,j+1} \mid j \in N\} \\
&\cup \{q_{i,j} \xrightarrow{\bot/\bot} q_{i,j+1},\, q_{i,j} \xrightarrow{\top/\top} \hat{q}_{i,j+1} \mid j \in P\}
\end{aligned}$$

mit $P = \{j \mid j \in \{r,s,t\}, \epsilon_j > 0\}$ und $N = \{j \mid j \in \{r,s,t\}, \epsilon_j < 0\}$. Außerdem konstruieren wir Automaten $\mathcal{T}_\alpha, \mathcal{T}_\beta$ ebenfalls analog zum vorherigen Kapitel und einen weiteren Automaten $\mathcal{T}_\sigma$. Sei $\mathcal{T}_\alpha = (Q_\alpha, \Sigma, \delta_\alpha)$ mit

$$\begin{aligned}
Q_\alpha &= \{\alpha_i \mid i \in \{0, D-1\}\} \cup \{\alpha\} \\
\delta_\alpha &= \{\alpha_i \xrightarrow{a/a} \alpha_{i+1} \mid a \in \Sigma,\, i \in \{0,\dots,D-2\}\} \cup \{\alpha_{D-1} \xrightarrow{a/a} \alpha \mid a \in \Sigma\}
\end{aligned}$$

Die Automaten $\mathcal{T}_\beta$ und $\mathcal{T}_\sigma$ werden analog konstruiert. Um später die Allquantoren in unser Zustandswort kodieren zu können, benötigen wir außerdem Teilautomaten $\mathcal{T}_{t_i} = \{Q_{t_i}, \Sigma, \delta_{t_i}\}$. Dieser Automat soll an der Stelle i im Eingabewort den Eingabebuchstaben invertieren, also ein $a \in \Sigma_\top$ in ein $b \in \Sigma_\bot$ transformieren und umgekehrt. Für jede Variable in φ konstruieren wir so einen Teilautomaten, wobei der Gedanke ist, dass wir für einen Allquantor sowohl eine

Belegung von φ mit 0 als auch mit 1 simulieren werden. Sei also $\mathcal{T}_{t_i} = (Q_{t_i}, \Sigma, \delta_{t_i})$ ein Automat mit

$$
\begin{aligned}
Q_{t_i} &= \{ t_{i,j} \mid j \in \{0, \dots, D\} \} \\
\delta_{t_i} &= \{ t_{i,j} \xrightarrow{a/a} t_{i,j+1} \mid a \in \Sigma, j \in \{0, \dots, i-1\} \} \\
&\quad \cup \{ t_{i,i} \xrightarrow{a/b} t_{i,i+1}, t_{i,i} \xrightarrow{b/a} t_{i,i+1} \mid a \in \Sigma_\top, b \in \Sigma_\bot \} \\
&\quad \cup \{ t_{i,j} \xrightarrow{a/a} t_{i,j+1} \mid a \in \Sigma, j \in \{i+1, \dots, D-1\} \}
\end{aligned}
$$

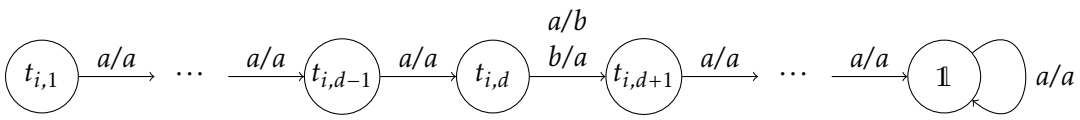


Abbildung 5.1: Graphische Darstellung eines $\mathcal{T}_{t_i}$.

Der Gesamtautomat $\mathcal{T} = (Q, \Sigma, \delta)$ wird nun konstruiert als die Vereinigung aller zuvor konstruierten Teilautomaten. Außerdem fügen wir der Zustandsmenge von $\mathcal{T}$ noch einen weiteren Zustand $\mathbb{1}$ sowie der Übergangsrelation δ weitere Kanten hinzu.

$$
\begin{aligned}
Q &= \biguplus_{i=1}^{k} Q_i \cup \biguplus_{j=1}^{D} Q_{t_j} \cup Q_\alpha \cup Q_\beta \cup Q_\sigma \cup \{\mathbb{1}\} \\
\delta &= \biguplus_{i=1}^{k} \delta_i \cup \biguplus_{j=1}^{D} \delta_{t_j} \cup \delta_\alpha \cup \delta_\beta \cup \delta_\sigma \\
&\quad \cup \{ q_{i,D} \xrightarrow{a/a} \mathbb{1}, \hat{q}_{i,D} \xrightarrow{a/\sigma(a)} \mathbb{1} \mid i \in \{1, \dots, k\}, a \in \Sigma \} \\
&\quad \cup \{ \mathbb{1} \xrightarrow{a/a} \mathbb{1}, \sigma \xrightarrow{a/\sigma(a)} \mathbb{1}, \alpha \xrightarrow{a/\alpha(a)} \mathbb{1}, \beta \xrightarrow{a/\beta(a)} \mathbb{1} \mid a \in \Sigma \} \\
&\quad \cup \{ t_{i,D} \xrightarrow{a/a} \mathbb{1} \mid a \in \Sigma, i \in \{1, D-1\} \} \\
&\quad \cup \{ t_{D,D} \xrightarrow{a/b} \mathbb{1}, t_{D,D} \xrightarrow{b/a} \mathbb{1} \mid a \in \Sigma_\top, b \in \Sigma_\bot \}
\end{aligned}
$$

Wie auch im vorangegangenen Kapitel gilt, dass die Zustände $\hat{q}_{i,D}$ wie das σ aus Lemma 4.2 auf Σ operiert. Die Funktionen α, β und σ verhalten sich ebenfalls so, wie die in Lemma 4.2. Da außer den Zuständen $\hat{q}_{i,D}, \sigma, \alpha, \beta$ alle anderen Zustände nur triviale Zustandsübergänge haben und diese die Buchstaben aus Σ permutieren, sind die induzierten Funktionen aller Zustände in $\mathcal{T}$ Bijektionen und somit invertierbar. Also ist $\mathcal{T}$ ein $\mathcal{G}$-Automat. Der konstruierte Automat ist ebenfalls finitär, da keine Zyklen außer dem Zyklus des trivialen Zustands $\mathbb{1}$ existieren.

Nun, da wir den Automaten $\mathcal{T}$ konstruiert haben, definieren wir einige Teilzustandswörter über Q, aus denen wir letztendlich das Zustandswort q zusammensetzen werden. Seien

$$a_0 = B_{\mathcal{T};\alpha_0,\beta_0}[q_{k,0},\dots,q_{1,0}]$$
$$c_0 = \begin{cases} a_0, & \text{falls } \varphi_0 = \bigwedge_{i=1}^{k}\bigvee_{j=1}^{3} L_{i,j} \\ a_0^{-1}\sigma_0, & \text{falls } \varphi_0 = \neg\left(\bigwedge_{i=1}^{k}\bigvee_{j=1}^{3} L_{i,j}\right) \end{cases}$$

und

$$a_d = B_{\mathcal{T};\alpha_0,\beta_0}[t_d^{-1}c_{d-1}t_d, c_{d-1}]$$
$$c_d = \begin{cases} a_d, & \text{falls } \varphi_d = \forall x_d : \varphi_{d-1} \\ a_d^{-1}\sigma_0, & \text{falls } \varphi_d = \neg\forall x_d : \varphi_{d-1} \end{cases}$$

Dann ist a_D unser gesuchtes Zustandswort. Es gilt nun einerseits zu zeigen, dass a_D auf mindestens einem Wort nichttrivial operiert, genau dann wenn φ wahr ist. Andererseits muss gezeigt werden, dass wir ein SLP S mit $\mathrm{val}(s) = c_D$ in logarithmischem Platz konstruieren können. Zunächst beweisen wir die Korrektheit der Reduktion.

Beweis der Korrektheit der Konstruktion

Zunächst definieren wir eine Hilfsfunktion $\mu : \Sigma \to \{0,1\}$, die die Buchstaben des Alphabets Σ mit ihrem, von uns angedachten, Wahrheitswert in einer Belegung von φ identifiziert.

$$\mu : a \mapsto \begin{cases} 1, & \text{falls } a \in \Sigma_\top \\ 0, & \text{falls } a \in \Sigma_\bot \end{cases}$$

Wir stellen die folgende Invariante auf.

$$\forall w \in \{0,1\}^d : \forall \mathcal{A} : \mathrm{dom}(\mathcal{A}) = \{x_D,\dots,x_{d+1}\} :$$

$$c_d \quad \begin{array}{c} \langle\mathcal{A}\rangle \\ \downarrow \\ \langle\mathcal{A}\rangle \end{array} \quad \begin{array}{c} w \\ \downarrow \\ w \end{array} \quad \begin{cases} \sigma, & \text{falls } \mathcal{A} \models \varphi_d \\ \mathbb{1} & \text{sonst} \end{cases}$$

Mit $\mathrm{dom}(\mathcal{A})$ ist hier, wie bereits in den Grundlagen erklärt, der Definitionsbereich der Belegung $\mathcal{A}$ gemeint, was für eine Formel φ_d den Variablen $x_D,\dots,x_{d+1}$ entspricht, da alle weitere Variablen durch die Formel φ_d gebunden und somit nicht frei belegbar sind. Mit $\langle\mathcal{A}\rangle$ bezeichnen wir im Allgemeinen die Menge aller möglichen Ausprägungen einer solchen Belegungen $\mathcal{A}$, also Wörter aus $\{0,1\}^{|\mathrm{dom}(\mathcal{A})|}$. Da unser Automat hier jedoch über einem Alphabet Σ operiert, dessen Buchstaben wir als Belegungswerte einer aussagenlogischen Formel interpretieren, ist $\langle\mathcal{A}\rangle = \sigma^{|\mathrm{dom}(\mathcal{A})|}$. Mit Hilfe der Invariante soll im Folgenden gezeigt werden, dass die

Formel φ genau dann wahr ist, wenn das Zustandswort auf beliebigen Eingaben nichttrivial operiert. Diese Invariante werden wir nun per vollständiger Induktion über die Anzahl der gebundenen Variablen in φ beweisen. Man beachte auch, dass das „shiften“ des balancierten Kommutators durch das Eingabewort auch hier korrekt funktioniert (vgl. Proposition 4.8).

Induktionsanfang Sei zunächst also $d = 0$. Aufgrund der Definition der c_i müssen wir zwei Fälle unterscheiden.

- Im ersten Fall ist φ_0 von der Form $\bigwedge_{i=1}^{k} \bigvee_{j=1}^{3} L_{i,j}$. Dann gilt, dass $c_0 = a_0 = B_{\mathcal{T};\alpha_0,\beta_0}[q_{k,0},\dots,q_{1,0}]$. Da hier w gleich ε ist, gilt per Konstruktion von $\mathcal{T}_i$, dass

$$q_{i,0} \xrightarrow{\langle\mathcal{A}\rangle} \begin{cases} \sigma, & \text{falls } \mathcal{A} \models L_{i,1} \vee L_{i,2} \vee L_{i,3} \\ \mathbb{1} & \text{sonst} \end{cases}$$

  und es folgt direkt, dass

$$\begin{matrix} \lceil \\ q_{1,0} \\ , \\ q_{2,0} \\ , \\ \vdots \\ , \\ q_{k,0} \\ \lfloor \\ B_{\mathcal{T};\alpha_0,\beta_0} \end{matrix} \xrightarrow{\langle\mathcal{A}\rangle} \begin{matrix} \lceil \\ \sigma \\ , \\ \sigma \\ , \\ \vdots \\ , \\ \sigma \\ \lfloor \\ B_{\mathcal{T};\alpha,\beta} \end{matrix} = \begin{cases} \sigma, & \text{falls } \mathcal{A} \models \bigwedge_{i=1}^{k} K_i \\ \mathbb{1} & \text{sonst.} \end{cases}$$

  Also gilt die Invariante für diesen Fall.

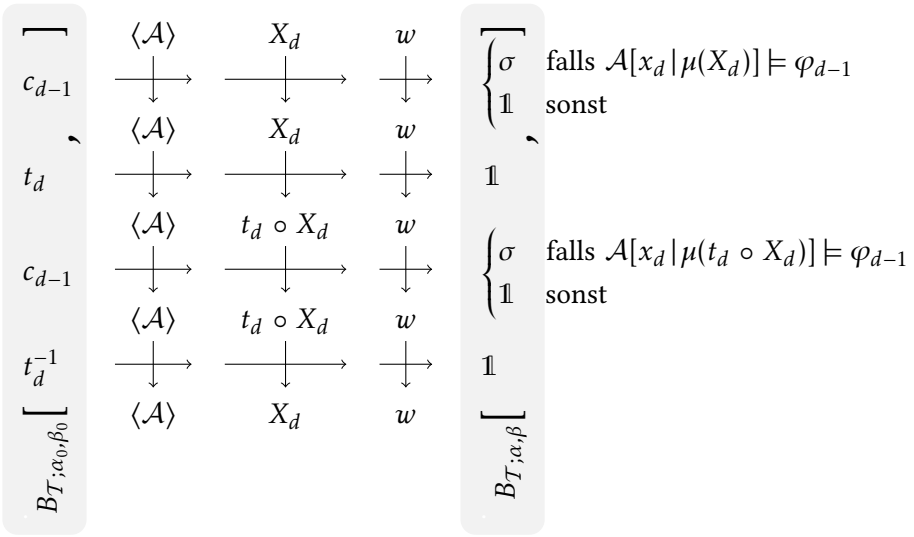
- Im zweiten Fall ist φ_0 von der Form $\neg\left(\bigwedge_{i=1}^{k} \bigvee_{j=1}^{3} L_{i,j}\right)$. Dann gilt, dass $c_0 = a_0^{-1}\sigma_0$. Wie im ersten Fall bereits gezeigt, verhält sich a_0 genau so, dass

$$a_0 \cdot \langle\mathcal{A}\rangle = \begin{cases} \sigma, & \text{falls } \mathcal{A} \models \varphi_0 \\ \mathbb{1} & \text{sonst.} \end{cases}$$

  Daraus folgt, dass wenn $\mathcal{A}$ die Formel $\neg(\varphi_0)$ erfüllt, $a_0 \cdot \langle\mathcal{A}\rangle = \mathbb{1}$ ist und somit $c_0 \cdot \langle\mathcal{A}\rangle = \mathbb{1}^{-1}\sigma = \sigma$ ist. Wenn $\mathcal{A}$ die Formel $\neg(\varphi_0)$ nicht und somit die Formel φ_0 erfüllt, dann ist $a_0 \cdot \langle\mathcal{A}\rangle = \sigma$ und somit $c_0 \cdot \langle\mathcal{A}\rangle = \sigma^{-1}\sigma = \mathbb{1}$. Also ist auch hier die Invariante erfüllt.

Induktionsschritt Sei nun $d > 0$. Auch hier müssen wieder die zwei Fälle wie im Induktionsanfang unterschieden werden.

- Im ersten Fall ist φ_d von der Form $\forall x_d : \varphi_{d-1}$, also gilt $c_d = a_d = B_{\mathcal{T};\alpha_0,\beta_0}[t_d^{-1} c_{d-1} t_d, c_{d-1}]$. Das nachfolgende Kreuzdiagramm verdeutlicht dies.



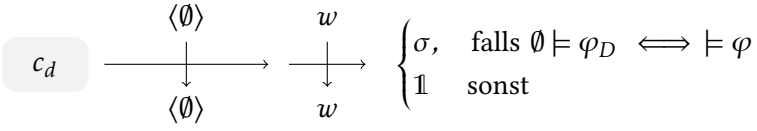
  Man bemerke, dass für ein $X_d \in \Sigma$ gilt, dass $\mu(X_d) = \overline{\mu(t_d \circ X_d)}$ gilt. Da nach Induktionsvoraussetzung die Invariante für $d-1$ gilt und $\mathbb{1}\sigma\mathbb{1} = \sigma$ ist, ist also nach Gleichung (4.2) auch

$$B_{\mathcal{T};\alpha_0,\beta_0}[t_d^{-1} c_{d-1} t_d, c_{d-1}] = \begin{cases} \sigma, & \text{falls } \mathcal{A}[x_d \mid \mu(X_d)] \models \varphi_{d-1} \\ & \quad \text{und } \mathcal{A}[x_d \mid \mu(t_d \circ x_d)] \models \varphi_{d-1} \\ \mathbb{1} & \text{sonst.} \end{cases}$$

- Im zweiten Fall ist φ_d von der Form $\neg\forall x_d : \varphi_{d-1}$, also gilt $c_d = a_d^{-1}\sigma_0$. Es gilt, dass a_d wie im ersten Fall bewiesen genau dann σ ist, wenn beide Belegungen der Variable x_d die Formel φ_{d-1} erfüllen. Wenn φ_d von der Form $\neg\forall x_d : \varphi_{d-1}$ ist, also nicht alle Belegungen die Formel φ_{d-1} erfüllen dürfen, so ist $c_d = a_d^{-1}\sigma^0 = \mathbb{1}\sigma = \sigma$ genau dann, wenn mindestens eine Belegung von x_d die Formel φ_{d-1} nicht erfüllt.

Somit ist die Invariante bewiesen und es gilt, dass

$$\forall w \in \{0,1\}^D : \forall \mathcal{A} : \mathrm{dom}(\mathcal{A}) = \emptyset :$$



□

Beweis der Platzschranke

Da das Zustandswort c_D exponentiell groß werden kann, können wir es nicht in logarithmischem Platz konstruieren. Allerdings können wir ein SLP $S_\mathcal{T}$ angeben, sodass $\mathrm{val}(S_\mathcal{T}) = c_D$ ist, indem wir die rekursive Definition von c_D und des balancierten Kommutators direkt in ein solches SLP übersetzen. Zur Berechnung einer Lösung von $\mathrm{UCWP}(\mathcal{T})$ wird $S_\mathcal{T}$ während der Berechnung ausgewertet und so die Platzschranke nicht überschritten. Für die Reduktion bilden wir also φ auf $(\mathcal{T}, S_\mathcal{T})$ ab.

Da die Konstruktion korrekt ist und die Platzschranke eingehalten wird, ist $\overline{\texttt{UCWPfA}}$ also PSPACE-vollständig. Die deterministischen Platzklassen sind unter Komplement abgeschlossen und deshalb ist UCWPfA ebenfalls PSPACE-vollständig, wie direkt aus der Abschlusseigenschaft folgt. Somit ist Satz 5.1 bewiesen.

6 Zusammenfassung und Ausblick

Mit Hilfe der zunächst definierten Grundlagen in Kapitel 2 konnte in Kapitel 3 zunächst bewiesen werden, dass Automatenmonoide beschränkter Aktivität kontrahierend sind. Diese Eigenschaft wurde anschließend genutzt, um zu zeigen, dass das Wortproblem von Automatenmonoiden beschränkter Aktivität in deterministisch logarithmischem Platz entschieden werden kann. Zwar ist bei dieser Variante des Wortproblems das Berechnen wichtiger Konstanten im Voraus möglich und das Problem aufgrunddessen einfacher als die uniforme Variante. Dennoch bietet sich eine weitere Untersuchung der uniformen Variante des Wortproblems solcher Automatenmonoide anhand der Kontraktionseigenschaft an. Ein weiteres Thema von Interesse wäre die Untersuchung des uniformen Wortproblems von Automatenhalbgruppen beschränkter Aktivität nach der Definition von Bartholdi, Godin, Klimann und Picantin in [BGKP18].

In Kapitel 4 und Kapitel 5 wurde dann das Hauptaugenmerk auf Automatengruppen finitärer Aktivität gelegt. in Kapitel 4 konnte zunächst gezeigt werden, dass das uniforme Wortproblem finitärer Automatengruppen coNP-vollständig ist. Aus diesem Resultat kann direkt abgeleitet werden, dass das uniforme Wortproblem von Automatengruppen beschränkter Aktivität coNP-schwierig ist. Betrachtet man die Polynomialzeit-Hierarchie, so liegt die Vermutung nahe, dass das uniforme Wortproblem höherer Klassen der Aktivitäts-Hierarchie von Automatengruppen zumindest schwierig, wenn nicht vollständig für höhere Klassen der Polynomialzeit-Hierarchie sein könnte. Diese Fragestellung ging über den Horizont der vorliegenden Arbeit hinaus, bietet aber Raum für weitere Forschung. In Kapitel 5 konnte schließlich gezeigt werden, dass das uniforme komprimierte Wortproblem finitärer Automatengruppen PSPACE-vollständig ist. Interessant ist hier vor allem, dass die Konstruktion der Reduktionsfunktion der im vorangegangenen Kapitel stark ähnelt und nur durch Elemente erweitert wurde. Hier war insbesondere die Allquantifizierung der gebundenen Variablen aufgrund der resultierenden, potentiell exponentiell großen Konstruktion des Zustandswortes problematisch. Insgesamt lässt jedoch auch dieses Ergebnis die Fragestellung zu, inwieweit die Aktivitätshierarchie mit der Polynomialzeithierarchie zusammenhängt.

Abschließend lässt sich sagen, dass besonders die Untersuchung der Aktivitätsklassen von Automatenhalbgruppen hinsichtlich ihrer algorithmischen Probleme ein breites Feld für zukünftige Forschung bietet und interessante Resultate erzielt werden können.

Literatur

[Adi57] Sergei I. Adian. „Unsolvability of some algorithmic problems in the theory of groups“. In: *Trudy Moskovskogo Matematicheskogo Obshchestva* 6 (1957), S. 231–298. URL: http://mi.mathnet.ru/mmo62.

[Bar89] David A. Mix Barrington. „Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1“. In: *Journal of Computer and System Sciences* 38.1 (1989), S. 150–164. DOI: 10.1016/0022-0000(89)90037-8.

[BFLW20] Laurent Bartholdi, Michael Figelius, Markus Lohrey und Armin Weiß. „Groups with ALOGTIME-Hard Word Problems and PSPACE-Complete Circuit Value Problems“. In: *35th Computational Complexity Conference (CCC 2020)*. Hrsg. von Shubhangi Saraf. Bd. 169. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 29:1–29:29. ISBN: 978-3-95977-156-6. DOI: 10.4230/LIPIcs.CCC.2020.29. URL: https://drops.dagstuhl.de/opus/volltexte/2020/12581.

[BGKP18] Laurent Bartholdi, Thibault Godin, Ines Klimann und Matthieu Picantin. „A New Hierarchy for Automaton Semigroups“. In: *Implementation and Application of Automata*. Hrsg. von Cezar Câmpeanu. Bd. 10977. Springer International Publishing, 2018, S. 71–83. ISBN: 978-3-319-94812-6. DOI: 10.1007/978-3-319-94812-6_7.

[Bon11] Ievgen V. Bondarenko. „Growth of Schreier graphs of automaton groups“. In: *Mathematische Annalen* 354.2 (2011), S. 765–785. ISSN: 1432-1807. DOI: 10.1007/s00208-011-0757-x. arXiv: 1101.3200.

[Bon14] Ievgen V. Bondarenko. „The word problem in Hanoi Towers groups“. In: *Algebra and Discrete Mathematics* 17.2 (2014), S. 248–255.

[BBSZ13] Ievgen V. Bondarenko, Natalia V. Bondarenko, Said N. Sidki und Flavia R. Zapata. „On the conjugacy problem for finite-state automorphisms of regular rooted trees“. In: *Groups, Geometry and Dynamics* 7.2 (2013), S. 232–355. DOI: 10.4171/GGD/184.

[BW19] Ievgen V. Bondarenko und Jan Philipp Wächter. „On Orbits and the Finiteness of Bounded Automaton Groups“. In: *arXiv preprint* (2019). arXiv: 1101.3200.

[Boo59] William W. Boone. „The Word Problem“. In: *Annals of Mathematics* 70.2 (1959), S. 207–265. ISSN: 0003486X. DOI: 10.2307/1970103.

[Coo71] Stephen A. Cook. „The Complexity of Theorem-Proving Procedures“. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. STOC '71. Shaker Heights, Ohio, USA: Association for Computing Machinery, 1971, S. 151–158. ISBN: 9781450374644. DOI: 10.1145/800157.805047.

[Deh11] Max Dehn. „Über unendliche diskontinuierliche Gruppen“. In: *Mathematische Annalen* 71.1 (1911), S. 116–144. ISSN: 1432-1807. DOI: 10.1007/BF01456932.

[Gri80] Rostislav I. Grigorchuk. „Burnside problem on periodic groups“. Russian. In: *Funktsional'nyi Analiz i ego Prilozheniya* 14.1 (1980), S. 53–54. Englische Übersetzung: „Burnside problem on periodic groups“. In: *Funct. Anal. Appl.* 14 (1 1980), S. 41–43. DOI: 10.1007/BF01078416.

[Gri84] Rostislav I. Grigorchuk. „Degrees of growth of finitely generated groups, and the theory of invariant means“. In: *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya* 48.5 (1984), S. 939–985. ISSN: 0373-2436.

[GP08] Rostislav I. Grigorchuk und Igor Pak. „Groups of intermediate growth: an introduction“. In: *L'Enseignement Mathématique* 54.3-4 (2008), S. 251–272. ISSN: 0013-8584. DOI: 10.5169/seals-109938.

[GŻ01] Rostislav I. Grigorchuk und Andrzej Żuk. „The Lamplighter Group as a Group Generated by a 2-state Automaton, and its Spectrum“. In: *Geometriae Dedicata* 87.1-3 (2001), S. 209–244. ISSN: 1572-9168. DOI: 10.1023/A:1012061801279.

[HRR17] Derek F. Holt, Sarah Rees und Claas E. Röver. *Groups, Languages and Automata*. London Mathematical Society Student Texts. Cambridge University Press, 2017. DOI: 10.1017/9781316588246.

[Loh14] Markus Lohrey. *The compressed word problem for groups*. Springer, New York, NY, 2014. ISBN: 978-1-4939-0747-2. DOI: 10.1007/978-1-4939-0748-9.

[Mak85] Gennadiĭ S. Makanin. „Decidability of the universal and positive theories of a free group“. In: *Mathematics of the USSR-Izvestiya* 25.1 (1985), S. 75–88. DOI: 10.1070/IM1985v025n01ABEH001269.

[Mal62] Anatolij I. Mal'cev. „On the equation zxyx- 1y- 1z- 1= aba- 1b- 1 in a free group“. In: *Algebra i logika* 1.5 (1962), S. 45–50. ISSN: 0373-9252.

[Nek05] Volodymyr V. Nekrashevych. *Self-similar groups*. Mathematical Surveys and Monographs 117. American Mathematical Society, Providence, RI, 2005. ISBN: 978-0-8218-3831-0. DOI: 10.1090/surv/117.

[Nov54] Petr S. Novikov. „Unsolvability of the conjugacy problem in the theory of groups“. In: *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya* 18.6 (1954), S. 485–524.

[Pap93] Christos H. Papadimitriou. *Computational Complexity*. 1. Aufl. Addison-Wesley Publishing Company, Reading, Mass., 1993. ISBN: 0-201-53082-1.

[Sav70] Walter J. Savitch. „Relationships between nondeterministic and deterministic tape complexities“. In: *Journal of Computer and System Sciences* 4.2 (1970), S. 177–192. ISSN: 0022-0000. DOI: 10.1016/S0022-0000(70)80006-X.

[Sid00] Said N. Sidki. „Automorphisms of one-rooted trees: growth, circuit structure, and acyclicity“. In: *Journal of Mathematical Sciences* 100.1 (2000), S. 1925–1943. DOI: 10.1007/BF02677504.

[Ste15] Benjamin Steinberg. „On some algorithmic properties of finite state automorphisms of rooted trees“. In: *Contemp. Math., Algorithmic problems of group theory, their complexity, and applications to cryptography* 633 (2015), S. 115–123. DOI: 10.1090/conm/633/12655.

[ŠV12] Zoran Šunić und Enric Ventura. „The conjugacy problem in automaton groups is not solvable“. In: *Journal of Algebra* 364 (2012), S. 148–154. ISSN: 0021-8693. DOI: 10.1016/j.jalgebra.2012.04.014.

[WW20] Jan Philipp Wächter und Armin Weiß. „An Automaton Group with PSPACE-Complete Word Problem“. In: *37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)* (Montpellier, France, 10.–13. März 2020). Hrsg. von Christophe Paul und Markus Bläser. Bd. 154. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 4. März 2020, 6:1–6:17. ISBN: 978-3-95977-140-5. DOI: 10.4230/LIPIcs.STACS.2020.6. arXiv: 1906.03424 [cs.FL].

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