Institut für Formale Methoden der Informatik

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit Nr. 18

Minimierung und effiziente Algorithmen für erkennende Homomorphismen über omega-regulären Sprachen

Lukas Fleischer

Studiengang:	Informatik
Prüfer:	PD Dr. Manfred Kufleitner
Betreuer:	PD Dr. Manfred Kufleitner
Beginn am:	26. November 2014
Beendet am:	28. Mai 2015
CR-Klassifikation:	F.2.2, F.4.1, F.4.3

Kurzfassung

Automaten über unendlichen Wörtern sind ein wichtiges Werkzeug für die Modellierung nicht-terminierender Systeme. Häufig werden Systemspezifikationen und Systemeigenschaften durch sogenannte Büchi-Automaten beschrieben, da in dieser Darstellung viele Entscheidbarkeitsfragen einfach beantwortet werden können. Eine Schwierigkeit bei der Handhabung von Büchi-Automaten ist jedoch die Komplexität der Komplementbildung und der Minimierung; zwei Operationen, die in der Praxis bei der Überführung von logischen Spezifikationen in Automaten benötigt werden. Homomorphismen zwischen freien und endlichen Halbgruppen sind ein alternativer Formalismus, um die Klasse der von Büchi-Automaten akzeptierten Sprachen, die sogenannten ω-regulären Sprachen, zu beschreiben. Dort sind, im Gegensatz zu Büchi-Automaten, die Komplementierung und die Minimierung mit geringem Berechnungsaufwand möglich. In dieser Arbeit werden zunächst grundlegende Algorithmen für verschiedene Operationen und Entscheidbarkeitsprobleme auf erkennenden Homomorphismen erarbeitet. Anschließend wird der Einsatz von Homomorphismen in der Praxis getestet.

Inhaltsverzeichnis

1 Einleitung

Bereits in den Anfängen der Informatik wurden endliche Automaten als einfaches Berechnungsmodell untersucht. Die erste formale Beschreibung wird McCulloch und Pitts zugeschrieben, die 1943 in ihrer Arbeit über neuronale Netze [MP43] ein dem heutigen Automatenbegriff ähnliches Modell einführten. In den 1950er-Jahren wurde dieses Konzept abstrahiert und von einem theoretischen Standpunkt aus untersucht. Kleenes Theorem über reguläre Sprachen [Kle56] wird häufig als der Beginn der Automatentheorie angesehen.

Eine Erweiterung auf unendliche Wörter erfolgte 1962 im Kontext eines Entscheidbarkeitsresultats der mathematischen Logik [Büc62]. Büchi konnte zeigen, dass das von ihm eingeführte Modell der *Büchi-Automaten* dieselbe Ausdrucksstärke wie monadische Prädikatenlogik zweiter Stufe (engl. *monadic second-order logic*, MSO) über unendlichen Wörtern besitzt. Beide Formalismen können zur Beschreibung von sogenannten *ω-regulären Sprachen* verwendet werden. Bis heute ist dieser Zusammenhang einer der Hauptgründe für das Interesse an Büchi-Automaten und verwandten Automatenmodellen. Mit dem Instrumentarium der Logik können viele Eigenschaften von Systemen auf natürliche Art und Weise formalisiert werden. Die Umwandlung in eine Darstellung durch Automaten ermöglicht die Beantwortung wichtiger Entscheidungsfragen. Praktische Anwendung findet dieser Zusammenhang beispielsweise bei der vollautomatischen Verifikation von Systembeschreibungen, oft als *Model Checking* bezeichnet.

Mit der Einführung des Begriffes der *syntaktischen Halbgruppe* durch Schützenberger [Sch56] wurde 1956 der Grundstein für eine algebraische Sicht auf endliche Automaten und die Klasse der regulären Sprachen gelegt. Die Erweiterung auf unendliche Wörter erfolgte 1985 durch Arnold [Arn85]. Das Bindeglied zwischen der Sprachebene und Halbgruppen sind sogenannte *Homomorphismen*. Daher wird die Beschreibung einer regulären oder ω-regulären Sprache durch einen Homomorphismus in eine endliche Halbgruppe auch als *erkennender Homomorphismus* bezeichnet.

Während der algebraische Ansatz zu einigen wichtigen theoretischen Resultaten führte, konnte er sich bei praktischen Implementierungen bisher nicht durchsetzen. Dies ist im Fall von endlichen Wörtern auf die Überlegenheit des automatentheoretischen Ansatzes zurückzuführen. Endliche Automaten sind häufig kompakter als die zugehörigen syntaktischen Halbgruppen und alle für die Umwandlung von logischen Formeln relevanten Operationen lassen sich dort effizient implementieren. Anders verhält es sich jedoch bei unendlichen Wörtern: Die Komplementierung und Minimierung von Büchi-Automaten ist mit einem enormen Berechnungsaufwand verbunden. Michel konnte 1988 zeigen, dass es Büchi-Automaten mit n Zuständen gibt, für die der kleinste das Komplement erkennende Automat mindestens $(n-1)!$ Zustände besitzt [Mic88]. Die Berechnung eines minimalen Büchi-Automaten zu einer gegebenen ω-regulären Sprache ist als Spezialfall des Universalitätsproblems PSPACE-hart [MS72, SVW87]. Dementgegen sind

in der Darstellung durch erkennende Homomorphismen auch bei unendlichen Wörtern alle benötigten Operationen effizient durchführbar. Der im Fall von endlichen Wörtern so abwegige Einsatz von Halbgruppen könnte sich in der Praxis also durchaus als konkurrenzfähiger Ansatz herausstellen. Diese Arbeit stellt eine Sammlung von grundlegenden, für die Realisierung dieser Idee notwendigen Algorithmen zur Verfügung und untersucht diese auf Praxistauglichkeit.

Zunächst werden in Kapitel 2 die algebraischen Grundlagen eingeführt und Konventionen zur Form der Eingabe festgelegt. Aus algebraischer Sicht sind für die hier beschriebenen Verfahren vor allem die Begriffe der *linked Pairs* und der *Konjugationsklassen* von linked Pairs von großer Bedeutung. In Kapitel 3 werden einige grundlegende Eigenschaften dieser Objekte untersucht und Abschätzungen für deren Größe gegeben. Anschließend werden Verfahren zur effizienten Berechnung der zu einer Halbgruppe zugehörigen linked Pairs und Konjugationsklassen vorgestellt. Kapitel 4 beschreibt, wie die Operationen, die bei der Umwandlung einer Formel aus monadischer Logik zweiter Stufe in erkennende Homomorphismen benötigt werden, effektiv auf der Ebene von Halbgruppen und Homomorphismen berechnet werden können. Verfahren zum Lösen verschiedener Entscheidungsprobleme, wie dem *Leerheitsproblem*, dem *Universalitätsproblem*, dem *Inklusions-* und *Äquivalenzproblem*, werden in Kapitel 5 eingeführt. In Kapitel 6 wird schließlich ein effizientes Minimierungsverfahren für erkennende Homomorphismen hergeleitet. Abschließend enthält Kapitel 7 einige Testergebnisse, die aus der Implementierung einer Software zur Umwandlung von MSO-Formeln in erkennende Homomorphismen stammen.

2 Grundlagen

2.1 Wörter und Sprachen

Als *Alphabet* bezeichnet man eine beliebige endliche, nicht-leere Menge. Die Elemente eines Alphabets heißen *Buchstaben*. Ein *endliches Wort* über einem Alphabet Σ ist eine endliche Sequenz $a_1 a_2 \cdots a_n$ von Buchstaben $a_1, a_2, \ldots, a_n \in \Sigma$. Die Menge aller in einem endlichen Wort $u = a_1 a_2 \cdots a_n$ vorkommenden Buchstaben $\{ a_1, a_2, \ldots, a_n \}$ wird mit $\mathrm{alph}(u)$ bezeichnet und die *Länge* von u ist $|u| = n$. Für das eindeutige *leere Wort* mit Länge 0 wird die Notation ε verwendet. Mit Σ^+ wird die Menge $\{ a_1 \cdots a_n \mid n \geqslant 1, a_i \in \Sigma \text{ für alle } i \text{ mit } 1 \leqslant i \leqslant n \}$ aller nicht-leeren, endlichen Wörter und mit Σ^* die Menge $\Sigma^+ \cup \{ \varepsilon \}$ aller endlichen Wörter über Σ bezeichnet. Eine Teilmenge von Σ^* heißt *Sprache* über dem Alphabet Σ.

Jedes endliche Wort lässt sich auch als Abbildung $u \colon \{ 1, \ldots, n \} \to \Sigma$ von Positionen auf Buchstaben auffassen. Diese Betrachtung liefert eine natürliche Erweiterung auf unendliche Wörter, indem man Abbildungen mit abzählbar unendlichem Definitionsbereich betrachtet. Eine Abbildung $\alpha \colon \mathbb{N} \to \Sigma$ heißt *unendliches Wort* über dem Alphabet Σ und wird üblicherweise mit $a_1 a_2 \cdots$ bezeichnet, wobei $a_i = \alpha(i)$ für alle $i \in \mathbb{N}$. Wie bei endlichen Wörtern werden mit $\mathrm{alph}(\alpha) = \{ a_1, a_2, \ldots \}$ die in α vorkommenden Buchstaben beschrieben. Auch wenn α ein unendliches Wort ist, ist die Größe von $\mathrm{alph}(\alpha)$ durch $|\Sigma|$ beschränkt. Analog zu Σ^* und Σ^+ beschreibt Σ^ω die Menge aller unendlichen Wörter über Σ und Teilmengen von Σ^ω heißen entsprechend *ω-Sprachen*. Im Kontext von unendlichen Wörtern verzichtet man oft auf den Präfix ω- und verwendet für Teilmengen von Σ^ω ebenfalls die Bezeichnung *Sprache*. Da in dieser Arbeit in erster Linie Sprachen über unendlichen Wörtern betrachtet werden, wird dieser Konvention gefolgt, sofern dadurch keine Missverständnisse auftreten.

Für zwei endliche Wörter $u = a_1 a_2 \cdots a_n$ und $v = b_1 b_2 \cdots b_m$ ist $uv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m$ das *Produkt* von u und v. Analog lässt sich auch das Produkt $u\alpha$ von einem endlichen Wort $u \in \Sigma^*$ und einem unendlichen Wort $\alpha \in \Sigma^\omega$ definieren. Für ein endliches Wort $u \in \Sigma^+$ sei zudem $u^\omega = uuu \cdots$ das unendliche Produkt, das durch die Iteration des Faktors u entsteht. Eine natürliche Erweiterung auf das leere Wort ist $\varepsilon^\omega = \varepsilon$.

Die oben beschriebenen Operationen lassen sich wie folgt auf Sprachen erweitern: Für $K \subseteq \Sigma^*$ und $L \subseteq \Sigma^*$ oder $L \subseteq \Sigma^\omega$ ist $KL \coloneqq \{ uv \mid u \in K, v \in L \}$. Das Produkt KL ist genau dann eine ω-Sprache, wenn es sich bereits bei L um eine ω-Sprache handelt. Für eine Teilmenge $L \subseteq \Sigma^+$ ist $L^\omega = \{ u_1 u_2 \cdots \mid u_i \in L \text{ für alle } i \geqslant 1 \}$.

2.2 Büchi-Automaten und ω-reguläre Sprachen

Ein *Büchi-Automat* ist ein 5-Tupel $\mathcal{A} = (Q, \Sigma, \delta, I, F)$. Die Menge der *Zustände* Q und das *Alphabet* Σ sind jeweils endlich, die sogenannte *Übergangsrelation* δ ist eine Teilmenge von $Q \times \Sigma \times Q$. Elemente $(q, a, q') \in \delta$ heißen *Transitionen*. Zudem sind I und F Teilmengen von Q und heißen *Startzustände* bzw. *Endzustände* von $\mathcal{A}$.

Ein *(endlicher) Pfad* ist eine Sequenz der Form $q_0 a_1 q_1 a_2 q_2 \cdots q_{n-1} a_n q_n$ mit $n \geqslant 0$ und $(q_i, a_{i+1}, q_{i+1}) \in \delta$ für alle $i \in \{ 0, \cdots, n-1 \}$. Das endliche Wort $a_1 a_2 \cdots a_n$ heißt *Beschriftung* des Pfades. Für ein unendliches Wort $\alpha = a_1 a_2 \cdots$ wird eine unendliche Sequenz der Form $q_0 a_1 q_1 a_2 q_2 \cdots$ mit $(q_i, a_{i+1}, q_{i+1}) \in \delta$ für alle $i \geqslant 0$ als *Lauf* von α auf $\mathcal{A}$ bezeichnet. Häufig werden die Notationen

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 - \cdots \to q_{n-1} \xrightarrow{a_n} q_n \quad \text{bzw.} \quad q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 - \cdots$$

für Pfade bzw. Läufe verwendet, um Zustände und Beschriftung optisch voneinander zu trennen. Ein Lauf $r = q_0 a_1 q_1 a_2 q_2 \cdots$ heißt *akzeptierend*, falls er in einem Startzustand $q_0 \in I$ startet und $q_i \in F$ für unendlich viele $i \geqslant 1$. Äquivalent dazu ist die Forderung, dass ein Endzustand existiert, der unendlich oft im Lauf auftaucht. Die von $\mathcal{A}$ *akzeptierte Sprache* $L(\mathcal{A})$ ist die Menge aller unendlichen Wörter, die einen akzeptierenden Lauf auf $\mathcal{A}$ haben, und eine beliebige ω-Sprache L heißt *ω-regulär*, falls ein Büchi-Automat $\mathcal{A}$ mit $L(\mathcal{A}) = L$ existiert.

2.3 Erkennbarkeit durch endliche Halbgruppen

Eine *Halbgruppe* ist eine mit einer assoziativen Verknüpfung ausgestattete Menge. Falls diese Menge ein neutrales Element bezüglich der Verknüpfung enthält, so wird sie als *Monoid* bezeichnet. Für eine Halbgruppe S bezeichnet S^1 das Monoid S selbst, falls S bereits ein Monoid ist, und ansonsten die Erweiterung $S \cup \{ 1 \}$ von S um ein neutrales Element 1. Eine Teilmenge $T \subseteq S$ wird *Unterhalbgruppe* von S genannt, wenn T selbst eine (abgeschlossene) Halbgruppe bildet. Das *direkte Produkt* $S \times T$ endlicher Halbgruppen S und T ist die Produktmenge $S \times T$, ausgestattet mit der komponentenweisen Verknüpfung.

Ein Element $e \in S$ heißt *idempotent*, falls $e^2 = e$ bezüglich der Verknüpfung von S. In einer endlichen Halbgruppe ist für jedes Element $s \in S$ die Menge $\{ s, s^2, s^3, \dots \}$ aller Potenzen von s endlich und sie enthält genau ein idempotentes Element [Pin86, Proposition 1.1.6]. Dieses eindeutige idempotente Element wird im Folgenden mit s^π bezeichnet. Ferner sei $E(S) \coloneqq \{ e \in S \mid e^2 = e \}$ die Menge aller idempotenten Elemente der Halbgruppe S.

Beispiel 2.1. *Für alle $e \in E(S)$ bildet die Menge $Se = \{ se \mid s \in S \}$ eine Unterhalbgruppe von S, denn aufgrund der Assoziativität der Verknüpfung gilt für alle $s, t \in S$ die Gleichung $se \cdot te = (set)e$ und aufgrund der Abgeschlossenheit ist $set \in S$.*

Beispiel 2.2. *Die Menge Σ^+ formt mit dem Produkt von Wörtern eine Halbgruppe. Durch die Ergänzung von Σ^+ um eine neutrales Element erhält man das Monoid Σ^*. Letzteres enthält außer dem neutralen Element ε keine idempotenten Elemente.*

Als *Halbgruppenhomomorphismus* (oder auch einfach *Homomorphismus*, sofern aus dem Kontext hervorgeht, dass es sich um Halbgruppen handelt) bezeichnet man eine strukturerhaltende Abbildung $\varphi\colon S \to T$ von einer Halbgruppe S in eine Halbgruppe T. Strukturerhaltend bedeutet in diesem Zusammenhang, dass $\varphi(s) \cdot \varphi(t) = \varphi(st)$ für alle $s, t \in S$. Die unendliche Halbgruppe Σ^+ wird als *freie Halbgruppe* bezeichnet und erfüllt folgende universale Eigenschaft:

Proposition 2.3. *Sei S eine beliebige Halbgruppe, dann kann jede Abbildung $f\colon \Sigma \to S$ eindeutig zu einem Homomorphismus $\varphi\colon \Sigma^+ \to S$ mit $\varphi(a) = f(a)$ für alle $a \in \Sigma$ ergänzt werden.*

Beweis. Die natürliche Erweiterung $\varphi(a_1 a_2 \cdots a_n) = f(a_1)f(a_2)\cdots f(a_n)$ für $n \geqslant 1$ und $a_i \in \Sigma$ liefert einen Homomorphismus. Umgekehrt gilt für jeden Homomorphismus $\psi\colon \Sigma^+ \to S$ mit $\psi(a) = f(a)$ bereits $\psi(a_1 a_2 \cdots a_n) = \psi(a_1)\psi(a_2)\cdots\psi(a_n) = f(a_1)f(a_2)\cdots f(a_n)$. □

Eine reflexive, transitive und symmetrische Relation $\equiv$ auf einer Menge S wird *Äquivalenzrelation* genannt. Sie teilt S in disjunkte Untermengen, sogenannte *Äquivalenzklassen* ein. Die Menge aller Äquivalenzklassen wird nachfolgend durch $S/\equiv$ bezeichnet. Seien $\mathcal{R}_1$ und $\mathcal{R}_2$ Äquivalenzrelationen über einer Menge S, so ist $\mathcal{R}_1$ *feiner* als $\mathcal{R}_2$ (oder $\mathcal{R}_1$ *verfeinert* $\mathcal{R}_2$ bzw. $\mathcal{R}_2$ ist *gröber* als $\mathcal{R}_1$), falls für alle $s, t \in S$ aus $s\ \mathcal{R}_1\ t$ folgt, dass $s\ \mathcal{R}_2\ t$.

Sei nun S eine Halbgruppe. Eine Äquivalenzrelation $\equiv$ auf S heißt *linksverträglich*, falls für alle $s, t, p \in S$ aus $s \equiv t$ folgt, dass $ps \equiv pt$. Sie heißt *rechtsverträglich*, falls für alle $s, t, q \in S$ aus $s \equiv t$ folgt, dass $sq \equiv tq$. Eine Äquivalenzrelation, die linksverträglich und rechtsverträglich ist, wird *Kongruenzrelation* oder *Kongruenz* genannt. Falls S eine Halbgruppe und $\equiv$ eine Kongruenz ist, bildet die Menge $S/\equiv$ ebenfalls eine Halbgruppe mit der durch S induzierten Multiplikation, den sogenannten *Quotienten* von S nach $\equiv$. Um dies zu sehen, muss lediglich verifiziert werden, dass der Quotient wohldefiniert ist, das Ergebnis einer Multiplikation also nicht von der Wahl der Vertreter abhängt. Dies folgt direkt aus der Definition einer Kongruenz: Seien $s, s', t, t' \in S$ mit $s \equiv t$ und $s' \equiv t'$, so gilt aufgrund der Links- und Rechtsverträglichkeit $ss' \equiv st' \equiv tt'$. Eine weitere Beobachtung ist, dass die Abbildung $\varphi\colon S \to S/\equiv$ von Elementen auf die zugehörigen Äquivalenzklassen ein Homomorphismus ist.

2.3.1 Cayley-Graphen und Greens Relationen

Ein wichtiges Werkzeug für das Studium von Halbgruppen sind *Greens Relationen*. Sei S eine Halbgruppe und seien $s, t \in S$. Dann ist

$$
\begin{aligned}
&s \leqslant_{\mathcal{L}} t \text{ falls ein } p \in S^1 \text{ existiert, sodass } s = p \cdot t\\
&s \leqslant_{\mathcal{R}} t \text{ falls ein } q \in S^1 \text{ existiert, sodass } s = t \cdot q\\
&s \leqslant_{\mathcal{J}} t \text{ falls } p, q \in S^1 \text{ existieren, sodass } s = p \cdot t \cdot q\\
&s \leqslant_{\mathcal{H}} t \text{ falls } s \leqslant_{\mathcal{L}} t \text{ und } s \leqslant_{\mathcal{R}} t
\end{aligned}
$$

Diese Quasiordnungen können wie folgt zu Äquivalenzrelationen $\mathcal{L}$, $\mathcal{R}$, $\mathcal{J}$ auf S erweitert werden:

$$s \mathcal{L} t \text{ falls } s \leqslant_{\mathcal{L}} t \text{ und } t \leqslant_{\mathcal{L}} s$$
$$s \mathcal{R} t \text{ falls } s \leqslant_{\mathcal{R}} t \text{ und } t \leqslant_{\mathcal{R}} s$$
$$s \mathcal{J} t \text{ falls } s \leqslant_{\mathcal{J}} t \text{ und } t \leqslant_{\mathcal{J}} s$$
$$s \mathcal{H} t \text{ falls } s \leqslant_{\mathcal{H}} t \text{ und } t \leqslant_{\mathcal{H}} s$$

Die Äquivalenzklassen der Relation $\mathcal{L}$ (bzw. $\mathcal{R}$, $\mathcal{J}$, $\mathcal{H}$) heißen *$\mathcal{L}$-Klassen* (bzw. *$\mathcal{R}$-Klassen*, *$\mathcal{J}$-Klassen*, *$\mathcal{H}$-Klassen*) von S. Eine Halbgruppe S heißt *$\mathcal{L}$-trivial* (bzw. *$\mathcal{R}$-trivial*, *$\mathcal{J}$-trivial*, *$\mathcal{H}$-trivial*), falls für alle $s, t \in S$ genau dann $s \mathcal{L} t$ (bzw. $s \mathcal{R} t$, $s \mathcal{J} t$, $s \mathcal{H} t$) gilt, wenn $s = t$.

Das Studium vom Greens Relationen führte zu einigen bemerkenswerten Ergebnissen in der Theorie endlicher Halbgruppen. Für Details sei auf [Pin86] verwiesen. Ein Resultat, das später noch benötigt wird, ist folgendes (vgl. [Pin86, Proposition 3.1.4]):

Lemma 2.4. *Seien s und t Elemente einer endlichen Halbgruppe S. Dann ist $t \leqslant_{\mathcal{J}} st$ genau dann, wenn $t \mathcal{L} st$.*

Beweis. Sei zunächst $t \leqslant_{\mathcal{J}} st$. Dann existieren nach der Definition von Greens Relationen $p, q \in S^1$ mit $pstq = t$. Durch Iteration dieser Gleichung folgt $(ps)^n tq^n = t$ für alle $n \in \mathbb{N}$ und damit insbesondere $etf = t$ für $e = (ps)^\pi$ und $f = q^\pi$. Sei nun $k \geqslant 1$, sodass $(ps)^k ps = e$. Dann ist $(ps)^k p(st) = et = eetf = etf = t$, also $t \leqslant_{\mathcal{L}} st$. Die andere Richtung folgt direkt aus der Beobachtung, dass $\mathcal{L}$ die Relation $\mathcal{J}$ verfeinert. □

Mit einer ähnlichen Argumentation kann das folgende Resultat gezeigt werden (vgl. [Pin86, Proposition 3.1.3]):

Lemma 2.5. *Seien s und t Elemente einer endlichen Halbgruppe S. Dann ist $s \mathcal{J} t$ genau dann, wenn ein $x \in S$ mit $s \mathcal{R} x \mathcal{L} t$ existiert.*

Beweis. Sei S eine endliche Halbgruppe und seien $s, x, t \in S$ mit $s \mathcal{R} x \mathcal{L} t$. Nach der Definition von Greens Relationen existieren $p, p', q, q' \in S^1$, sodass $px = t$, $p't = x$, $sq = x$ und $xq' = s$. Es folgt $psq = px = t$ und $p'tq' = xq' = s$, also gilt $s \mathcal{J} t$.

Seien nun umgekehrt $s, t \in S$ mit $s \mathcal{J} t$. Dann existieren $p, q, p', q' \in S^1$ mit $psq = t$ und $p'tq' = s$. Folglich gilt $(p'p)^n s(qq')^n = s$ für alle $n \in \mathbb{N}$ und damit insbesondere $esf = s$ für $e = (p'p)^\pi$ und $f = (qq')^\pi$. Sei nun $x \coloneqq sq$ und $k \geqslant 1$, sodass $(p'p)^k p'p = e$. Dann ist $px = psq = t$ und $(p'p)^k p't = (p'p)^k p'psq = esq = eesfq = esfq = sq = x$, also $t \mathcal{L} x$. Eine duale Betrachtung liefert $x \mathcal{R} s$. □

Weitere Objekte, die Einsicht in die Struktur einer Halbgruppe gewähren, sind *Cayley-Graphen*. Der *(rechte) Cayley-Graph* eines surjektiven Homomorphismus $\varphi\colon \Sigma^+ \to S$ ist ein gerichteter, kantenbeschrifteter Graph, bestehend aus der Knotenmenge S und beschrifteten Kanten $E = \{ (s, a, s\varphi(a)) \mid s \in S, a \in \Sigma \}$. Oft bietet es sich an, ein neutrales Element 1 hinzuzufügen und die Knotenmenge S^1 zu betrachten, um einen gemeinsamen Ausgangsknoten zu erhalten,

von dem aus alle anderen Knoten erreichbar sind. Auf analoge Weise lassen sich der *linke Cayley-Graph* und der *zweiseitige Cayley-Graph* definieren, indem man die Kantenmenge $E' = \{ (s, a, \varphi(a)s) \mid s \in S, a \in \Sigma \}$ bzw. $E \cup E'$ betrachtet. Im zweiseitigen Cayley-Graph werden häufig zusätzliche Kantenmarkierungen eingefügt, um zwischen Kanten aus E und Kanten aus E' zu unterscheiden.

Folgende Beobachtung erlaubt es, die Äquivalenzklassen von Greens Relationen mithilfe des Algorithmus von Tarjan [Tar72] in Linearzeit $\mathcal{O}(|\Sigma| \cdot |S|)$ zu berechnen:

Proposition 2.6. *Sei S eine endliche Halbgruppe und $\varphi\colon \Sigma^+ \to S$ ein surjektiver Homomorphismus. Zwei Elemente $s, t \in S$ liegen genau dann in derselben starken Zusammenhangskomponente des rechten (bzw. linken, zweiseitigen) Cayley-Graphen von φ, wenn $s \mathcal{R} t$ (bzw. $s \mathcal{L} t$, $s \mathcal{J} t$).*

Beweis. Es gilt $s \mathcal{R} t$ genau dann, wenn $q, q' \in S^1$ mit $s = t \cdot q$ und $t = s \cdot q'$ existieren. Dies ist genau dann der Fall, wenn im rechten Cayley-Graph von φ ein mit q beschrifteter Pfad von s nach t, sowie ein mit q' beschrifteter Pfad von t nach s existieren, was wiederum äquivalent dazu ist, dass s und t in derselben starken Zusammenhangskomponente liegen. Für die anderen beiden Aussagen kann dasselbe Argument verwendet werden. □

2.3.2 Gruppen

Sei M ein Monoid mit neutralem Element 1 und sei $x \in M$. Ein Element $y \in M$ heißt *rechtsinvers* zu x, falls $xy = 1$, und *linksinvers* zu x, falls $yx = 1$. Ein Monoid heißt *Gruppe*, falls jedes Element ein Linksinverses und ein Rechtsinverses besitzt. Äquivalent dazu ist die Forderung $x \mathcal{H} 1$ für alle $x \in M$.

Später wird folgendes elementare Resultat aus der Gruppentheorie benötigt:

Proposition 2.7. *Sei S eine Halbgruppe und e ein Element aus S, sodass $se = s$ für alle $s \in S$. Existiert für jedes Element $s \in S$ ein Element $\overline{s} \in S$ mit $s\overline{s} = e$, so ist S eine Gruppe.*

Beweis. Wie in der Voraussetzung soll für ein Element $s \in S$ in diesem Beweis mit $\overline{s}$ ein Element mit $s\overline{s} = e$ bezeichnet werden. Nun ist $\overline{s}s\overline{s}s = \overline{s}es = \overline{s}s$ und folglich $\overline{s}s = \overline{s}se = \overline{s}s(\overline{s}s)\overline{(\overline{s}s)} = (\overline{s}s)\overline{(\overline{s}s)} = e$. Ferner gilt $es = s\overline{s}s = se = s$. Also ist e ein neutrales Element bezüglich der Verknüpfung und für jedes $s \in S$ existiert ein $\overline{s}$, das sowohl links- als auch rechtsinvers zu s ist. □

2.3.3 Schwache Erkennbarkeit

Im Fall von endlichen Wörtern können endliche Halbgruppen als zu endlichen Automaten äquivalentes Akzeptanzmodell für reguläre Sprachen betrachtet werden. Ein ähnlicher Zusammenhang gilt auch für ω-reguläre Sprachen. Hierfür wird zunächst folgendes Resultat benötigt, das eine direkte Konsequenz aus einem Theorem von Ramsey [Ram29] ist:

Proposition 2.8. *Seien $u, v_1, v_2, \ldots \in \Sigma^+$ endliche Wörter, S eine endliche Halbgruppe und $\varphi\colon \Sigma^+ \to S$ ein Homomorphismus. Dann existieren Indizes $i_1, i_2, \ldots$ und Elemente $s, e \in S$, sodass $\varphi(uv_1v_2\cdots v_{i_1}) = s$ und $\varphi(v_{i_j+1}v_{i_j+2}\cdots v_{i_k}) = e$ für alle $k > j \geqslant 1$.*

Aus dieser Aussage lassen sich weitere Eigenschaften des Paares (s, e) ableiten. Zunächst gilt $e^2 = e \cdot e = \varphi(v_{i_1+1}v_{i_1+2}\cdots v_{i_2}) \cdot \varphi(v_{i_2+1}v_{i_2+2}\cdots v_{i_3}) = \varphi(v_{i_1+1}v_{i_1+2}\cdots v_{i_3}) = e$, also ist das Element e idempotent. Zudem hat $s' = \varphi(uv_1v_2\cdots v_{i_2})$ die Eigenschaft, dass $s'e = see = se = s'$. Diese beiden Beobachtungen führen auf den folgenden Begriff: Ein Paar $(s, e) \in S \times S$ heißt *linked Pair* von S, falls $se = s$ und $e^2 = e$. Die Menge aller linked Pairs von S wird mit $F(S)$ bezeichnet.

Korollar 2.9. *Seien $u, v_1, v_2, \ldots \in \Sigma^+$ endliche Wörter, S eine endliche Halbgruppe und $\varphi\colon \Sigma^+ \to S$ ein Homomorphismus. Dann existieren Indizes $i_1, i_2, \ldots$ und ein linked Pair (s, e) von S, sodass $\varphi(uv_1v_2\cdots v_{i_1}) = s$ und $\varphi(v_{i_j+1}v_{i_j+2}\cdots v_{i_k}) = e$ für alle $k > j \geqslant 1$.*

Eine weitere unmittelbare Konsequenz ist folgendes Resultat zur Existenz bestimmter Zerlegungen unendlicher Wörter:

Korollar 2.10. *Seien $\alpha \in \Sigma^\omega$ ein unendliches Wort, S eine endliche Halbgruppe und $\varphi\colon \Sigma^+ \to S$ ein Homomorphismus. Dann existiert eine Zerlegung $\alpha = uv_1v_2\cdots$ mit $u, v_1, v_2, \ldots \in \Sigma^+$ und $\varphi(u) = s$, sowie $\varphi(v_i) = e$ für alle $i \geqslant 1$ und ein linked Pair (s, e) von S.*

Eine etwas andere Sichtweise, die zur selben Aussage führt, ist folgende:

Lemma 2.11. *Seien s und t Elemente einer endlichen Halbgruppe S und $\varphi\colon \Sigma^+ \to S$ ein Homomorphismus. Dann ist $\varphi^{-1}(s)(\varphi^{-1}(t))^\omega \subseteq \varphi^{-1}(st^\pi)(\varphi^{-1}(t^\pi))^\omega$.*

Beweis. Sei $n \geqslant 1$ mit $t^n = t^\pi$ und $uv_1v_2\cdots$ ein unendliches Wort mit $u, v_1, v_2, \ldots \in \Sigma^+$ und $\varphi(u) = s$, sowie $\varphi(v_i) = t$ für alle $i \geqslant 1$. Dann ist $\varphi(uv_1v_2\cdots v_n) = st^\pi$ und $\varphi(v_{kn+1}v_{kn+2}\cdots v_{kn+n}) = t^\pi$ für alle $k \geqslant 1$. □

Mit Proposition 2.8 und der Beobachtung, dass (st^π, t^π) ein linked Pair von S ist, liefert dies direkt einen Beweis für Korollar 2.10.

Der Rest dieses Abschnitts beschäftigt sich mit einem ersten Erkennbarkeitsbegriff. Sei S eine endliche Halbgruppe und $\varphi\colon \Sigma^+ \to S$ ein Homomorphismus. Für eine Teilmenge P von $S \times S$ bezeichnet $L_\varphi(P)$ die endliche Vereinigung aller Sprachen $\varphi^{-1}(s)(\varphi^{-1}(t))^\omega$ mit $(s, t) \in P$. Der Homomorphismus φ *erkennt* $L \subseteq \Sigma^\omega$ *schwach*, falls eine Menge von linked Pairs $P \subseteq F(S)$ existiert, sodass $L = L_\varphi(P)$. Im weiteren Sinne wird auch häufig gesagt, eine Halbgruppe S erkennt die Sprache $L \subseteq \Sigma^\omega$ schwach, falls ein Homomorphismus $\varphi\colon \Sigma^+ \to S$ existiert, der L schwach erkennt.

Theorem 2.12. *Eine Sprache $L \subseteq \Sigma^\omega$ ist genau dann ω-regulär, wenn sie von einem Homomorphismus $\varphi\colon \Sigma^+ \to S$ schwach erkannt wird.*

Für beide Richtungen existieren Standardkonstruktionen, die hier kurz vorgestellt werden.

Beweis. Sei $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ ein Büchi-Automat mit $L(\mathcal{A}) = L$. Zugunsten einer einfacheren Notation wird $Q = \{1, \ldots, n\}$ angenommen. Für alle $i, j \in Q$ sei Q_{ij} die Menge aller Zustände, die auf Pfaden von i nach j auftauchen, d. h. ein Zustand $k \in Q$ ist genau dann in Q_{ij}, wenn in $\mathcal{A}$ ein Pfad der Form $i - \cdots \to k - \cdots \to j$ existiert. Insbesondere sind i und j selbst in Q_{ij} enthalten, sofern es einen Pfad von i nach j gibt. Wir betrachten den kommutativen und idempotenten Halbring $\{0, 1, 2\}$ mit der Addition $i + j \coloneqq \max(i, j)$ und der folgenden Multiplikation:

$$i \cdot j \coloneqq \begin{cases} 0 & \text{falls } i = 0 \text{ oder } j = 0 \\ 1 & \text{falls } i = 1 \text{ und } j = 1 \\ 2 & \text{sonst} \end{cases}$$

Diese beiden Verknüpfungen können auf natürliche Art und Weise zu einer Matrixmultiplikation erweitert werden. Nachfolgend sei S die durch diese Multiplikation entstehende Halbgruppe aller Matrizen der Größe $n \times n$. Darüber hinaus definieren wir einen Homomorphismus $\varphi\colon \Sigma^+ \to S$ durch $\varphi(u) = (r_{ij}(u))_{i,j \in Q}$, wobei

$$r_{ij}(u) = \begin{cases} 0 & \text{falls } Q_{ij} = \emptyset \\ 1 & \text{falls } Q_{ij} \neq \emptyset \text{ und } Q_{ij} \cap F = \emptyset \\ 2 & \text{falls } Q_{ij} \cap F \neq \emptyset \end{cases}$$

Ein Element aus S mit Urbild $u \in \Sigma^+$ beschreibt also für jedes Paar (i, j) von Zuständen, ob ein mit u beschrifteter Pfad über Endzustände, ein mit u beschrifteter Pfad ohne Endzustände oder kein mit u beschrifteter Pfad von i nach j existiert. Nun sei φ ohne Einschränkung surjektiv (ansonsten wird $\varphi(\Sigma^+)$ anstelle von S betrachtet). Ein linked Pair (R, E) mit $R = (r_{ij})_{i,j \in Q}$ und $E = (e_{ij})_{i,j \in Q}$ heißt *akzeptierend*, falls $r_{ij} \geqslant 1$ und $e_{jj} = 2$ für Zustände $i, j \in Q$. Es verbleibt zu zeigen, dass L die Vereinigung aller Sprachen der Form $\varphi^{-1}(R)(\varphi^{-1}(E))^\omega$ für akzeptierende Paare (R, E) ist. Die Inklusion $\varphi^{-1}(R)(\varphi^{-1}(E))^\omega \subseteq L$ ist mit der Definition von S und φ trivial. Umgekehrt betrachten wir für ein Wort $\alpha \in L$ eine Zerlegung $\alpha = uv_1v_2\cdots$, sodass ein akzeptierender Lauf von α auf $\mathcal{A}$ für jedes $i \geqslant 0$ nach dem Lesen von $uv_1v_2\cdots v_i$ denselben Endzustand besucht. Nach Korollar 2.9 kann diese Faktorisierung so gewählt werden, dass $\varphi(u) = R$ und $\varphi(v_i) = E$ für alle $i \geqslant 1$ und für ein akzeptierendes Paar (R, E).

Der Beweis der umgekehrten Richtung erfolgt durch Konstruktion eines Automaten $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ aus einem Homomorphismus $\varphi\colon \Sigma^+ \to S$ und einer Menge P von linked Pairs von S. Hierzu erweitern wir S um ein neues neutrales Element 1, auch wenn S bereits ein Monoid bildet. Anschließend wählen wir $Q = (S \cup \{1\}) \times E(S)$, $I = P$, $F = \{1\} \times E(S)$ und setzen δ aus allen Tupeln $((s, e), a, (t, e))$ mit $s, t \in S \cup \{1\}$, $e \in E(S)$ und $a \in \Sigma$ zusammen, für die $\varphi(a)t = s$ oder die Eigenschaften $s = 1$ und $\varphi(a)t = e$ erfüllt sind. Der Automat akzeptiert ein Wort $\alpha \in \Sigma^\omega$ genau dann, wenn ein $(s, e) \in P$ mit $\alpha \in \varphi^{-1}(s)(\varphi^{-1}(e))^\omega$ existiert. □

Für die Umwandlung eines Büchi-Automaten in einen erkennenden Homomorphismus kann auch eine einfachere Standardkonstruktion auf der Basis der so genannten Transitionshalbgruppe [CPP08, Kapitel 5.1] verwendet werden. Die hier vorgestellte Variante hat jedoch den Vorteil, dass der entstehende Homomorphismus die vom Automaten akzeptierte Sprache auch unter einem stärkeren Erkennbarkeitsbegriff noch erkennt.

2.3.4 Starke Erkennbarkeit

Das Konzept von schwacher Erkennbarkeit hat den Nachteil, dass, ähnlich wie bei Büchi-Automaten, keine einfachen Konstruktionen zur Erkennung des Komplements einer Sprache und zur Minimierung existieren. Daher hat sich vorwiegend ein anderer Erkennbarkeitsbegriff durchgesetzt. Sei $L \subseteq \Sigma^\omega$ eine Sprache und S eine endliche Halbgruppe. Ein Homomorphismus $\varphi\colon \Sigma^+ \to S$ *erkennt* L *stark*, falls für alle $(s,t) \in S \times S$ gilt:

$$\varphi^{-1}(s)(\varphi^{-1}(t))^\omega \subseteq L \text{ oder } \varphi^{-1}(s)(\varphi^{-1}(t))^\omega \cap L = \emptyset$$

Wie bei schwacher Erkennbarkeit lässt sich der Begriff auf Halbgruppen erweitern. Falls $\varphi\colon \Sigma^+ \to S$ eine Sprache L stark erkennt, wird L von φ bereits schwach erkannt, denn L kann in diesem Fall als endliche Vereinigung aller Sprachen der Form $\varphi^{-1}(s)(\varphi^{-1}(e))^\omega$ mit $\varphi^{-1}(s)(\varphi^{-1}(e))^\omega \subseteq L$ geschrieben werden. Die Umkehrung gilt im Allgemeinen nicht, jedoch kann, wie später in Theorem 2.15 gezeigt wird, zu jeder ω-regulären Sprache L eine Halbgruppe S und ein Homomorphismus $\varphi\colon \Sigma^+ \to S$ konstruiert werden, der L stark erkennt.

Einen alternativen Zugang zur starken Erkennbarkeit liefert folgende Definition: Zwei linked Pairs (s,e) und (t,f) einer Halbgruppe heißen *konjugiert*, falls $x, y \in S^1$ existieren, sodass $sx = t$, $xy = e$ und $yx = f$.

Proposition 2.13. *Konjugiertheit ist eine Äquivalenzrelation auf der Menge der linked Pairs von S. Sei $\varphi\colon \Sigma^+ \to S$ ein surjektiver Homomorphismus, dann sind zwei linked Pairs (s,e) und (t,f) genau dann zueinander konjugiert, wenn*

$$\varphi^{-1}(s)(\varphi^{-1}(e))^\omega \cap \varphi^{-1}(t)(\varphi^{-1}(f))^\omega \neq \emptyset.$$

Beweis. Seien (s,e) und (t,f) linked Pairs von S und $x, y \in S^1$ mit $sx = t$, $xy = e$ und $yx = f$. Dann ist $ty = sxy = se = s$, die Relation ist also symmetrisch. Zugleich ist Konjugiertheit reflexiv, denn mit $x = 1$ und $y = e$ erhält man $t = s$ und $f = e$. Für den Nachweis der Transitivität werden noch ein drittes linked Pair (u,g) und $x', y' \in S^1$ mit $tx' = u$, $x'y' = f$, $y'x' = g$ herangezogen. Dann ist $sxx' = tx' = u$, $(xx')(y'y) = xfy = xyxy = e^2 = e$ und $(y'y)(xx') = y'fx' = y'x'y'x' = g^2 = g$.

Für den Beweis des zweiten Teils der Aussage werden zunächst wieder zwei linked Pairs (s,e) und (t,f) und $x, y \in S^1$ mit $sx = t$, $xy = e$ und $yx = f$ betrachtet. Aufgrund der Surjektivität von φ existieren Wörter $u, v, w \in \Sigma^+$ mit $\varphi(u) = s$, $\varphi(v) = x$ und $\varphi(w) = y$. Aus den obigen Gleichungen ergibt sich dann $u(vw)^\omega = uv(wv)^\omega \in \varphi^{-1}(s)(\varphi^{-1}(e))^\omega \cap \varphi^{-1}(t)(\varphi^{-1}(f))^\omega$.

Umgekehrt kann davon ausgegangen werden, dass ein Wort $\alpha = uv_1v_1'v_2v_2'\cdots$ mit $\varphi(u) = s$, $\varphi(uv_1) = t$, sowie $\varphi(v_iv_i') = e$ und $\varphi(v_i'v_{i+1}) = f$ für alle $i \geqslant 1$ existiert. Die Existenz einer solchen Zerlegung folgt mit Korollar 2.10 aus der Voraussetzung. Da S endlich ist, existieren ferner Indizes $i, j \in \mathbb{N}$ mit $i < j$, sodass $\varphi(v_i) = \varphi(v_j)$. Setze nun $x \coloneqq \varphi(v_i) = \varphi(v_j)$ und $y \coloneqq \varphi(v_i'v_{i+1}\cdots v_{j-1}v_{j-1}')$. Dann ist $sx = se^{i-1}x = \varphi(uv_1v_1'\cdots v_{i-1}v_{i-1}'v_i) = tf^{i-1} = t$. Überdies gilt $xy = \varphi(v_iv_i'\cdots v_{j-1}v_{j-1}') = e^{j-i} = e$ und $yx = \varphi(v_i'v_{i+1}\cdots v_{j-1}'v_j) = f^{j-i} = f$. □

Sei (s, e) ein linked Pair, dann heißt die Äquivalenzklasse aller zu (s, e) konjugierten linked Pairs *Konjugationsklasse* von (s, e) und wird mit $[s, e]$ bezeichnet. Eine Menge P von linked Pairs heißt *unter Konjugation abgeschlossen*, falls für jedes linked Pair (s, e) aus P bereits $[s, e] \subseteq P$ gilt. Es ergibt sich nun folgende äquivalente Charakterisierung von starker Erkennbarkeit:

Proposition 2.14. *Sei $L \subseteq \Sigma^\omega$ eine Sprache, S eine endliche Halbgruppe und $\varphi\colon \Sigma^+ \to S$ ein surjektiver Homomorphismus. Dann wird L genau dann von φ stark erkannt, wenn eine unter Konjugation abgeschlossene Menge $P \subseteq F(S)$ existiert, sodass $L = L_\varphi(P)$.*

Beweis. Für den Fall, dass φ die Sprache L stark erkennt, sei P die Vereinigung aller linked Pairs (s, e) von S mit $\varphi^{-1}(s)(\varphi^{-1}(e))^\omega \subseteq L$. Falls ein linked Pair (t, f) zu einem der $(s, e) \in P$ konjugiert ist, folgt aus Proposition 2.13 die Existenz eines unendlichen Wortes $\alpha \in \Sigma^\omega$ im Durchschnitt von $\varphi^{-1}(s)(\varphi^{-1}(e))^\omega \subseteq L$ und $\varphi^{-1}(t)(\varphi^{-1}(f))^\omega$. Da φ stark erkennend ist, gilt dann jedoch bereits $\varphi^{-1}(t)(\varphi^{-1}(f))^\omega \subseteq L$ und es ist $(t, f) \in P$.

Sei nun umgekehrt P eine unter Konjugation abgeschlossene Menge, sodass L die endliche Vereinigung aller Sprachen $\varphi^{-1}(s)(\varphi^{-1}(e))^\omega$ mit $(s, e) \in P$ ist. Gilt $\varphi^{-1}(s)(\varphi^{-1}(e))^\omega \setminus L \neq \emptyset$ für ein linked Pair (s, e), dann ist kein zu (s, e) konjugiertes linked Pair in P enthalten. Nach Proposition 2.13 folgt $\varphi^{-1}(s)(\varphi^{-1}(e))^\omega \cap L = \emptyset$. □

Bei starker Erkennbarkeit ist es möglich, effizient aus einer beliebigen Menge $P \subseteq S \times S$ eine unter Konjugation abgeschlossene Menge $\hat{Q} \subseteq F(S)$ mit $L_\varphi(\hat{Q}) = L_\varphi(P)$ zu berechnen. Dies erfolgt in zwei Schritten: Zunächst wird $Q = \{\, (st^\pi, t^\pi) \mid (s, t) \in P \,\}$ berechnet und anschließend der Abschluss von Q unter Konjugation gebildet. Nach Lemma 2.11 ist $L_\varphi(P) \subseteq L_\varphi(Q)$ und aufgrund der starken Erkennbarkeit gilt auch die umgekehrte Inklusion $L_\varphi(Q) \subseteq L_\varphi(P)$. Die Gleichheit $L_\varphi(\hat{Q}) = L_\varphi(Q)$ folgt aus der Argumentation im Beweis von Proposition 2.14.

Das folgende Resultat stellt eine Verbindung zwischen starker Erkennbarkeit, schwacher Erkennbarkeit und Büchi-Automaten her:

Theorem 2.15. *Eine Sprache $L \subseteq \Sigma^\omega$ ist genau dann ω-regulär, wenn sie von einem Homomorphismus $\varphi\colon \Sigma^+ \to S$ stark erkannt wird.*

Beweis. Wir zeigen Äquivalenz von schwacher und starker Erkennbarkeit. Dass starke Erkennbarkeit schwache Erkennbarkeit impliziert, wurde zu Beginn dieses Abschnitts bereits angemerkt. Für die umgekehrte Richtung wird gezeigt, dass die Konstruktion der Halbgruppe und des Homomorphismus im Beweis von Theorem 2.12 bereits starke Erkennbarkeit liefert. Man betrachte hierzu ein akzeptierendes linked Pair (R, E) und ein dazu konjugiertes linked Pair (T, F), d. h. $RX = T$, $E = XY$ und $F = YX$ für Matrizen $X, Y \in S^1$. Wie im vorigen Beweis werden Elemente einer Matrix durch die entsprechenden, mit Doppelindizes versehenen, Kleinbuchstaben dargestellt.

Da (R, E) akzeptierend ist, existieren $i, j \in Q$ mit $r_{ij} \geqslant 1$ und $e_{jj} = 2$. Aus der Faktorisierung $E = XY$ folgt, dass dann für ein $k \in Q$ auch $x_{jk} \geqslant 1$ und $y_{kj} \geqslant 1$ gilt und dass mindestens einer der beiden Werte den Wert 2 annimmt. Einsetzen in die Gleichung $F = YX$ ergibt $f_{kk} = 2$. Der Gleichung $RX = T$ kann zudem entnommen werden, dass t_{ik} mindestens 1 ist.

Also ist auch (T, F) akzeptierend. Die Menge der akzeptierenden linked Pairs ist demgemäß unter Konjugation abgeschlossen und mit Proposition 2.14 kann gefolgert werden, dass L stark erkannt wird. □

2.3.5 Die syntaktische Halbgruppe

Das Konzept der syntaktischen Kongruenz wurde erstmals von Arnold [Arn85] auf Sprachen über unendlichen Wörtern übertragen. Sei $L \subseteq \Sigma^\omega$ eine ω-Sprache, dann ist die *syntaktische Kongruenz* von L auf Σ^+ definiert durch $u \equiv_L v$ genau dann, wenn

$$xuyz^\omega \in L \Leftrightarrow xvyz^\omega \in L \text{ und } z(xuy)^\omega \in L \Leftrightarrow z(xvy)^\omega \in L$$

für alle $x, y, z \in \Sigma^*$. Um zu sehen, dass es sich hierbei tatsächlich um eine Kongruenzrelation handelt, betrachtet man zwei Paare $u, v \in \Sigma^+$ und $u', v' \in \Sigma^+$ mit $u \equiv_L v$ und $u' \equiv_L v'$. Seien $x, y, z \in \Sigma^*$ beliebig. Nun folgt aus $xuu'yz^\omega \in L$ wegen $u \equiv_L v$ bereits $xvu'yz^\omega \in L$ und wegen $u' \equiv_L v'$ zudem $xvv'yz^\omega \in L$. Analog folgt aus $z(xuu'y)^\omega \in L$ zunächst $z(xvu'y) \in L$ und in einem zweiten Schritt $z(xvv'y)^\omega \in L$. Die jeweils umgekehrten Implikationen gelten aus Symmetriegründen ebenfalls. Der Quotient $\Sigma^+/\equiv_L$ heißt *syntaktische Halbgruppe* von L und der Homomorphismus $\eta_L \colon \Sigma^+ \to \Sigma^+/\equiv_L$ mit $\eta_L(u) = \{\, v \in \Sigma^+ \mid v \equiv_L u \,\}$ für alle $u \in \Sigma^+$ heißt *syntaktischer Homomorphismus* von L. Syntaktische Homomorphismen erlauben eine weitere Charakterisierung von starker Erkennbarkeit:

Theorem 2.16. *Sei $L \subseteq \Sigma^\omega$ eine Sprache, S eine endliche Halbgruppe und $\varphi \colon \Sigma^+ \to S$ ein surjektiver Homomorphismus. Dann erkennt φ die Sprache L genau dann stark, wenn η_L die Sprache L stark erkennt und ein surjektiver Homomorphismus $\psi \colon S \to \Sigma^+/\equiv_L$ mit $\psi \circ \varphi = \eta_L$ existiert.*

Beweis. Wir betrachten zunächst den Fall, dass $\varphi \colon \Sigma^+ \to S$ die Sprache L stark erkennt, und definieren einen Homomorphismus $\psi \colon S \to \Sigma^+/\equiv_L$ durch $\psi(s) = \eta_L(\varphi^{-1}(s))$ für alle $s \in S$. Für den Nachweis der Wohldefiniertheit dieses Homomorphismus reicht die Beobachtung, dass zwei Wörter $u, v \in \Sigma^+$ mit $\varphi(u) = \varphi(v)$ äquivalent bezüglich der syntaktischen Kongruenz sind. Noch zu zeigen ist, dass η_L die Sprache L stark erkennt. Sei hierzu (s, t) ein beliebiges Paar von Elementen aus $\Sigma^+/\equiv_L$ und seien $\alpha, \beta \in \eta_L^{-1}(s)(\eta_L^{-1}(t))^\omega$. Zunächst wissen wir nach Lemma 2.11 und durch zweifache Anwendung des Argumentes aus Korollar 2.9, dass Faktorisierungen $\alpha = uv_1v_2\cdots$ und $\beta = u'v_1'v_2'\cdots$ existieren, sodass $\eta_L(u) = \eta_L(u') = st^\pi$, $\eta_L(v_i) = \eta_L(v_i') = t^\pi$, $\varphi(v_i) = \varphi(v_{i+1})$ und $\varphi(v_i') = \varphi(v_{i+1}')$ für alle $i \geqslant 1$. Da φ die Sprache L stark erkennt, ist α (bzw. β) genau dann in L enthalten, wenn $uv_1^\omega \in L$ (bzw. $u'(v_1')^\omega \in L$) ist. Ferner gilt wegen $u \equiv_L u'$ und $v_1 \equiv_L v_1'$ die Äquivalenz $uv_1^\omega \in L \Leftrightarrow u'v_1^\omega \in L \Leftrightarrow u'(v_1')^\omega \in L$. Insgesamt ist also α genau dann in L, wenn β in L ist. Da diese Argumentation für beliebige $\alpha, \beta \in \eta_L^{-1}(s)(\eta_L^{-1}(t))^\omega$ gilt, tritt also einer der beiden Fälle $\eta_L^{-1}(s)(\eta_L^{-1}(t))^\omega \subseteq L$ oder $\eta_L^{-1}(s)(\eta_L^{-1}(t))^\omega = \emptyset$ ein.

Für die umgekehrte Richtung erkennt η_L nach Voraussetzung die Sprache L stark und die Verkettung $\psi = \eta_L \circ \varphi^{-1}$ ist ein wohldefinierter Homomorphismus. Seien nun $\alpha, \beta \in \varphi^{-1}(s)(\varphi^{-1}(e))^\omega$ für ein linked Pair (s, e) von S, so ist $\alpha, \beta \in \eta_L^{-1}(\psi(s))(\eta_L^{-1}(\psi(e)))^\omega$ und da L von η_L stark erkannt wird, ist α genau dann in L, wenn β in L ist. □

Die syntaktische Halbgruppe ist also die (bis auf Isomorphie) eindeutige kleinste Halbgruppe, die eine Sprache L stark erkennt. Das folgende Beispiel deutet an, wieso ein verwandtes Konzept für schwache Erkennbarkeit nicht existiert:

Beispiel 2.17. *Sei $S = \{a, b\}$ die Halbgruppe mit der Multiplikation $aa = ab = a$ und $ba = bb = b$ und $T = \{a, b\}$ die Halbgruppe mit der Multiplikation $aa = ba = a$ und $ab = bb = b$. Es lässt sich leicht nachprüfen, dass sowohl S als auch T die Sprache $(\Sigma^* a)^\omega$ schwach erkennen.*

2.4 Algorithmen auf Halbgruppen und Homomorphismen

Um Algorithmen auf Halbgruppen und Halbgruppen-Homomorphismen zu entwerfen und sinnvoll analysieren zu können, muss zunächst festgelegt werden, in welcher Form die Eingabe vorliegt. In den folgenden Abschnitten wird dabei von diesen Konventionen ausgegangen:

- Die Halbgruppe S ist in eine größere Halbgruppe T eingebettet, in der Multiplikationen in konstanter Zeit ausgeführt werden können. In der Praxis handelt es sich hierbei häufig um Einbettungen in die sogenannte *Transitionshalbgruppe* eines Automaten, in eine Halbgruppe der im Beweis von Theorem 2.12 konstruierten Form oder die Halbgruppe S ist explizit durch eine Verknüpfungstabelle gegeben. Die Menge der Elemente von S ist jedoch im Allgemeinen nicht explizit verfügbar.
- Zwei Elemente $s, t \in S$ können auf Gleichheit geprüft werden. Für die Implementierung von Algorithmen bietet es sich zudem an, die Elemente linear zu ordnen oder in einer Hashtabelle zu verwalten, damit Operationen auf Teilmengen von Elementen effizient realisierbar sind.
- Der Homomorphismus $\varphi\colon \Sigma^+ \to S$ ist implizit durch eine Abbildung $f\colon \Sigma \to S$ mit $f(a) = \varphi(a)$ für alle $a \in \Sigma$ gegeben.
- Einige der Algorithmen erwarten außer einer Halbgruppe und einem Homomorphismus zudem eine Menge $P \subseteq S \times S$ als Teil der Eingabe. Es wird bei allen Eingaben dieser Form davon ausgegangen, dass jedes Paar $(s, t) \in S \times S$ in konstanter Zeit auf Enthaltensein in P getestet werden kann.

Um alle Elemente von S aufzuzählen, kann der Cayley-Graph von S in Zeit $\mathcal{O}(|\Sigma| \cdot |S|)$ berechnet und dann dessen Knotenmenge ausgegeben werden. Entsprechend ist die Berechnung der vollständigen Multiplikationstabelle mit einem Zeitaufwand in $\mathcal{O}(|\Sigma| \cdot |S| + |S|^2)$ möglich.

Einige der folgenden Algorithmen führen lediglich Multiplikationen der Form $\varphi(a)s$ und der Form $s\varphi(a)$ für $s \in S$ und $a \in \Sigma$ aus. Für diese Algorithmen bietet sich die Kodierung von Halbgruppe und Homomorphismus als (zweiseitiger) Cayley-Graph an. Diese Darstellung ist mit einem Platzbedarf in $\mathcal{O}(|\Sigma| \cdot |S|)$ sehr kompakt und erfüllt bei der Einschränkung auf Multiplikationen der oben genannten Form alle Anforderungen an die Zeitkomplexität der zur Verfügung gestellten Operationen.

3 Linked Pairs und Konjugationsklassen

In diesem Kapitel werden zunächst einige strukturelle Eigenschaften von linked Pairs und Konjugationsklassen von linked Pairs beschrieben. Anschließend werden Konjugationsklassen spezieller Klassen von Halbgruppen (sogenannter *Pseudovarietäten*) untersucht. Diese Analysen erlauben es unter anderem, schärfere Schranken für die Laufzeiten von Algorithmen auf Instanzen aus der jeweiligen Familie von Halbgruppen anzugeben. Den Abschluss bilden Algorithmen für die effiziente Berechnung der linked Pairs einer Halbgruppe und für die Partitionierung von linked Pairs in Konjugationsklassen.

3.1 Eigenschaften und Komplexität

Als Einstieg werden Eigenschaften von linked Pairs untersucht, die die spätere Analyse erleichtern und ein wichtiger Baustein für nachfolgende Algorithmen sind. Die beiden wichtigsten Hilfssätze sind Lemma 3.2 und Lemma 3.4, die Charakterisierungen von linked Pairs und Konjugiertheit durch Greens Relationen liefern, sofern man sich auf Paare beschränkt, bei denen beide Komponenten in derselben $\mathcal{J}$-Klasse liegen.

Lemma 3.1. *Seien $s, e \in S$ mit $e^2 = e$. Dann ist $se = s$ genau dann, wenn $s \leqslant_{\mathcal{L}} e$.*

Beweis. Sei $p \in S^1$ mit $s = pe$. Dann ist $se = pee = pe = s$. Die andere Richtung folgt direkt aus der Definition der Relation $\leqslant_{\mathcal{L}}$. □

Lemma 3.2. *Seien $s \in S$ und $e \in E(S)$ mit $e \leqslant_{\mathcal{J}} s$. Dann ist (s, e) genau dann ein linked Pair, wenn $s \mathcal{L} e$ gilt.*

Beweis. Nach Lemma 2.4 folgt aus $e \leqslant_{\mathcal{J}} se$ direkt $e \mathcal{L} se$. Falls (s, e) ein linked Pair ist, gilt $se = s$ und daher auch $s \mathcal{L} e$. Die umgekehrte Richtung folgt aus Lemma 3.1. □

Lemma 3.3. *Seien (s, e) und (t, f) zueinander konjugierte linked Pairs einer Halbgruppe S. Dann ist $s \mathcal{R} t$ und $e \mathcal{J} f$.*

Beweis. Seien $x, y \in S^1$ fest mit $sx = t$, $xy = e$ und $yx = f$. Aus $sx = t$ folgt direkt $t \leqslant_{\mathcal{R}} s$ und mit $ty = sxy = se = s$ folgt $s \leqslant_{\mathcal{R}} t$. Zudem ist $e = ee = xyxy = xfy$ und $f = ff = yxyx = yex$, also $e \mathcal{J} f$. □

Lemma 3.4. *Seien $(s, e), (t, f)$ linked Pairs einer endlichen Halbgruppe S mit $e \leqslant_{\mathcal{J}} s$ und $f \leqslant_{\mathcal{J}} t$. Dann sind (s, e) und (t, f) genau dann zueinander konjugiert, wenn $s \mathcal{R} t$.*

Beweis. Die Richtung von links nach rechts ist eine direkte Konsequenz aus Lemma 3.3. Bei der Umkehrung ist laut Voraussetzung $s \mathcal{R} t$ und folglich existieren $q, q' \in S^1$ mit $sq = t$ und $tq' = s$. Setze nun $x \coloneqq eq$ und $y \coloneqq fq'$, dann ist $sx = seq = sq = t$. Ferner gilt nach Lemma 3.2 bereits $s \mathcal{L} e$. Es existiert also ein $p \in S^1$ mit $ps = e$ und damit gilt $xy = eqfq' = psqfq' = ptfq' = ptq' = ps = e$. Analog kann gezeigt werden, dass $yx = f$. □

Im Fall von endlichen Wörtern ist die triviale Halbgruppe die einzige, die für beliebige Alphabete Σ ausschließlich die trivialen Sprachen $\emptyset$ und Σ^* erkennt. Bei unendlichen Wörtern und starker Erkennbarkeit verhält sich dies etwas anders: Beispielsweise gibt es keine nicht-triviale Sprache, die von einer Gruppe stark erkannt wird. Folgende Proposition charakterisiert die Klasse von Halbgruppen, welche für beliebige Alphabete Σ ausschließlich die Mengen $\emptyset$ und Σ^ω stark erkennen:

Proposition 3.5. *Sei S eine endliche Halbgruppe. Dann sind folgende Eigenschaften äquivalent:*

1. *Es gibt genau eine Konjugationsklasse von linked Pairs von S.*
2. *Für alle $s, e \in S$ mit $e^2 = e$ gilt $e \mathcal{R} se$.*
3. *Für alle $s, e \in S$ mit $e^2 = e$ gilt $e \leqslant_{\mathcal{R}} se$.*
4. *Für alle $e \in E(S)$ bildet die Unterhalbgruppe $Se = \{ se \mid s \in S \}$ eine Gruppe.*

Beweis. (1) ⇒ (2): Sei $e \in S$ idempotent und s ein beliebiges Element aus S. Laut Voraussetzung sind (e, e) und (se, e) zueinander konjugierte linked Pairs von S und mit Lemma 3.3 folgt $se \mathcal{R} e$.

(2) ⇒ (3): trivial.

(3) ⇒ (4): Sei $e \in S$ idempotent. Für jedes $s \in S$ existiert wegen $e \leqslant_{\mathcal{R}} se$ ein $q \in S^1$ mit $seq = e$ und folglich auch $(se)(qe) = e$. Mit Proposition 2.7 kann nun gefolgert werden, dass Se eine Gruppe bildet.

(4) ⇒ (1): Seien (s, e) und (t, f) linked Pairs von S. Da Se und Sf laut Voraussetzung Gruppen sind, ist $s = se \mathcal{H} e$ und $t = tf \mathcal{H} f$. Wegen $ef \mathcal{H} f$ und $fe \mathcal{H} e$ gilt außerdem $e \mathcal{R} f$ und damit auch $s \mathcal{R} t$. Mit Lemma 3.4 folgt nun direkt, dass (s, e) und (t, f) zueinander konjugiert sind. □

Als Spezialfall ergibt sich unmittelbar das folgende Resultat für erkennende Gruppen:

Korollar 3.6. *Sei G eine endliche Gruppe. Dann gibt es $|G|$ linked Pairs von G, die paarweise zueinander konjugiert sind.*

Beweis. Sei G eine endliche Gruppe mit neutralem Element 1. Für jedes idempotente Element e existiert ein Rechtsinverses $\overline{e}$ und es ist $e = ee\overline{e} = e\overline{e} = 1$. Daher gilt $E(G) = \{ 1 \}$ und $F(G) = G \times \{ 1 \}$. Mit der Charakterisierung aus Proposition 3.5 und $G \cdot 1 = G$ folgt die Konjugiertheit aller Paare von linked Pairs. □

Für linked Pairs existiert ein zu Proposition 3.5 ähnliches Resultat, das hier der Vollständigkeit halber erwähnt sei:

Proposition 3.7. *Sei S eine endliche Halbgruppe. Dann sind folgende Eigenschaften äquivalent:*

1. *Es gibt genau ein linked Pair von S.*
2. *Es existiert genau ein idempotentes Element e in S und dieses erfüllt für alle $s \in S$ die Gleichung $se = e$.*
3. *Für alle $e \in E(S)$ und alle $s \in S$ gilt $es = e = se$.*

Beweis. (1) $\Rightarrow$ (2): Für alle $s \in S$ und alle $e \in E(S)$ ist (se, e) ein linked Pair von S. Insbesondere bilden für alle $e, f \in E(S)$ die Paare (e, e) und (f, f) linked Pairs.

(2) $\Rightarrow$ (3): Sei e das eindeutige idempotente Element von S und sei $s \in S$ beliebig. Nun ist $s^\pi = e$ und folglich $es = s^\pi s = ss^\pi = se = e$.

(3) $\Rightarrow$ (1): Seien (s, e) und (t, f) linked Pairs von S. Nach Bedingung (3) gilt $s = se = e$ und $t = tf = f$, sowie $e = ef = f$. □

Eine weitere interessante Klasse von Halbgruppen ist die Familie der Halbgruppen mit trivialen Konjugationsklassen. Sie stellt in gewisser Weise ein Gegenstück zu der in Proposition 3.5 beschriebenen Klasse dar. Auf Sprachebene haben diese Halbgruppen die Eigenschaft, dass alle Sprachen $\varphi^{-1}(s)(\varphi^{-1}(e))^\omega$ für linked Pairs (s, e) paarweise disjunkt sind.

Proposition 3.8. *Eine endliche Halbgruppe ist genau dann $\mathcal{J}$-trivial, wenn alle Konjugationsklassen von linked Pairs trivial sind.*

Beweis. Sei S eine endliche $\mathcal{J}$-triviale Halbgruppe. Angenommen, zwei linked Pairs (s, e) und (t, f) seien zueinander konjugiert, dann werden nach Lemma 3.3 die beiden Bedingungen $s \mathcal{R} t$ und $e \mathcal{J} f$ erfüllt. Da alle $\mathcal{J}$-Klassen trivial sind, folgt $s = t$ und $e = f$.

Betrachte nun eine endliche Halbgruppe S, die nur aus trivialen Konjugationsklassen besteht. Wir zeigen in zwei Schritten, dass die Halbgruppe $\mathcal{R}$-trivial und $\mathcal{L}$-trivial ist. Dies liefert, unter Berücksichtigung von Lemma 2.5, ein hinreichendes Kriterium für die $\mathcal{J}$-Trivialität. Seien im Folgenden $s, t \in S$ mit $s \neq t$.

Angenommen, es gelte $s \mathcal{R} t$, dann existieren $q, q' \in S$ mit $sq = t$ und $tq' = s$. Offensichtlich ist $s(qq')^n = s$ und $t(q'q)^n = t$ für alle $n \in \mathbb{N}$ und es ist leicht zu überprüfen, dass $(s, (qq')^\pi)$ und $(t, (q'q)^\pi)$ zueinander konjugierte linked Pairs von S sind. Da $s \neq t$ ist, handelt es sich um zwei verschiedene Paare, ein Widerspruch zur Voraussetzung.

Nun betrachten wir den analogen Fall $s \mathcal{L} t$, d. h. es existieren $p, p' \in S$ mit $ps = t$ und $p't = s$. Wir setzen $e = (pp')^\pi$ und $f = (p'p)^\pi$. Wieder kann einfach überprüft werden, dass (e, e) und (pf, f) zueinander konjugierte linked Pairs sind. Mit der Voraussetzung folgt dann $pf = e = f$. Schließlich ist $(p'p)^n s = s$ für alle $n \in \mathbb{N}$ und damit $s = fs = pfs = ps = t$, ein Widerspruch zur Voraussetzung $s \neq t$. □

Nachdem die Extremfälle untersucht wurden, folgen noch einige Abschätzungen für einfache, häufig auftauchende Klassen von Halbgruppen. Die Klasse **D** (bzw. **K**) besteht aus allen endlichen Halbgruppen, in denen die Gleichung $se = e$ (bzw. $es = e$) für alle idempotenten Elemente $e \in S$ und für alle $s \in S$ erfüllt ist.

Proposition 3.9. *Sei $S \in \mathbf{D}$, dann gibt es $|E(S)|$ linked Pairs von S, die alle paarweise zueinander konjugiert sind.*

Beweis. Sei (s, e) ein linked Pair von S, dann ist $s = se = e$. Demnach ist $F(S)$ die Menge aller Paare (e, e) mit $e \in E(S)$. Zudem folgt für alle $s \in S$ und $e \in E(S)$ aus der Gleichheit $se = e$ insbesondere $se \mathcal{R} e$ und nach Proposition 3.5 gibt es genau eine Konjugationsklasse. □

Proposition 3.10. *Sei $S \in \mathbf{K}$, dann gibt es $|E(S)|^2$ linked Pairs und $|E(S)|$ Konjugationsklassen von linked Pairs von S.*

Beweis. Sei (s, e) ein linked Pair von S, dann ist $s^2 = ses = se = s$. Ferner bildet für alle $e, f \in E(S)$ das Paar (e, f) ein linked Pair, da $ef = e$. Also ist $F(S) = E(S) \times E(S)$.

Für zwei idempotente Elemente e und f gilt $e \mathcal{L} f$, denn $ef = e$ und $fe = f$. Zudem ist $e \mathcal{R} f$ genau dann, wenn $e = f$, denn falls ein $q \in S^1$ mit $eq = f$ existiert, folgt $e = eq = f$. Insgesamt ergibt dies mit Lemma 3.4, dass zwei linked Pairs genau dann zueinander konjugiert sind, wenn die ersten Komponenten übereinstimmen. □

Die Klasse **CS** setzt sich aus den endlichen Halbgruppen S zusammen, die für alle $s \in S$ die Gleichung $SsS = S$ erfüllen. An dieser Stelle wird folgende strukturelle Eigenschaft dieser Klasse von Halbgruppen vorausgesetzt, die eine direkte Konsequenz des Rees-Sushkevich-Theorems [Pin86, Proposition 3.3.2] ist:

Lemma 3.11. *Sei $S \in \mathbf{CS}$ und r die Anzahl der $\mathcal{R}$-Klassen von S. Dann enthält jede $\mathcal{L}$-Klasse von S genau r idempotente Elemente.*

Diese Eigenschaft ermöglicht es, für Halbgruppen aus **CS** die Anzahl an linked Pairs und Konjugationsklassen genau anzugeben:

Proposition 3.12. *Sei $S \in \mathbf{CS}$ und r die Anzahl der $\mathcal{R}$-Klassen von S. Dann gibt es $r \cdot |S|$ linked Pairs und r Konjugationsklassen von linked Pairs von S.*

Beweis. Aus der Gleichung $SsS = S$ folgt direkt, dass alle Elemente von S in derselben $\mathcal{J}$-Klasse liegen. Für jedes Element $s \in S$ gibt es nach Lemma 3.2 und Lemma 3.11 also genau r Elemente e, sodass (s, e) ein linked Pair bildet. Zudem legt die $\mathcal{R}$-Klasse der ersten Komponente eines linked Pairs die Konjugationsklasse eindeutig fest, vgl. Lemma 3.4. □

Abschließend werden noch allgemeine obere Schranken für die Anzahl an linked Pairs und Konjugationsklassen angegeben. Für die Anzahl der linked Pairs liegt nahe, dass die naive Schranke $|S|^2$ scharf ist, und tatsächlich lässt sich ein entsprechendes Beispiel leicht konstruieren: Die Menge $S = \{ e_1, e_2, \ldots, e_n \}$ bildet zusammen mit der Verknüpfung $e_i \cdot e_j = e_i$ eine Halbgruppe aus $\mathbf{K}$. Es ist $E(S) = S$ und nach Proposition 3.10 gibt es genau n^2 linked Pairs von S.

Etwas anders verhält es sich mit der Anzahl der Konjugationsklassen: Dort liegt die Anzahl der maximal vorhandenen Klassen bei etwa der Hälfte der naiven oberen Schranke.

Proposition 3.13. *Sei S eine endliche Halbgruppe. Dann gibt es maximal $(|S|^2 + |S|)/2$ Konjugationsklassen von linked Pairs von S.*

Beweis. Wir verwenden ein einfaches Zählargument. Markiere zunächst die Konjugationsklassen aller linked Pairs der Form (e, e) mit $e \in E(S)$. Für alle Paare $(s, t) \in S$ mit $s \neq t$ tritt nun einer von zwei Fällen ein. Die erste Möglichkeit ist, dass mindestens eines der Paare (s, t) oder (t, s) kein linked Pair von S ist. Anderenfalls gilt $s \mathcal{L} t$ und nach Lemma 3.4 liegen (s, t) und (s, s) in derselben Konjugationsklasse. In beiden Fällen wird nur durch höchstens eines der beiden Paare (s, t) oder (t, s) eine neue Konjugationsklasse markiert. Die Anzahl an Klassen insgesamt ist also durch $|S| + |S|(|S| - 1)/2 = (|S|^2 + |S|)/2$ nach oben beschränkt. □

Um zu zeigen, dass die angegebene Schranke scharf ist, eignen sich $\mathcal{J}$-triviale Halbgruppen besonders, denn nach Proposition 3.8 reicht es, eine solche Halbgruppe mit einer ausreichenden Anzahl von linked Pairs anzugeben. Ein derartiges Beispiel erhält man durch Erweiterung der Menge $S = \{ 1, 2, \ldots, n \}$ um die Multiplikation $i \cdot j = \max\{ i, j \}$. Für $i, j \in S$ bildet (i, j) genau dann ein linked Pair, wenn $i \geqslant j$ gilt. Demzufolge gibt es $\sum_{k=1}^{n} k = (n^2 + n)/2$ linked Pairs von S.

3.2 Effiziente Berechnung von linked Pairs

Der naive Algorithmus zur Berechnung aller linked Pairs einer endlichen Halbgruppe S ermittelt zunächst in $\mathcal{O}(|\Sigma| \cdot |S|)$ Zeit alle Elemente von S und testet dann für jedes Tupel (s, t) aus $S \times S$, ob die Gleichungen $st = s$ und $t^2 = t$ erfüllt sind. Der Zeitaufwand für den zweiten Schritt ist offensichtlich quadratisch in der Größe von S.

Ein effizienteres Verfahren basiert auf der Eigenschaft aus Lemma 3.1. Zuerst wird der linke Cayley-Graph von S in Zeit $\mathcal{O}(|\Sigma| \cdot |S|)$ berechnet. Anschließend wird für jeden Knoten t aus der Knotenmenge S getestet, ob $t^2 = t$ gilt. Ist $t^2 \neq t$, so wird der Knoten verworfen. Anderenfalls wird ausgehend von t eine Tiefensuche im linken Cayley-Graph gestartet. Für jeden erreichbaren Knoten s ist $s \leqslant_{\mathcal{L}} t$ und daher bildet (s, t) ein linked Pair. Die gesamte Laufzeit hierfür beträgt $\mathcal{O}(|\Sigma| \cdot (|S| + |F(S)|))$.

3.3 Effiziente Berechnung der Konjugationsklassen

Als *Partition* $\mathcal{P}$ einer Menge M bezeichnet man die Menge $M/\equiv$ aller Äquivalenzklassen bezüglich einer Äquivalenzrelation $\equiv$ auf M. Es handelt sich also um eine disjunkte Zerlegung $\mathcal{P} = \{ C_1, \ldots, C_k \}$ in nicht-leere Teilmengen $C_1, \ldots, C_k$ von M. Diese Teilmengen werden *Klassen* von $\mathcal{P}$ genannt. Eine wichtige Beobachtung ist, dass eine Partition die vollständige Information über die zugrundeliegende Äquivalenzrelation enthält, denn für alle $x, y \in M$ gilt genau dann $x \equiv y$, wenn x und y in derselben Klasse von $\mathcal{P}$ liegen. Bei Festlegung einer gemeinsamen Grundmenge M existiert also eine Bijektion zwischen der Menge aller Partitionen und der Menge aller Äquivalenzrelationen. Zugunsten einer etwas einfacheren Notation wird im Folgenden der Bezeichner einer Partition in einigen Zusammenhängen als Synonym für die entsprechende Äquivalenzrelation verwendet, sofern dadurch keine Verwechslungen auftreten. Beispielsweise ist eine Partition $\mathcal{P}$ feiner als eine Äquivalenzrelation $\sim$, wenn die (eindeutige) der Partition zugrundeliegende Äquivalenzrelation $\equiv$ feiner als $\sim$ ist.

Eine effiziente Datenstruktur, um "dynamische" Partitionen zu verwalten, ist die sogenannte *Union-Find-Struktur*, in der englischsprachigen Literatur oft auch als *disjoint-set data structure* bezeichnet. Diese Datenstruktur stellt zwei elementare Operationen zur Verfügung: Die Operation Find(x) ermittelt die Klasse, in der sich ein Element M momentan befindet, und die Operation Union(C, D) verschmilzt zwei Klassen C und D der Partition zu einer neuen Klasse, sie ersetzt die momentane Partition $\mathcal{P}$ also durch eine neue Partition $(\mathcal{P} \setminus \{ C, D \}) \cup \{ C \cup D \}$. Im Folgenden verwenden wir für die oben beschriebene, klassische Union-Operation den Begriff *atomare* Union-*Operation* und erweitern die Routine, sodass mehrere Klassen gleichzeitig vereinigt werden können. Eine mögliche, von der Implementierung der atomaren Union-Operation unabhängige, Realisierung dieser Erweiterung ist in Algorithmus 3.1 dargestellt. Wir nehmen zudem im Folgenden ohne Einschränkung an, dass Klassen durch Repräsentanten identifiziert werden und dass Find(x) einen eindeutigen Vertreter je Klasse zurückgibt. Insbesondere ist Find(x) $= x$, falls x in einer einelementigen Klasse liegt.

Algorithmus 3.1 Erweiterung der Union-Operation auf mehrere Klassen

```
1: procedure Union(R)
2:     wähle ein r ∈ R
3:     R ← R \ { r }
4:     for all r' ∈ R do Union(r, r') end for
5: end procedure
```

Die folgende Definition erweitert den Begriff der Linksverträglichkeit auf linked Pairs: Eine Äquivalenzrelation $\equiv$ über $F(S)$ heißt *linksverträglich*, falls für alle $p \in S$ und für alle $(s, e), (t, f) \in F(S)$ mit $(s, e) \equiv (t, f)$ gilt, dass $(ps, e) \equiv (pt, f)$. Wir betrachten in diesem Abschnitt zwei Äquivalenzrelationen $\sim$ und $\approx$ über $F(S)$, die wie folgt definiert sind:

$$(s, e) \sim (t, f) \text{ falls } (s, e) \text{ und } (t, f) \text{ zueinander konjugiert sind}$$
$$(s, e) \approx (t, f) \text{ falls } e \mathrel{\mathcal{L}} s \mathrel{\mathcal{R}} t \mathrel{\mathcal{L}} f \text{ oder } (s, e) = (t, f)$$

Die Beziehung der beiden Relationen ist folgende:

Lemma 3.14. *Die Relation $\sim$ ist die feinste linksverträgliche Äquivalenzrelation, die gröber als $\approx$ ist.*

Beweis. Nach Lemma 3.4 verfeinert $\approx$ die Relation $\sim$. Wir betrachten nun ein beliebiges Element p einer endlichen Halbgruppe S. Falls (s, e) ein linked Pair von S ist, dann ist auch (ps, e) ein linked Pair. Seien (s, e) und (t, f) zueinander konjugierte linked Pairs von S und seien $x, y \in S^1$ mit $sx = t$, $xy = e$ und $yx = f$. Dann ist $(ps)x = pt$ und es folgt $(ps, e) \sim (pt, f)$. Dies zeigt die Linksverträglichkeit.

Für den Nachweis, dass $\sim$ die feinste Relation mit diesen Eigenschaften ist, betrachten wir eine beliebige linksverträgliche Relation $\simeq$ über $F(S)$, die gröber als $\approx$ ist. Wir zeigen nun, dass aus $(s, e) \sim (t, f)$ bereits $(s, e) \simeq (t, f)$ folgt. Seien also (s, e) und (t, f) zueinander konjugierte linked Pairs. Dann existieren $x, y \in S^1$, sodass $sx = t$, $xy = e$ und $yx = f$. Nun ist $ex = xyx = xf$ und $xfy = xyxy = ee = e$. Dies zeigt $e \mathrel{\mathcal{R}} xf$. Außerdem gilt $xf \mathrel{\mathcal{L}} f$, denn es ist $yxf = ff = f$. Per Definition gilt also $(e, e) \approx (xf, f)$ und da $\simeq$ von $\approx$ verfeinert wird, folgt auch $(e, e) \simeq (xf, f)$. Die Linksverträglichkeit von $\simeq$ liefert jetzt unmittelbar $(s, e) = (se, e) \simeq (sxf, f) = (t, f)$. □

Folgender Algorithmus teilt nun die Menge $F(S)$ in Konjugationsklassen ein. Gestartet wird mit einer Partition, in der jedes Element aus $F(S)$ eine einelementige Klasse bildet.

Algorithmus 3.2 Berechnung der Konjugationsklassen

Eingabe: Eine endliche Halbgruppe S und die Menge $F(S)$ von linked Pairs von S

```
1:  Q ← (F(S)/≈) \ { { (s,e) } | (s,e) ∈ F(S) }
2:  for all R ∈ Q do Union(R) end for
3:  while Q ≠ ∅ do
4:      wähle eine Menge R ∈ Q
5:      Q ← Q \ { R }
6:      for all a ∈ Σ do
7:          R' ← ∅
8:          for all (s,e) ∈ R do R' ← R' ∪ { Find((φ(a)s, e)) } end for
9:          if |R'| > 1 then
10:             Q ← Q ∪ { R' }
11:             Union(R')
12:         end if
13:     end for
14: end while
```

Wie bereits zuvor erwähnt, kann davon ausgegangen werden, dass $\mathcal{L}$- und $\mathcal{R}$-Klassen eines Elements in konstanter Zeit ermittelt werden können. Dies erfordert lediglich eine Vorverarbeitung der Halbgruppe mit einem Zeitaufwand in $\mathcal{O}(|\Sigma| \cdot |S|)$. Die Initialisierung des Stapels $\mathcal{Q}$ kann dann wie folgt implementiert werden: Für jede $\mathcal{R}$-Klasse von S wird ein leerer Eintrag zu $\mathcal{Q}$ hinzugefügt. Anschließend wird für jedes linked Pair (s, e) getestet, ob $s \mathrel{\mathcal{L}} e$ gilt. Im Fall $s \mathrel{\mathcal{L}} e$ wird die $\mathcal{R}$-Klasse von s bestimmt und das Element (s, e) zu dem entsprechenden Eintrag in $\mathcal{Q}$

hinzugefügt. Ansonsten wird (s, e) verworfen. Einträge mit weniger als zwei Elementen werden am Ende der Initialisierung wieder aus $\mathcal{Q}$ entfernt.

Der Rest des Kapitels ist dem Nachweis der Korrektheit und der Laufzeitanalyse des Algorithmus gewidmet. Für den folgenden Beweis benötigen wir den Begriff von verbindenden Sequenzen. Eine *verbindende Sequenz* sei eine Sequenz $((s_1, e_1), (s_2, e_2), \ldots, (s_k, e_k))$ von linked Pairs, sodass für alle $i \in \{ 1, \ldots, k-1 \}$ eine der beiden folgenden Bedingungen erfüllt ist:

1. $\forall a \in \Sigma$: $\mathrm{Find}((\varphi(a)s_i, e_i)) = \mathrm{Find}((\varphi(a)s_{i+1}, e_{i+1}))$
2. Der Stapel $\mathcal{Q}$ enthält eine Menge R mit $\{ (s_i, e_i), (s_{i+1}, e_{i+1}) \} \subseteq R$.

Zwei linked Pairs (s, e) und (t, f) heißen *verbunden*, falls eine verbindende Sequenz $((s_1, e_1), (s_2, e_2), \ldots, (s_k, e_k))$ mit $(s_1, e_1) = (s, e)$ und $(s_k, e_k) = (t, f)$ existiert.

Lemma 3.15. *Falls (s, e) und (t, f) nach Termination des Algorithmus in derselben Klasse liegen, dann liegen für alle $a \in \Sigma$ jeweils auch die Paare $(\varphi(a)s, e)$ und $(\varphi(a)t, f)$ in derselben Klasse.*

Beweis. Wir zeigen die allgemeinere Aussage, dass zwei linked Pairs (s, e) und (t, f) vor und nach Ablauf jeder Iteration der äußeren Schleife verbunden sind, sofern sie zu diesem Zeitpunkt in derselben Klasse der Partition liegen. Da bei Termination des Algorithmus der Stapel $\mathcal{Q}$ leer ist, folgt daraus direkt die Behauptung.

Vor der ersten Iteration ist diese Invariante erfüllt, denn zu diesem Zeitpunkt entsprechen die Mengen auf dem Stapel $\mathcal{Q}$ genau den nicht-trivialen Klassen der Partition. Zwei Paare, die in einer Iteration verbunden sind, sind auch in allen nachfolgenden Iterationen verbunden, denn sobald Bedingung (2) durch Entnahme einer Menge aus $\mathcal{Q}$ verletzt wird, wird noch im selben Schleifendurchlauf Bedingung (1) erfüllt. Noch zu zeigen ist, dass nach dem Zusammenführen zweier Paare (s, e) und (t, f) durch eine Operation $\mathrm{Union}(R')$ eine verbindende Sequenz zwischen (s, e) und (t, f) existiert. Dies folgt per Induktion: Seien (s', e') und (t', f') zwei Elemente aus R', sodass sich (s, e) (bzw. (t, f)) vor Ausführung der Union-Operation in derselben Klasse wie (s', e') (bzw. (t', f')) befindet. Dann existieren laut Induktionsvoraussetzung verbindende Sequenzen $((s, e), (s_2, e_2), \ldots, (s', e'))$ und $((t', f'), (t_2, f_2), \ldots, (t, f))$. Da unmittelbar vor der Union-Operation die Menge R' in $\mathcal{Q}$ aufgenommen wird, ist dann auch $((s, e), (s_2, e_2), \ldots, (s', e'), (t', f'), (t_2, f_2), \ldots, (t, f))$ eine verbindende Sequenz. □

Die Korrektheit des Algorithmus folgt nun aus der Charakterisierung der Konjugationsrelation aus Lemma 3.14:

Theorem 3.16. *Bei Termination des Algorithmus liegen zwei linked Pairs (s, e) und (t, f) genau dann in derselben Klasse der Partition, wenn $(s, e) \sim (t, f)$.*

Beweis. Wir zeigen, dass die durch die Partition induzierte Äquivalenzrelation die feinste linksverträgliche Äquivalenzrelation ist, die von $\approx$ verfeinert wird. Einen Beweis für die Linksverträglichkeit liefert Lemma 3.15. Zudem liegen bereits nach der Initialisierung alle Paare $(s, e), (t, f)$ mit $(s, e) \approx (t, f)$ in derselben Klasse und zwei Klassen werden nur vereinigt, wenn dies aufgrund der Forderung nach Linksverträglichkeit notwendig ist. □

Die Laufzeit des Algorithmus wird maßgeblich durch die Anzahl der ausgeführten Union- und Find-Operationen bestimmt. Das folgende Theorem liefert eine Abschätzung für diese Größen:

Theorem 3.17. *Der Algorithmus führt nicht mehr als*

- $|F(S)| - 1$ *atomare* Union-*Operationen und*
- $2|\Sigma| \cdot (|F(S)| - 1)$ Find-*Operationen aus.*

Beweis. Bei jeder atomaren Union-Operation nimmt die Anzahl an Klassen in der Partition um 1 ab. Da die disjunkte Vereinigung aller Klassen genau $|F(S)|$ Elemente enthält, ist die Anzahl der atomaren Union-Operationen also durch $|F(S)| - 1$ beschränkt. Wird eine Menge R in den Stapel $\mathcal{Q}$ eingefügt, so werden zugleich $|R| - 1$ atomare Union-Operationen ausgeführt. Seien $R_1, \ldots, R_k$ die Mengen, die während des Ablaufs des Algorithmus in $\mathcal{Q}$ aufgenommen werden, so gilt also

$$\sum_{i=1}^{k} (|R_i| - 1) \leqslant |F(S)| - 1.$$

In der Iteration, in der die Menge R_i aus dem Stapel entfernt wird, werden genau $|\Sigma| \cdot |R_i|$ Find-Operationen ausgeführt. Die Anzahl solcher Operationen insgesamt ist daher beschränkt durch

$$\sum_{i=1}^{k} |\Sigma| \cdot |R_i| \leqslant \sum_{i=1}^{k} |\Sigma| \cdot (2|R_i| - 2) \leqslant 2|\Sigma| \cdot (|F(S)| - 1),$$

wobei die erste Ungleichung gilt, da jede der Mengen R_i mindestens zwei Elemente enthält. □

Es existieren Implementierungen der Union-Find-Datenstruktur, die für eine Sequenz von n Union- und m Find-Operationen eine Laufzeit von $\mathcal{O}(n + m \cdot \alpha(n))$ benötigen. Dabei bezeichnet $\alpha(n)$ die Umkehrfunktion der extrem schnell wachsenden Ackermannfunktion $A(n, n)$. Für alle praktischen Eingabegrößen n ist $\alpha(n) \leqslant 5$. Die Laufzeit des gesamten Algorithmus liegt bei Verwendung einer solchen Variante der Union-Find-Datenstruktur in $\mathcal{O}(|\Sigma| \cdot |F(S)| \cdot \alpha(|F(S)|))$. In der Praxis verhält sich der Algorithmus aufgrund des extrem langsamen Wachstums der inversen Ackermannfunktion also beinahe wie ein Linearzeitalgorithmus.

4 Operationen auf Sprachen

In diesem Kapitel wird gezeigt, dass die Klasse der ω-regulären Sprachen unter booleschen Operationen (Vereinigung, Durchschnitt und Komplement) und unter längenerhaltenden Homomorphismen abgeschlossen ist. Hierzu werden explizite Konstruktionen auf erkennenden Homomorphismen angegeben, die es ermöglichen, die genannten Operationen effektiv zu berechnen.

4.1 Vereinigung

Zunächst soll untersucht werden, wie aus zwei erkennenden Homomorphismen φ und ψ ein erkennender Homomorphismus für die Vereinigung der erkannten ω-Sprachen konstruiert werden kann. Im Fall von endlichen Automaten und Büchi-Automaten kann eine Konstruktion genutzt werden, die oft als *Produktautomat* bezeichnet wird. Das Pendant bei erkennenden Homomorphismen über endlichen Wörtern ist das direkte Produkt. Letzteres kann auch im Fall von unendlichen Wörtern verwendet werden:

Proposition 4.1. *Seien S und T endliche Halbgruppen, $\varphi\colon \Sigma^+ \to S$ und $\psi\colon \Sigma^+ \to T$ Homomorphismen, $P \subseteq F(S)$ und $Q \subseteq F(T)$. Falls $L_\varphi(P)$ von φ und $L_\psi(Q)$ von ψ schwach (bzw. stark) erkannt werden, wird $L_\varphi(P) \cup L_\psi(Q)$ von $S \times T$ schwach (bzw. stark) erkannt.*

Beweis. Sei $\chi\colon \Sigma^+ \to S \times T$ der durch $\chi(a) = (\varphi(a), \psi(a))$ für alle $a \in \Sigma$ eindeutig definierte Homomorphismus und sei

$$R = \{\, ((s,t),(e,f)) \in F(S \times T) \mid (s,e) \in P \text{ oder } (t,f) \in Q \,\}.$$

Es wird zunächst gezeigt, dass $L_\chi(R)$ die Vereinigung der beiden Sprachen $L_\varphi(P)$ und $L_\psi(Q)$ ist. Ein Wort $\alpha \in \Sigma^\omega$ liegt genau dann in $L_\chi(R)$, wenn ein Paar $((s,t),(e,f)) \in R$ und eine Zerlegung $\alpha = uv_1v_2\cdots$ mit $\chi(u) = (s,t)$ und $\chi(v_i) = (e,f)$ für alle $i \geqslant 1$ existieren. Dies ist nach Korollar 2.9 genau dann der Fall, wenn es eine Zerlegung dieser Form gibt, sodass

$$\begin{aligned} &\varphi(u) = s \text{ und } \varphi(v_i) = e \text{ für alle } i \geqslant 1 \text{ und ein linked Pair } (s,e) \in P \text{ oder} \\ &\psi(u) = t \text{ und } \psi(v_i) = f \text{ für alle } i \geqslant 1 \text{ und ein linked Pair } (t,f) \in Q. \end{aligned}$$

Also ist $L_\chi(R) = L_\varphi(P) \cup L_\psi(Q)$, wie gewünscht.

Für den Fall der starken Erkennbarkeit betrachten wir $s, s' \in S$ und $t, t' \in T$, sowie ein beliebiges unendliches Wort $\alpha = uv_1v_2\cdots$ mit $\chi(u) = (s,t)$ und $\chi(v_i) = (s',t')$ für alle $i \geqslant 1$. Ist $\alpha \in L_\varphi(P)$, so gilt $\varphi^{-1}(s)(\varphi^{-1}(s'))^\omega \subseteq L_\varphi(P)$, da φ die Sprache $L_\varphi(P)$ stark erkennt. Hieraus folgt sofort $\chi^{-1}((s,t))(\chi^{-1}((s',t')))^\omega \subseteq L_\varphi(P) \subseteq L_\chi(R)$. Der Fall $\alpha \in L_\psi(Q)$ kann analog untersucht werden. □

4.2 Komplement und Durchschnitt

Nun werden Konstruktionen für Komplement und Durchschnitt untersucht. Im Fall von endlichen Wörtern ist die Komplementbildung sowohl bei endlichen Automaten, als auch bei erkennenden Homomorphismen trivial. Ganz anders verhält es sich bei Automaten über unendlichen Wörtern. Dort sind die Konstruktionen zur Komplementierung sehr aufwendig, vgl. [TFVT11]. Auch bei Homomorphismen über unendlichen Wörtern ist Komplementbildung in gewisser Hinsicht schwieriger als die Mengenvereinigung. Einfache Konstruktionen sind nur bei starker Erkennbarkeit möglich.

Proposition 4.2. *Seien S eine endliche Halbgruppe, $\varphi\colon \Sigma \to S$ ein Homomorphismus und L eine Sprache, die von φ stark erkannt wird. Dann wird auch $\Sigma^\omega \setminus L$ von φ stark erkannt.*

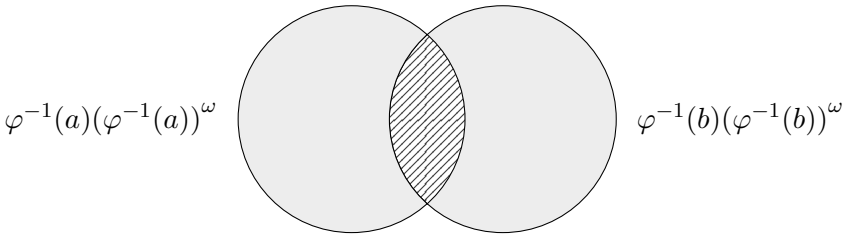
Beweis. Ohne Einschränkung sei φ surjektiv und $P \subseteq F(S)$ eine unter Konjugation abgeschlossene Menge mit $L_\varphi(P) = L$. Dann gilt nach Proposition 2.13, dass $L_\varphi(P) \cap L_\varphi(F(S) \setminus P) = \emptyset$. Andererseits ist $L_\varphi(P) \cup L_\varphi(F(S) \setminus P) = \Sigma^\omega$, also $L_\varphi(F(S) \setminus P) = \Sigma^\omega \setminus L_\varphi(P)$. □

Falls die vorliegende Menge P nicht bereits unter Konjugation abgeschlossen ist, kann sie mithilfe der in Abschnitt 2.3.4 beschriebenen Methode in eine solche umgewandelt werden. Als Folgerung von Proposition 4.1 und Proposition 4.2 ergibt sich das folgende weitere Resultat:

Korollar 4.3. *Seien S und T endliche Halbgruppen, $\varphi\colon \Sigma^+ \to S$ und $\psi\colon \Sigma^+ \to T$ Homomorphismen, $P \subseteq F(S)$ und $Q \subseteq F(T)$. Falls $L_\varphi(P)$ von φ und $L_\psi(Q)$ von ψ stark erkannt werden, wird $L_\varphi(P) \cap L_\psi(Q)$ von $S \times T$ stark erkannt.*

Beweis. Nach den Regeln von De Morgan gilt für zwei Sprachen $K, L \subseteq \Sigma^\omega$ die Gleichheit $K \cap L = \Sigma^\omega \setminus ((\Sigma^\omega \setminus K) \cup (\Sigma^\omega \setminus L))$. Die oben genannten Konstruktionen für Komplement und Vereinigung liefern nun eine Konstruktion für den Durchschnitt. □

Die folgende Abbildung visualisiert die Sprachen, die von den linked Pairs des Standardbeispiels $S = \{a, b\}$ mit der Verknüpfung $aa = ba = a$ und $ab = bb = b$ schwach erkannt werden, wenn man als erkennenden Homomorphismus $\varphi\colon \Sigma^+ \to S$ mit $\varphi(a) = a$ und $\varphi(b) = b$ wählt.



Der schraffiert dargestellte Durchschnitt der beiden Sprachen ist die Menge $(\Sigma^* a \Sigma^* b)^\omega$. Es ist offensichtlich nicht möglich, diesen durch die Vereinigung von Mengen der Form $\varphi^{-1}(s)(\varphi^{-1}(e))^\omega$ mit $(s, e) \in F(S)$ darzustellen. Überlappungen von durch linked Pairs erkannten Sprachen sind der Grund dafür, dass die oben vorgestellten Konstruktionen im Fall der schwachen

Erkennbarkeit nicht funktionieren. Als Ausweg kann aufgrund der in Abschnitt 2.3 gezeigten Äquivalenz von schwacher und starker Erkennbarkeit jedoch stets eine Umwandlung in einen stark erkennenden Homomorphismus vorgenommen werden. Um den Umweg über Büchi-Automaten zu vermeiden, kann hierzu für eine Halbgruppe S und einen Homomorphismus $\varphi\colon \Sigma^+ \to S$ folgende Konstruktion aus [CPP08] verwendet werden: Setze $T \coloneqq \mathcal{P}(S^1 \times S^1)$ mit $X \cdot Y = \{\, (s, ts't'), (sts', t') \mid (s,t) \in X, (s',t') \in Y \,\}$ und definiere einen Homomorphismus $\psi\colon \Sigma^+ \to T$ durch $\psi(a) = \{\, (\varphi(a), 1), (1, \varphi(a)) \,\}$ für alle $a \in \Sigma$. Jede von φ schwach erkannte Sprache wird von ψ stark erkannt.

4.3 Längenerhaltende Homomorphismen

Seien Σ und Γ Alphabete und $\tau\colon \Sigma^+ \to \Gamma^+$ ein Homomorphismus. Die Anwendung eines solchen Homomorphismus zwischen freien Halbgruppen kann wie folgt auf ω-Sprachen erweitert werden: Sei $L \subseteq \Sigma^\omega$, so ist $\tau(L) = \{\, \tau(a_1)\tau(a_2)\cdots \mid a_1a_2\cdots \in L \,\}$. Der Homomorphismus τ heißt *längenerhaltend*, falls $\tau(a) \in \Gamma$ für alle $a \in \Sigma$. Falls L eine ω-reguläre Sprache ist, ist auch $\tau(L)$ ω-regulär. Im Folgenden wird der Fall von längenerhaltenden Homomorphismen genauer untersucht, der für die Entscheidbarkeit von MSO-Logik von besonderem Interesse ist.

Grundlage der Konstruktion ist die folgende Beobachtung: Sei S eine Halbgruppe, so kann $\mathcal{P}(S)$ durch die Multiplikation $X \cdot Y = \{\, st \mid s \in X, t \in Y \,\}$ ebenfalls zu einer Halbgruppe ergänzt werden. Für einen Homomorphismus $\varphi\colon \Sigma^+ \to S$ sei $\varphi_\tau\colon \Gamma^+ \to \mathcal{P}(S)$ der durch $\varphi_\tau(a) = \varphi(\tau^{-1}(a))$ eindeutig festgelegte Homomorphismus.

Proposition 4.4. *Sei $\varphi\colon \Sigma^+ \to S$ ein Homomorphismus und $\tau\colon \Sigma^+ \to \Gamma^+$ ein längenerhaltender Homomorphismus. Falls L von φ schwach (bzw. stark) erkannt wird, so wird $\tau(L)$ von φ_τ schwach (bzw. stark) erkannt.*

Beweis. Sei $P \subseteq F(S)$ mit $L_\varphi(P) = L$. Sei Q die Menge aller linked Pairs (T, F) aus $\mathcal{P}(S) \times \mathcal{P}(S)$, für die ein $(s, e) \in P$ mit $s \in T$ und $e \in F$ existiert.

Wir untersuchen zunächst den Fall schwacher Erkennbarkeit und zeigen $L_{\varphi_\tau}(Q) = \tau(L)$. Ein unendliches Wort $\beta \subseteq \Gamma^\omega$ ist genau dann in $\tau(L)$, wenn ein Wort $\alpha = a_1a_2\cdots \in L$ mit $\tau(a_1)\tau(a_2)\cdots = \beta$ existiert. Für mindestens ein Paar $(s, e) \in P$ lässt sich das Wort α also als $\alpha = uv_1v_2\cdots$ mit $\varphi(u) = s$ und $\varphi(v_i) = e$ für alle $i \geqslant 1$ faktorisieren. Nach Konstruktion ist $s \in \varphi_\tau(\tau(u))$ und $e \in \varphi_\tau(\tau(v_i))$ für alle $i \geqslant 1$. Mit dem Argument aus Korollar 2.9 folgt dann $\beta = \tau(u)\tau(v_1)\tau(v_2)\cdots \in L_{\varphi_\tau}(Q)$. Der Beweis der umgekehrten Richtung erfolgt analog.

Für den Fall der starken Erkennbarkeit sei φ ohne Beschränkung der Allgemeinheit surjektiv. Wir fordern zudem, dass P unter Konjugation abgeschlossen ist und folgern, dass dann auch die Menge Q bei Einschränkung auf die Unterhalbgruppe $\varphi_\tau(\Gamma^+)$ unter Konjugation abgeschlossen sein muss. Sei (T, F) ein linked Pair von $\mathcal{P}(S)$, sodass gilt $\varphi_\tau^{-1}(T)(\varphi_\tau^{-1}(F))^\omega \cap \tau(L) \neq \emptyset$. Dann existieren $s \in T$, $t \in F$ und ein unendliches Wort $\alpha \in \varphi^{-1}(s)(\varphi^{-1}(t))^\omega \cap L$. Da P unter Konjugation abgeschlossen ist, folgt $(st^\pi, t^\pi) \in P$. Ferner gilt $st^n \in TF^n = T$ und $t^n \in F^n = F$ für alle $n \in \mathbb{N}$, also insbesondere $st^\pi \in T$ und $t^\pi \in F$. Dies zeigt $(T, F) \in Q$ und liefert damit den Nachweis der Abgeschlossenheit von Q unter Konjugation. □

5 Entscheidungsprobleme

Eine weitere Klasse interessanter Probleme sind sogenannte Entscheidungsprobleme. Entscheidungsprobleme sind Fragen über Objekte einer bestimmten Grundmenge, die für jedes Objekt aus dieser Menge mit der Antwort "ja" oder "nein" beantwortet werden können. Im Gegensatz zu den zuvor vorgestellten Algorithmen sollen also keine Objekte berechnet, sondern Eingaben auf gewisse Eigenschaften überprüft werden. Ein Beispiel für eine solche Eigenschaft ist folgende:

EINGABE: Ein surjektiver Homomorphismus $\varphi\colon \Sigma^+ \to S$ und eine Menge $P \subseteq F(S)$

FRAGE: Ist $L_\varphi(P) = \emptyset$?

Dieses Problem wird als *Leerheitsproblem* für Homomorphismen über unendlichen Wörtern bezeichnet. Für die Beantwortung dieser Frage reicht ein einfacher Test der Menge P auf Leerheit. Falls P leer ist, enthält offensichtlich auch die Menge $L_\varphi(P)$ kein Element. Falls P ein Element (s, e) enthält, existieren aufgrund der Surjektivität von φ zwei Wörter $u, v \in \Sigma^+$ mit $\varphi(u) = s$ und $\varphi(v) = e$ und es ist $uv^\omega \in L_\varphi(P)$. Dieser Test funktioniert auch dann, wenn φ die Sprache $L_\varphi(P)$ nur schwach erkennt.

Ein wesentlicher Grund für die Simplizität dieses Tests ist die Forderung nach Surjektivität des Homomorphismus. Ein effizienter Test ist jedoch auch noch dann möglich, wenn dies nicht vorausgesetzt wird: Die Unterhalbgruppe $\varphi(\Sigma^+)$ von S kann durch Berechnung der im rechten Cayley-Graphen von 1 aus erreichbaren Knoten in $\mathcal{O}(|\Sigma| \cdot |S|)$ Zeit bestimmt werden.

Eine zum Leerheitsproblem verwandte Fragestellung ist das *Universalitätsproblem*:

EINGABE: Ein surjektiver Homomorphismus $\varphi\colon \Sigma^+ \to S$ und eine Menge $P \subseteq F(S)$

FRAGE: Ist $L_\varphi(P) = \Sigma^\omega$?

Falls $L_\varphi(P)$ von φ stark erkannt wird, liefert das Verfahren zur Komplementbildung aus Abschnitt 4.2 unmittelbar eine effiziente Reduktion auf das Leerheitsproblem. Eine solche Reduktion funktioniert prinzipiell auch im Fall von schwacher Erkennbarkeit. In Abschnitt 4.2 wurde hierzu eine Konstruktion beschrieben, um einen schwach erkennenden Homomorphismus in einen stark erkennenden Homomorphismus umzuwandeln. Die dabei konstruierte Halbgruppe hat jedoch die Größe $2^{(n+1)^2}$, wenn n die Größe der ursprünglichen Halbgruppe ist. Besser wäre daher ein Werkzeug, mit dem das Universalitätsproblem für schwach erkennende Homomorphismen direkt gelöst werden kann. Hierbei wird der Begriff der Überdeckungen eine wichtige Rolle spielen.

Sei S eine endliche Halbgruppe und $\varphi\colon \Sigma^+ \to S$ ein Homomorphismus. Eine Menge $P \subseteq S \times S$ heißt *Überdeckung* einer Menge $Q \subseteq S \times S$, falls $L_\varphi(Q) \subseteq L_\varphi(P)$. Im Folgenden

sind insbesondere Überdeckungen von Mengen von linked Pairs von Interesse. So kann das Universalitätsproblem bei schwacher Erkennbarkeit auf einen Test reduziert werden, ob die gegebene Menge P eine Überdeckung von $F(S)$ ist. Auch das folgende, unter dem Namen *Inklusionsproblem* bekannte Problem ist mithilfe von Überdeckungen lösbar:

EINGABE: Homomorphismen $\varphi\colon \Sigma^+ \to S$, $\psi\colon \Sigma^+ \to T$ und Mengen $P \subseteq F(S)$, $Q \subseteq F(T)$

FRAGE: Ist $L_\varphi(P) \subseteq L_\psi(Q)$?

Ein einfaches Entscheidungsverfahren erhält man durch die Definition eines Homomorphismus $\chi\colon \Sigma^+ \to S \times T$ mit $\chi(u) = (\varphi(u), \psi(u))$ für alle $u \in \Sigma^+$. Ferner sei

$$P' = \{\, ((s,t),(e,f)) \mid (s,e) \in P, (t,f) \in F(T) \,\} \text{ und}$$
$$Q' = \{\, ((s,t),(e,f)) \mid (s,e) \in F(S), (t,f) \in Q \,\}.$$

Nun ist $L_\chi(P') = L_\varphi(P)$ und $L_\chi(Q') = L_\psi(Q)$. Es verbleibt zu testen, ob Q' eine Überdeckung von P' ist. Das *Äquivalenzproblem* erhält man, indem man die Teilmengenbeziehung in der Fragestellung des Inklusionsproblems durch Gleichheit ersetzt. Es kann durch zwei Tests $L_\varphi(P) \subseteq L_\psi(Q)$ und $L_\psi(Q) \subseteq L_\varphi(P)$ entschieden werden.

Darüber hinaus liefern Überdeckungen einen einfachen Test auf starke Erkennbarkeit. Sei $\varphi\colon \Sigma^+ \to S$ ein surjektiver Homomorphismus, $P \subseteq F(S)$ eine Menge und $\hat{P}$ der Abschluss von P unter Konjugation. Dann erkennt φ die Sprache $L_\varphi(P)$ genau dann stark, wenn P eine Überdeckung von $\hat{P}$ ist.

Von besonderem Interesse ist also das *Überdeckungsproblem*:

EINGABE: Ein surjektiver Homomorphismus $\varphi\colon \Sigma^+ \to S$ und Mengen $P, Q \subseteq F(S)$

FRAGE: Ist P eine Überdeckung von Q?

Es sei zunächst angemerkt, dass auch das Überdeckungsproblem eine einfache Lösung besitzt, wenn man starke Erkennbarkeit fordert. Für

$$\hat{P} = \bigcup_{(s,e)\in P} [s,e] \quad \text{und} \quad \hat{Q} = \bigcup_{(s,e)\in Q} [s,e]$$

ist $L_\varphi(\hat{Q}) \subseteq L_\varphi(\hat{P})$ genau dann, wenn $\hat{Q} \subseteq \hat{P}$. Wegen $L_\varphi(\hat{P}) = L_\varphi(P)$ und $L_\varphi(\hat{Q}) = L_\varphi(Q)$ liefert dies direkt einen Überdeckungstest für die Ausgangsmengen. Dementsprechend lassen sich auch das Inklusions- und Äquivalenzproblem mit der oben beschriebenen Reduktion bei starker Erkennbarkeit einfach lösen. Im Folgenden wird also lediglich schwache Erkennbarkeit vorausgesetzt.

Der hier vorgestellte Algorithmus löst das Entscheidungsproblem mit einer Worst-Case-Laufzeit in $\mathcal{O}(|\Sigma| \cdot |S|^3)$, bei einer durch eine Konstante nach oben beschränkten Größe von Q sogar in $\mathcal{O}(|\Sigma| \cdot |S|^2)$. Vor Beginn des eigentlichen Algorithmus werden der rechte Cayley-Graph von S und für jeden Knoten eine Liste aller eingehenden Kanten, gruppiert nach Beschriftung, berechnet. Dies ist mit einem Zeitaufwand in $\mathcal{O}(|\Sigma| \cdot |S|)$ möglich. Es kann daher davon ausgegangen werden, dass die Menge $s \cdot a^{-1} \coloneqq \{\, t \in S^1 \mid t\varphi(a) = s \,\}$ für alle $s \in S$ und alle $a \in \Sigma$ bekannt ist.

Die grundlegenden Datenstrukturen sind eine Warteschlange $\mathcal{Q}$ von Elementen aus $S \times S \times S$ sowie ein Feld R mit Indizes aus S und Elementen aus $\mathcal{P}(S \times S)$. Die Reihenfolge, in der Elemente aus $\mathcal{Q}$ entnommen werden, spielt für die Korrektheit des Algorithmus keine Rolle. Prinzipiell kann jede Datenstruktur verwendet werden, bei der Einfüge- und Löschoperationen in konstanter Zeit möglich sind.

Algorithmus 5.1 Algorithmus zum Testen auf Überdeckung

Eingabe: Ein surjektiver Homomorphismus $\varphi\colon \Sigma^+ \to S$ und Mengen $P, Q \subseteq F(S)$

```
1:  Q ← { (s, e, 1) | (s, e) ∈ Q }
2:  for all s ∈ S do R[s] ← ∅ end for
3:  while Q ≠ ∅ do
4:      wähle ein (s, x, y) ∈ Q
5:      Q ← Q \ { (s, x, y) }
6:      if (sx, yxyx) ∉ P then
7:          for all a ∈ Σ, p ∈ x · a^{-1} do
8:              if p = 1 then return "L_φ(P) ⊈ L_φ(Q)" end if
9:              if (s, φ(a)y) ∉ R[p] then
10:                 R[p] ← R[p] ∪ { (s, φ(a)y) }
11:                 Q ← Q ∪ { (s, p, φ(a)y) }
12:             end if
13:         end for
14:     end if
15: end while
16: return "L_φ(P) ⊆ L_φ(Q)"
```

Für den Korrektheitsbeweis wird zunächst das folgende technische Lemma benötigt:

Lemma 5.1. *Seien $u, v \in \Sigma^+$ und seien (s, e) und $(\varphi(u), \varphi(v))$ linked Pairs. Dann ist das unendliche Wort uv^ω genau dann in der Menge $\varphi^{-1}(s)(\varphi^{-1}(e))^\omega$ enthalten, wenn eine Zerlegung $v = v_1 v_2$ mit $\varphi(uv_1) = s$ und $\varphi(v_2 v v_1) = e$ existiert.*

Beweis. Sei $v = a_1 a_2 \cdots a_n$ mit $n \geqslant 1$ und $a_i \in \Sigma$ für alle $i \in \{1, \ldots, n\}$. Falls uv^ω in $\varphi^{-1}(s)(\varphi^{-1}(e))^\omega$ enthalten ist, dann existiert eine Zerlegung $uv^\omega = u' v_1' v_2' \cdots$ mit $\varphi(u') = s$ und $\varphi(v_i') = e$ für alle $i \geqslant 1$. Da u und v endliche Wörter sind, existieren Indizes $j \geqslant i \geqslant 1$, Potenzen $k, \ell \geqslant 1$ und eine Position $m \in \{1, \ldots, n\}$, sodass $u' v_1' v_2' \cdots v_{i-1}' = uv^k a_1 a_2 \cdots a_{m-1}$ und $v_i' v_{i+1}' \cdots v_j' = a_m a_{m+1} \cdots a_n v^\ell a_1 a_2 \cdots a_{m-1}$. Setze $v_1 = a_1 a_2 \cdots a_{m-1}$ und $v_2 = a_m a_{m+1} \cdots a_n$. Dann ist $v_1 v_2 = v$, sowie $\varphi(uv_1) = \varphi(uv^k a_1 a_2 \cdots a_{m-1}) = \varphi(u' v_1' v_2' \cdots v_{i-1}') = s$ und $\varphi(v_2 v v_1) = \varphi(a_m a_{m+1} \cdots a_n v^\ell a_1 a_2 \cdots a_{m-1}) = \varphi(v_i' v_{i+1}' \cdots v_j') = e$. Für die umgekehrte Richtung betrachtet man die Faktorisierung $uv^\omega = uv_1(v_2 v v_1)^\omega$. □

Eine Konsequenz des Lemmas ist, dass für ein linked Pair (s, e) aus der Existenz von endlichen Wörtern $u, v \in \Sigma^+$ mit $uv^\omega \in \varphi^{-1}(s)(\varphi^{-1}(e))^\omega$ direkt gefolgert werden kann, dass auch $u', v' \in \Sigma^+$ mit $\varphi(u') = s$, $\varphi(v') = e$ und $u'(v')^\omega = uv^\omega$ existieren.

Lemma 5.2. *Wenn der Algorithmus mit Ausgabe "$L_\varphi(Q) \subseteq L_\varphi(P)$" terminiert, dann ist $L_\varphi(Q) \setminus L_\varphi(P) = \emptyset$.*

Beweis. Wir starten mit der Annahme, dass die Differenz $L_\varphi(Q) \setminus L_\varphi(P)$ nicht-leer ist. Aus den Abschlusseigenschaften ω-regulärer Sprachen wissen wir, dass dann $L_\varphi(Q) \setminus L_\varphi(P)$ eine nicht-leere ω-reguläre Sprache ist und daher ein Wort α der Form $u(a_1a_2 \cdots a_n)^\omega$ enthält, wobei $u \in \Sigma^+$ und $a_i \in \Sigma$ für alle $i \in \{1, \ldots, n\}$. Nach Lemma 5.1 kann ohne Beschränkung der Allgemeinheit angenommen werden, dass $(s, e) \coloneqq (\varphi(u), \varphi(a_1a_2 \cdots a_n))$ ein linked Pair aus der Menge Q ist. Wir zeigen nun, dass bei Termination für alle $k \in \{1, \ldots, n-1\}$ das Tupel $(s, \varphi(a_{k+1}a_{k+2} \cdots a_n))$ in $R[\varphi(a_1a_2 \cdots a_k)]$ enthalten ist. Der Beweis erfolgt per Induktion über den Parameter k.

Den Induktionsanfang liefert $k = n - 1$. Spätestens bei Entnahme des Tripels $(s, e, 1)$ aus $\mathcal{Q}$ wird $(s, \varphi(a_n))$ in $R[\varphi(a_1a_2 \cdots a_{n-1})]$ aufgenommen. Sei nun $k < n - 1$. Per Induktionsannahme ist $(s, \varphi(a_{k+2}a_{k+3} \cdots a_n)) \in R[\varphi(a_1a_2 \cdots a_{k+1})]$, also wird das Tripel (s, x, y) mit $x = \varphi(a_1a_2 \cdots a_{k+1})$ und $y = \varphi(a_{k+2}a_{k+3} \cdots a_n)$ während Ablauf des Algorithmus zu $\mathcal{Q}$ hinzugefügt. Wir betrachten die Iteration der äußeren Schleife, in der dieses Tripel aus $\mathcal{Q}$ entnommen wird. Wegen $\alpha \notin L_\varphi(P)$ ist $(sx, yxyx) \notin P$ und es folgt $(s, \varphi(a_{k+1})y) \in R[\varphi(a_1a_2 \cdots a_k)]$.

Schließlich wird aufgrund $(s, \varphi(a_2a_3 \cdots a_n)) \in R[\varphi(a_1)]$ auch das Tripel $(s, \varphi(a_1), \varphi(a_2a_3 \cdots a_n))$ in $\mathcal{Q}$ aufgenommen. Da $1 \in \varphi(a_1) \cdot a_1^{-1}$ ist, terminiert der Algorithmus in der Iteration, in der dieses Tripel wieder aus $\mathcal{Q}$ entfernt wird, mit Ausgabe "$L_\varphi(Q) \not\subseteq L_\varphi(P)$". Die Aussage folgt nun durch Kontraposition. □

Der Beweis der umgekehrten Richtung erfolgt durch die Konstruktion eines Wortes in der Differenz $L_\varphi(Q) \setminus L_\varphi(P)$. Für jedes Tripel $(s, e, 1)$, das nach der Initialisierung in $\mathcal{Q}$ enthalten ist, sei $w[s, e, 1] = \varepsilon$. Wird später in der inneren Schleife ein Tripel $(s, p, \varphi(a)y)$ in $\mathcal{Q}$ eingefügt, so setzen wir zugleich $w[s, p, \varphi(a)y] = a \cdot w[s, p\varphi(a), y]$. Für ein Tripel (s, x, y), das noch nicht zu $\mathcal{Q}$ hinzugefügt wurde, ist $w[s, x, y]$ undefiniert. Eine wichtige Beobachtung ist, dass für $s, x, y \in S$ stets $\varphi(w[s, x, y]) = y$ gilt, sofern das Wort $w[s, x, y]$ definiert ist.

Lemma 5.3. *Wenn der Algorithmus mit Ausgabe "$L_\varphi(Q) \not\subseteq L_\varphi(P)$" terminiert, dann ist $L_\varphi(Q) \setminus L_\varphi(P) \neq \emptyset$.*

Beweis. Ist die Ausgabe "$L_\varphi(Q) \not\subseteq L_\varphi(P)$", so wird der Algorithmus durch die Anweisung in Zeile 8 abgebrochen. Sei (s, x, y) das unmittelbar vor der Termintation aus $\mathcal{Q}$ entnommene Tripel und sei $a \in \Sigma$ mit $1 \cdot \varphi(a) = x$. Wir betrachten ein beliebiges Wort $u \in \varphi^{-1}(s)$ und setzen $v \coloneqq a \cdot w[s, x, y]$. Zunächst folgt per Induktion $(s, xy) \in Q$. Dies zeigt $uv^\omega \in L_\varphi(Q)$, denn nach Wahl von u und v gilt $\varphi(u) = s$ und $\varphi(v) = \varphi(a)y = xy$. Für jede beliebige Zerlegung $v = av_1bv_2$ mit $v_1, v_2 \in \Sigma^*$ und $b \in \Sigma$ ist das Wort $w[s, \varphi(av_1), \varphi(bv_2)]$ definiert als bv_2 und daher gilt $(s\varphi(av_1b), \varphi(v_2av_1bv_2av_1b)) \notin P$. Durch die Bedingung in Zeile 6 wird unmittelbar vor der Termination außerdem sichergestellt, dass $(s\varphi(a), \varphi(v_1bv_2av_1bv_2a)) \notin P$ gilt. Nach Lemma 5.1 folgt daraus direkt $uv^\omega \notin L_\varphi(P)$. □

Zusammen liefern die letzten beiden Lemmata einen Korrektheitsbeweis für den Algorithmus zur Lösung des Überdeckungsproblems.

Theorem 5.4. *Der Entscheidungsalgorithmus für das Überdeckungsproblem hat eine Laufzeit in $\mathcal{O}(|\Sigma| \cdot |S|^3)$ und terminiert genau dann mit Antwort "$L_\varphi(Q) \subseteq L_\varphi(P)$", wenn P eine Überdeckung von Q ist.*

Beweis. Die Korrektheit des Algorithmus folgt aus Lemma 5.2 und Lemma 5.3. Für Abschätzung der Laufzeit beachte, dass für alle $s \in S$ die Menge $R[s]$ maximal $|S|^2$ Elemente enthält. Es werden während der Ausführung des Algorithmus also höchstens $|S|^3 + |S|^2$ Elemente zu $\mathcal{Q}$ hinzugefüht, wobei der zweite Summand die Anzahl der Elemente aus $\mathcal{Q}$ bei Initialisierung abschätzt. Folglich ist die Anzahl der Iterationen der äußeren Schleife ebenfalls durch $|S|^3+|S|^2$ beschränkt. Zudem ist $s \cdot a^{-1} \cap t \cdot a^{-1} = \emptyset$ für alle $s, t \in S$ mit $s \neq t$ und für alle $a \in \Sigma$. Jedes Element $p \in S$ wird in Zeile 7 also maximal $|\Sigma| \cdot (|S|^2 + |S|)$ mal betrachtet. Die Anzahl an Iterationen der inneren Schleife beträgt somit insgesamt höchstens $|\Sigma| \cdot (|S|^3 + |S|^2)$. □

Abschließend folgt noch eine untere Schranke für die Laufzeitkomplexität von Algorithmen zum Testen auf Universalität bei starker Erkennbarkeit.

Proposition 5.5. *Sei $k \geqslant 1$ und $\varepsilon > 0$ beliebig. Dann existiert kein Entscheidungsalgorithmus für das Universalitätsproblem bei starker Erkennbarkeit, dessen Laufzeit in $\mathcal{O}(|\Sigma|^k \cdot |S|^{2-\varepsilon})$ liegt.*

Beweis. Die Beweisführung erfolgt per Widerspruch. Wir nehmen an, dass ein deterministischer Algorithmus und eine Konstante $c \geqslant 1$ existieren, sodass jede Eingabe der Größe $n = |S|$ und $m = |\Sigma|/2$ in Zeit $T(n, m) \leqslant c \cdot m^k \cdot n^{2-\varepsilon}$ auf Universalität getestet werden kann. Da c, k und ε laut Voraussetzung konstant sind, existiert ein $m \in \mathbb{N}$ mit $2^{\varepsilon m} > 16c \cdot m^k$. Ein ausreichend großes Alphabet $\Sigma = \{ 1, \ldots, 2m \}$, sodass diese Bedingung an m erfüllt ist, soll im Folgenden betrachtet werden.

Definiere $\Sigma_1 \coloneqq \{ 1, \ldots, m \}$ und $\Sigma_2 \coloneqq \{ m+1, \ldots, 2m \}$, $S_1 \coloneqq (\mathcal{P}(\Sigma_1) \setminus \{ \emptyset \}) \times \{ \emptyset \}$ und $S_2 \coloneqq \{ \emptyset \} \times (\mathcal{P}(\Sigma_2) \setminus \{ \emptyset \})$. Nun formt $S_1 \cup S_2$ mit folgender Verknüpfung eine Halbgruppe S:

$$(A, C) \cdot (B, D) = \begin{cases} (\emptyset, C \cup D) & \text{falls } A = B = \emptyset \\ (A \cup B, \emptyset) & \text{sonst} \end{cases}$$

Ferner sei $\varphi\colon \Sigma^+ \to S$ definiert durch $\varphi(a) = (\{ a \}, \emptyset)$ für alle $a \in \Sigma_1$ und $\varphi(b) = (\emptyset, \{ b \})$ für alle $b \in \Sigma_2$. Die Mengen S_1 und S_2 bilden, zusammen mit der komponentenweisen Vereinigung, jeweils $\mathcal{J}$-triviale Unterhalbgruppen von S und jedes Paar aus $S_1 \times S_2$ ist ein linked Pair von S. Nach Lemma 3.3 liegen keine zwei verschiedenen solcher linked Pairs in derselben Konjugationsklasse, die Anzahl an Konjugationsklassen von S beträgt also mindestens $|S_1| \cdot |S_2| \geqslant 2^{m-1} \cdot 2^{m-1} = 4^{m-1}$. Die Größe von S ist $n = |S_1| + |S_2| \leqslant 2^m + 2^m = 2^{m+1}$.

Wir betrachten nun den Ablauf des Algorithmus auf der Eingabe φ und $P = F(S)$. Sei $(s_1, e_1), (s_2, e_2), \ldots, (s_\ell, t_\ell)$ die Sequenz von linked Pairs, für die geprüft wird, ob $(s_i, e_i) \in P$. Es ist $\ell \leqslant T(n, m) \leqslant c \cdot m^k \cdot n^{2-\varepsilon} < 4c \cdot m^k \cdot 2^{2m-\varepsilon m} = 16c \cdot m^k \cdot 2^{-\varepsilon m} \cdot 4^{m-1} < 4^{m-1}$ und folglich gibt es eine Konjugationsklasse $[s, e]$, in der kein Element auf Enthaltensein in P getestet wird. Da der Algorithmus deterministisch ist, liefert er auf Eingabe $P' \coloneqq P \setminus [s, e]$ dieselbe Ausgabe wie auf die Eingabe P. Nach Proposition 2.13 ist dies ein Widerspruch zur Korrektheit. □

6 Minimierung erkennender Homomorphismen

Stark erkennende Homomorphismen haben gegenüber anderen Akzeptanzmodellen für ω-reguläre Sprachen, wie Büchi-Automaten, den Vorteil, dass für jede Sprache L ein eindeutiges minimales Objekt existiert, das L beschreibt. Nach Abschnitt 2.3.5 ist die Minimierung eines stark erkennenden Homomorphismus äquivalent zur Berechnung der syntaktischen Halbgruppe einer Sprache. Die Kenntnis eines effizienten Minimierungsverfahrens ist vor allem dann hilfreich, wenn eine lange Sequenz von Operationen auf Halbgruppen ausgeführt wird. Durch Minimierung von Zwischenergebnissen kann verhindert werden, dass die entstehenden Objekte dabei zu groß werden.

6.1 Konstruktion der syntaktischen Halbgruppe

In Abschnitt 2.3.5 wurden die Begriffe der syntaktischen Kongruenz und der syntaktischen Halbgruppe eingeführt. Dieser Abschnitt beschreibt, wie diese Objekte effektiv berechnet werden können, d. h. wie bei gegebenem Homomorphismus $\varphi\colon \Sigma^+ \to S$ und Menge $P \subseteq S \times S$ die syntaktische Halbgruppe einer stark erkannten Sprache $L_\varphi(P)$ bestimmt werden kann.

Zugunsten einer etwas einfacheren Notation bietet es sich an, den Begriff der Abgeschlossenheit unter Konjugation für die Zwecke dieses Kapitels auf beliebige Teilmengen von $S \times S$ zu erweitern. Eine Menge $P \subseteq S \times S$ heißt *vollständig*, falls $P \cap F(S)$ unter Konjugation abgeschlossen ist und für alle $s, t \in S$ genau dann $(s, t) \in P$ gilt, wenn $(st^\pi, t^\pi) \in P$ ist. Aus Lemma 2.11 und Proposition 2.14 folgt nun direkt folgendes Resultat:

Proposition 6.1. *Sei S eine endliche Halbgruppe und $\varphi\colon \Sigma^+ \to S$ ein surjektiver Homomorphismus. Eine Sprache $L \subseteq \Sigma^\omega$ wird von φ genau dann stark erkannt, wenn $L = L_\varphi(P)$ für eine vollständige Menge P.*

Die Einschränkung auf eine vollständige Menge P erleichtert uns die folgende Definition: Für $P \subseteq S \times S$ sei die Äquivalenzrelation $\sim_P$ definiert durch $s \sim_P t$ genau dann, wenn

$$(xsy, z) \in P \Leftrightarrow (xty, z) \in P$$
$$(z, xsy) \in P \Leftrightarrow (z, xty) \in P$$

für alle $x, y \in S^1$ und $z \in S$. Ferner sei $s \approx_P t$ genau dann, wenn

$$(s, z) \in P \Leftrightarrow (t, z) \in P$$
$$(z, s) \in P \Leftrightarrow (z, t) \in P$$

für alle $z \in S$. Wir beweisen zunächst den folgenden Zusammenhang:

Lemma 6.2. *Sei $P \subseteq S \times S$ beliebig. Dann ist $\sim_P$ die gröbste Kongruenzrelation, die $\approx_P$ verfeinert.*

Beweis. Dass $\sim_P$ die Relation $\approx_P$ verfeinert, ist sofort zu sehen, wenn man in der Definition von $\sim_P$ den Fall $x = y = 1$ betrachtet. Seien nun $s, t, z \in S$ mit $s \sim_P t$ und $p, q, x, y \in S^1$ beliebig. Dann folgt aus $(xpsqy, z) \in P$ (bzw. $(z, xpsqy) \in P$) bereits $(xptqy, z) \in P$ (bzw. $(z, xptqy) \in P$) und umgekehrt. Also ist $psq \sim_P ptq$, was zeigt, dass $\sim_P$ eine Kongruenz ist.

Um zu zeigen, dass $\sim_P$ die gröbste Äquivalenzrelation mit den genannten Eigenschaften ist, werden $s, t \in S$ mit $s \not\sim_P t$ betrachtet. Notwendigerweise existieren dann $p, q \in S^1$ mit $psq \not\approx_P ptq$. Jede Äquivalenzrelation, bezüglich der s und t äquivalent sind, ist also entweder keine Kongruenz oder nicht feiner als $\approx_P$. □

Der nächste Satz ist das Hauptergebnis dieses Abschnittes:

Theorem 6.3. *Sei S eine endliche Halbgruppe, $\varphi\colon \Sigma^+ \to S$ ein surjektiver Homomorphismus und P eine vollständige Teilmenge von $S \times S$, sodass die Sprache $L = L_\varphi(P)$ von φ stark erkannt wird. Dann ist $S/\sim_P$ isomorph zur syntaktischen Halbgruppe $\Sigma^+/\equiv_L$.*

Beweis. Seien $x, y \in \Sigma^*$ beliebig. Da P vollständig ist, liegt für alle $z \in \Sigma^+$ das unendliche Wort $xuyz^\omega$ genau dann in L, wenn $(\varphi(xuy), \varphi(z)) \in P$ gilt. Analog ist für alle $z \in \Sigma^*$ das Wort $z(xuy)^\omega$ genau dann in L, wenn $(\varphi(z), \varphi(xuy)) \in P$ ist. Diese beiden Beobachtungen zeigen die Äquivalenz $u \equiv_L v \Leftrightarrow \varphi(u) \sim_P \varphi(v)$. Die Abbildung $\varphi \circ \eta_L^{-1}$ ist also ein wohldefinierter Isomorphismus zwischen $\Sigma^+/\equiv_L$ und $S/\sim_P$. □

Abschließend sei noch einmal angemerkt, dass die Einschränkung auf vollständige Mengen in diesem Kapitel lediglich zugunsten einer einfacheren Notation gewählt wurde. In der Praxis bietet es sich an, lediglich die in einer vollständigen Menge enthaltenen linked Pairs zu speichern. Die Berechnung einer solchen Darstellung bei Eingabe einer beliebigen Menge P wird in Abschnitt 2.3.4 beschrieben.

6.2 Effiziente Berechnung der syntaktischen Halbgruppe

Im vorigen Abschnitt wurde beschrieben, wie bei gegebenem Homomorphismus $\varphi\colon \Sigma^+ \to S$ und gegebener Menge $P \subseteq S \times S$ die syntaktische Halbgruppe der von φ stark erkannten Sprache $L_\varphi(P)$ konstruiert werden kann. Nun wird ein Algorithmus, basierend auf einem Ansatz von Hopcroft zur Minimierung von endlichen Automaten [Hop71], beschrieben, der diese Konstruktion effizient berechnet.

Wie der Algorithmus zur Berechnung von Konjugationsklassen von linked Pairs aus Abschnitt 3.3, arbeitet der Minimierungsalgorithmus mit einer Partition $\mathcal{P}$ und wir identifizieren eine Partition mit der dadurch induzierten Äquivalenzrelation, sofern es sich anbietet. Anders als beim dort beschriebenen Verfahren wird bei der Minimierung jedoch mit einer groben Partition gestartet, die dann schrittweise verfeinert wird, bis die Klassen schließlich den

Äquivalenzklassen bezüglich $\sim_P$ entsprechen. Die zugrundeliegende Datenstruktur wird in der englischsprachigen Literatur häufig als *partition refinement data structure* bezeichnet und stellt eine Operation Split$(\mathcal{P}, \mathcal{Q}, R)$ zur Verfügung. Hierbei ist $\mathcal{P}$ eine Partition, $\mathcal{Q}$ eine Warteschlange und R eine Teilmenge der Grundmenge von $\mathcal{P}$. Die Semantik dieser Operation ist in dem folgenden Algorithmus dargestellt. Es sei jedoch bereits an dieser Stelle angemerkt, dass aus Effizienzgründen in der Regel eine andere Implementierung gewählt wird.

Algorithmus 6.1 Split-Operation zur Verfeinerung einer Partition

```
 1: procedure Split(P, Q, R)
 2:     for all C ∈ P do
 3:         C_1 ← C ∩ R
 4:         C_2 ← C \ R
 5:         if C_1 ≠ ∅ ∧ C_2 ≠ ∅ then
 6:             P ← (P \ { C }) ∪ { C_1, C_2 }
 7:             if C ∈ Q then
 8:                 Q ← (Q \ { C }) ∪ { C_1, C_2 }
 9:             else if |C_1| ⩽ |C_2| then
10:                 Q ← Q ∪ { C_1 }
11:             else
12:                 Q ← Q ∪ { C_2 }
13:             end if
14:         end if
15:     end for
16: end procedure
```

Die Idee einer effizienten Implementierung ist, für jedes Element eine Referenz auf sein Vorkommen in der Klasse der Partition zu verwalten. Dadurch kann die unnötige Betrachtung der von einer Operation nicht betroffenen Klassen vermieden werden.

Lemma 6.4. *Die Operation* Split$(\mathcal{P}, \mathcal{Q}, R)$ *kann in Laufzeit* $\mathcal{O}(|R|)$ *ausgeführt werden.*

Beweis. Verwalte die Klassen von $\mathcal{P}$ als doppelt verkettete Listen. Führe zusätzliche Zeiger ein, sodass Elemente $s \in S$ in konstanter Zeit in der Liste, welche die zugehörige Klasse aus $\mathcal{P}$ beschreibt, lokalisiert werden können. Iteriere über alle $s \in R$ und erstelle für jede zu einem solchen s zugehörige Klasse C mit $C \cap R \neq \emptyset$ eine neue Liste C', die mit C durch einen Zeiger verknüpft wird. Iterierte nochmals über alle $s \in R$, entferne s jeweils in konstanter Zeit aus der momentanen Klasse C und füge s in die neue, jeweils von C aus referenzierte Liste C' ein. Aktualisiere dabei alle von der Verschiebung betroffenen Zeiger. □

Es folgt die Hauptroutine zur Minimierung. Für $C \subseteq S$ und $a \in \Sigma$ seien die Mengen $C \cdot a^{-1}$ und $a^{-1} \cdot C$ definiert als

$$C \cdot a^{-1} \coloneqq \{\, s \in S \mid s\varphi(a) \in C \,\} \text{ und}$$
$$a^{-1} \cdot C \coloneqq \{\, s \in S \mid \varphi(a)s \in C \,\}.$$

Wie bereits in Kapitel 5 beschrieben kann nach einem Vorverarbeitungsschritt mit einer Laufzeit in $\mathcal{O}(|\Sigma| \cdot |S|)$ davon ausgegangen werden, dass diese Mengen für alle $C \subseteq S$ und alle $a \in \Sigma$ zur Verfügung stehen.

Algorithmus 6.2 Hopcroft-Algorithmus zur Minimierung eines Homomorphismus

Eingabe: Ein surjektiver Homomorphismus $\varphi\colon \Sigma^+ \to S$, eine vollständige Menge $P \subseteq S \times S$

1: $\mathcal{P} \leftarrow S/{\approx_P}$
2: $\mathcal{Q} \leftarrow \mathcal{P}$
3: **while** $\mathcal{Q} \neq \emptyset$ **do**
4: wähle eine Klasse $C \in \mathcal{Q}$
5: $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{ C \}$
6: **for all** $a \in \Sigma$ **do**
7: Split$(\mathcal{P}, \mathcal{Q}, C \cdot a^{-1})$
8: Split$(\mathcal{P}, \mathcal{Q}, a^{-1} \cdot C)$
9: **end for**
10: **end while**

Das folgende Lemma zeigt, dass die durch die Partition induzierte Äquivalenzrelation bei Termination des Algorithmus eine Kongruenz bildet:

Lemma 6.5. *Falls s und t nach Termination des Algorithmus in derselben Klasse liegen, dann liegen für alle $a \in \Sigma$ auch die Paare $\varphi(a)s$ und $\varphi(a)t$ (bzw. $s\varphi(a)$ und $t\varphi(a)$) jeweils in derselben Klasse.*

Beweis. Betrachte $s, t \in S$ und ein $a \in \Sigma$, sodass $\varphi(a)s$ und $\varphi(a)t$ nach Termination in verschiedenen Klassen liegen. Wir untersuchen den Zeitpunkt, zu dem diese Elemente getrennt werden. Dies findet bei der Initialisierung oder bei einer Split-Operation statt. In beiden Fällen wird darauffolgend eine Menge, die entweder $\varphi(a)s$ oder $\varphi(a)t$ enthält, in $\mathcal{Q}$ hinzugefügt. Sobald diese Menge C aus $\mathcal{Q}$ entnommen wird, wird durch die Operation Split$(\mathcal{P}, \mathcal{Q}, a^{-1} \cdot C)$ sichergestellt, dass auch s und t in verschiedenen Klassen enthalten sind. Eine symmetrisches Argument gilt für den rechtsseitigen Fall. □

Für den Korrektheitsbeweis wird auf die Charakterisierung der Relation $\sim_P$ aus Lemma 6.2 zurückgegriffen:

Theorem 6.6. *Bei Termination des Algorithmus sind zwei Elemente s und t genau dann in derselben Klasse der Partition, wenn $s \sim_P t$ gilt.*

Beweis. Wir zeigen, dass die durch die Partition induzierte Äquivalenzrelation gröbste Kongruenzrelation ist, die $\approx_P$ verfeinert. Nach Lemma 6.5 ist die Relation eine Kongruenz. Die Initialisierung der Partition erfolgt mit den Äquivalenzklassen von $\approx_P$. Jede Operation Split$(\mathcal{P}, \mathcal{Q}, R)$ resultiert in einer Partition $\mathcal{P}'$, die $\mathcal{P}$ verfeinert. Es existiert keine gröbere Relation mit diesen Eigenschaften, da jeder der vom Algorithmus ausgeführten Verfeinerungsschritte notwendig ist, um die Links- oder Rechtsverträglichkeit zu gewährleisten. □

Die Idee, bei jeder Aufteilung einer Klasse durch eine Split-Operation jeweils nur die kleinere der resultierenden Klassen in $\mathcal{Q}$ aufzunehmen, ist grundlegend für die Laufzeit des Algorithmus und erlaubt die folgende Abschätzung:

Theorem 6.7. *Seien $C_1, \ldots, C_k$ die Elemente, die im Laufe des Algorithmus durch die Anweisung in Zeile 5 aus $\mathcal{Q}$ entfernt werden. Dann ist*

$$\sum_{i=1}^{k} \sum_{a \in \Sigma} (|C_i \cdot a^{-1}| + |a^{-1} \cdot C_i|) \leqslant 2|\Sigma| \cdot |S| \log |S|.$$

Beweis. Sei $s \in S$ beliebig. Zu jedem Zeitpunkt gibt es höchstens eine Menge in $\mathcal{Q}$, die s enthält. Wenn eine solche Menge C durch die Anweisung in Zeile 5 aus $\mathcal{Q}$ entfernt und zu einem späteren Zeitpunkt eine Menge C' mit $s \in C'$ in $\mathcal{Q}$ aufgenommen wird, gilt $|C'| \leqslant \frac{1}{2}|C|$. Daher gilt für alle $s \in S$ die Ungleichung $n_s \coloneqq |\{\, i \mid s \in C_i \,\}| \leqslant \log |S|$ und folglich ist

$$\sum_{i=1}^{k} \sum_{a \in \Sigma} (|C_i \cdot a^{-1}| + |a^{-1} \cdot C_i|) = \sum_{a \in \Sigma, s \in S} (n_{s\varphi(a)} + n_{\varphi(a)s}) \leqslant 2|\Sigma| \cdot |S| \log |S|. \qquad \square$$

Aus dieser Abschätzung folgt sofort, dass die Laufzeit der äußeren Schleife bei Verwendung der Datenstruktur aus dem Beweis von Lemma 6.4 in $\mathcal{O}(|\Sigma| \cdot |S| \log |S|)$ liegt.

Nun muss noch die Initialisierung betrachtet werden. Eine einfache Implementierung der Operation $\mathcal{P} \leftarrow S/{\approx_P}$ ist nachfolgend dargestellt. Die Belegung des zweiten Parameters der Split-Operation mit $\emptyset$ soll andeuten, dass keine Warteschlangen-Operationen durchgeführt werden. Die Warteschlange $\mathcal{Q}$ wird ohnehin unmittelbar nach der Initialisierung mit Elementen gefüllt.

Algorithmus 6.3 Hopcroft-Initialisierung $\mathcal{P} \leftarrow S/{\approx_P}$

```
1: P ← { S }
2: for all s ∈ S do
3:     Split(P, ∅, { t ∈ S | (s, t) ∈ P })
4:     Split(P, ∅, { t ∈ S | (t, s) ∈ P })
5: end for
```

Die Laufzeit dieser Implementierung ist quadratisch in der Anzahl der Elemente von S und dominiert daher die Laufzeit des eigentlichen Algorithmus. Die Frage, ob eine (wesentlich) effizientere Methode zur Initialisierung existert, kann durch Proposition 5.5 teilweise beantwortet werden: Läge die Laufzeit des gesamten Algorithmus in $\mathcal{O}(|\Sigma|^k \cdot |S|^{2-\varepsilon})$ für ein $\varepsilon > 0$, könnte auch das Universalitätsproblem mit diesem Zeitaufwand gelöst werden, indem die Minimierung durchgeführt und das Resultat mit der trivialen Halbgruppe verglichen wird. Die beschriebene Routine zur Initialisierung ist also in gewissem Sinne optimal. Es sei jedoch angemerkt, dass der Optimalitätsbegriff aufgrund der Abhängigkeit von den beiden Parametern $|\Sigma|$ und $|S|$ etwas unscharf ist. Proposition 5.5 schließt beispielsweise die Existenz eines Algorithmus zur Initialisierung mit einer Laufzeit von $\mathcal{O}(2^{|\Sigma|} \cdot |S|)$ a priori nicht aus.

7 Experimentelle Ergebnisse

Ein weiterer Formalismus zur Beschreibung von ω-Sprachen ist *monadische Logik zweiter Stufe* über unendlichen Wörtern, häufig auch als MSO-Logik (engl. *monadic second-order logic*) bezeichnet. Die Definition von Sprachen durch prädikatenlogische Formeln erlaubt es, viele Eigenschaften von Systemen auf einfache Art und Weise zu beschreiben und ist daher vor allem im Bereich des *Model Checking* sehr verbreitet. Nach einem bekannten Resultat von Büchi [Büc62] entspricht die Klasse der durch MSO-Formeln definierbaren ω-Sprachen genau den ω-regulären Sprachen. Daher ist es möglich, für jede MSO-Formel einen stark erkennenden Homomorphismus zu berechnen, der dieselbe Sprache beschreibt. Anschließend können diese, wie in Kapitel 5 beschrieben, auf verschiedene Eigenschaften getestet werden. Um die Eignung von erkennenden Homomorphismen und Halbgruppen für den Einsatz in Werkzeugen zur Modellprüfung zu untersuchen, wurde die Umwandlung von MSO-Formeln in stark erkennende Homomorphismen implementiert und auf einigen Formeln getestet. Vor der Präsentation der Ergebnisse dieser Tests folgt eine kurze Einführung in die Syntax und Semantik der MSO-Logik. Für eine detailliertere Beschreibung sei auf [Tho97] verwiesen.

In MSO-Formeln treten zwei Arten von Variablen auf: Variablen erster Ordnung, die im Folgenden mit Kleinbuchstaben bezeichnet werden, und Variablen zweiter Ordnung, die mit Großbuchstaben bezeichnet werden. Die Syntax von MSO-Formeln ist

$$\varphi \ ::= \ \lambda(x) = a \ \mid \ x \leqslant y \ \mid \ x \in X \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \exists x\, \varphi \ \mid \ \exists X\, \varphi$$

wobei x, y Variablen erster Ordnung, X eine Variable zweiter Ordnung und $a \in \Sigma$ ein Buchstabe sind. Häufig werden zudem einige Abkürzungen benutzt:

$$\begin{aligned}
\varphi \wedge \psi &:= \neg(\neg\varphi \vee \neg\psi) \\
\varphi \to \psi &:= \neg\varphi \vee \psi \\
\forall x\, \varphi &:= \neg\exists x\, \neg\varphi \\
\forall X\, \varphi &:= \neg\exists X\, \neg\varphi \\
x = y &:= x \leqslant y \wedge y \leqslant x \\
x \neq y &:= \neg x = y \\
x < y &:= x \leqslant y \wedge x \neq y \\
x \notin X &:= \neg x \in X \\
y = x + 1 &:= x < y \wedge \forall z\, (z \leqslant x \vee y \leqslant z) \\
\lambda(x) = \lambda(y) &:= \bigvee_{a \in \Sigma} (\lambda(x) = a \wedge \lambda(y) = a) \\
\lambda(x) \neq \lambda(y) &:= \neg\lambda(x) = \lambda(y)
\end{aligned}$$

Variablen erster Ordnung werden als Positionen in einem unendlichen Wort interpretiert und die atomare Formel $\lambda(a) = x$ ist erfüllt, falls sich an Position x im Wort der Buchstabe a befindet. Variablen zweiter Ordnung entsprechen Mengen von Positionen. Die MSO-Formel

$$\exists X\, \forall x \forall y \forall z\, (x \in X \wedge y = x + 1 \wedge z = y + 1) \rightarrow (y \notin X \wedge z \in X \wedge \lambda(x) = \lambda(y)) \wedge \exists x\, (x \in X)$$

beschreibt über dem Alphabet $\Sigma = \{\, a, b \,\}$ beispielsweise die Sprache $\Sigma^*(aa|bb)^\omega$.

Die Umwandlung einer MSO-Formel in einen stark erkennenden Homomorphismus erfolgt nach demselben Schema wie bei Büchi-Automaten [Tho97]. Für boolesche Operationen können die Konstruktionen aus Abschnitt 4.1 und Abschnitt 4.2 verwendet werden, die Berechnungsvorschrift für Formeln der Form $\exists x\, \varphi$ und $\exists X\, \varphi$ basiert auf der Technik aus Abschnitt 4.3. Bei der praktischen Umsetzung ist es zudem empfehlenswert, alle Zwischenergebnisse mithilfe des Verfahrens aus Kapitel 6 zu minimieren. Formeln mit freien Variablen werden wie im Fall von Büchi-Automaten über einem erweiterten Alphabet $\Sigma \times \{\, 0, 1 \,\}^\ell$ mit $\ell \geqslant 1$ interpretiert.

In der nachfolgenden Tabelle sind für einige Formeln jeweils die Größen der berechneten syntaktischen Halbgruppen $|S|$, die Anzahl der linked Pairs $|F|$, sowie die Anzahl der akzeptierenden linked Pairs $|P|$ (für eine unter Konjugation abgeschlossene Menge P) zu finden:

MSO-Formel	$\lvert S\rvert$	$\lvert F\rvert$	$\lvert P\rvert$
$\exists X\, \forall x \forall y \forall z\, (x \in X \wedge y = x + 1 \wedge z = y + 1) \rightarrow (y \notin X \wedge z \in X \wedge \lambda(x) = \lambda(y)) \wedge \exists x\, (x \in X)$	14	25	24
$\forall X\, (\exists x \forall y\, (x \leqslant y \wedge x \notin X) \wedge \forall x \forall y \forall z\, ((y = x + 1 \wedge z = y + 1 \wedge x \notin X) \rightarrow z \notin X)) \rightarrow (\forall x\, (x \in X \rightarrow \lambda(x) = a))$	7	12	4
$\forall x \forall y\, (x < y \wedge \lambda(x) = a \wedge \lambda(y) = a) \rightarrow \exists z_1 \exists z_2 \exists z_3\, (x < z_1 \wedge z_1 < z_2 \wedge z_2 < z_3 \wedge z_3 < y)$	19	43	33
$\forall x \forall y\, (x = y + 1) \rightarrow \lambda(x) \neq \lambda(y)$	5	6	3

Weiterhin wurde die Größe der entstehenden Zwischenergebnisse untersucht, da diese maßgeblich für den Speicher- und Zeitaufwand bei der Verarbeitung größerer Formeln ist. Zu diesem Zweck wurden drei Sequenzen von Formeln mit Parameter $k \geqslant 1$ und freien Variablen zweiter Ordnung $X_1 = X_{k+1}, X_2, \ldots, X_k$ betrachtet:

$$\begin{aligned}
\varphi_k &= \forall x \bigwedge_{i=1}^{k} \exists y\, (x < y \wedge y \in X_i) \\
\psi_k &= \forall x \forall y\, (y = x + 1) \rightarrow \bigwedge_{i=1}^{k} (x \in X_i \rightarrow y \in X_{i+1}) \\
\chi_k &= \forall x \bigwedge_{i=1}^{k} (x \in X_i \rightarrow \exists y\, (x < y \wedge (y \in X_{i-1} \vee y \in X_{i+1})))
\end{aligned}$$

Testergebnisse für die Parameterwerte $k \in \{2, 3, 4, 5, 6\}$ sind nachfolgend zu sehen. Alle Berechnungen wurden auf einem System mit einem Intel Core i5-3320M und 4GiB RAM ausgeführt. Die Ausführungszeit lag für jede der Berechnungen unter drei Sekunden.

	φ_k			ψ_k			χ_k		
	|S|	|F|	|P|	|S|	|F|	|P|	|S|	|F|	|P|
$k = 2$	4	5	1	12	15	10	7	14	11
$k = 3$	8	22	1	43	50	41	11	26	15
$k = 4$	16	74	1	148	163	146	17	61	30
$k = 5$	32	232	1	539	570	537	41	227	85
$k = 6$	64	710	1	1863	1926	1861	105	716	184

Halbgruppen mit einer Größe von bis zu 30 000 Elementen ließen sich auf dem oben genannten System problemlos speichern und weiterverarbeiten. Die Testergebnisse zeigen also, dass stark erkennende Homomorphismen nicht nur aus theoretischer Sicht interessante Objekte, sondern für Entscheidungsprobleme auf MSO-Logik über unendlichen Wörtern durchaus konkurrenzfähig zu Automatenmodellen sind. Der Hauptvorteil von stark erkennenden Homomorphismen ist, dass alle Zwischenergebnisse durch einen Minimierungsschritt kompakt dargestellt werden können und dass die Komplementbildung extrem effizient ist. Bei Büchi-Automaten müssen hingegen Heuristiken zur Minimierung der Teilergebnisse eingesetzt werden oder es werden lediglich Fragmente der MSO-Logik betrachtet, in denen effizientere Konstruktionen auf den zugehörigen Automatenmodellen möglich sind.

8 Zusammenfassung und Ausblick

In dieser Arbeit wurden grundlegende Algorithmen für (schwach und stark) erkennende Homomorphismen vorgestellt. Zudem wurde durch Tests bestätigt, dass sich diese Verfahren auch in der Praxis eignen.

Die Resultate aus Kapitel 3 geben einen Einblick in wichtige Eigenschaften von linked Pairs und Konjugationsklassen. Ein Ansatzpunkt für zukünftige Arbeiten ist die Frage, inwiefern schärfere Abschätzungen für andere interessante Klassen gegeben werden können. Es sei jedoch angemerkt, dass das Enthaltensein von Worst-Case-Beispielen in der Familie der $\mathcal{J}$-trivialen Halbgruppen zeigt, dass schärfere Schranken für viele Klassen nicht möglich sind, sofern man nicht weitere Parameter betrachtet. So sind die $\mathcal{J}$-trivialen Halbgruppen beispielsweise in der Klasse der sogenannten *aperiodischen* Halbgruppen enthalten, die auf Logikebene mit *linearer temporaler Logik* und *Prädikatenlogik erster Stufe* korrespondiert.

Kapitel 4 beschreibt alle Operationen auf Sprachebene, die für die Umwandlung von MSO-Formeln in stark erkennende Homomorphismen benötigt werden. Es existiert eine Vielzahl weiterer interessanter Operationen, die in dieser Arbeit nicht behandelt wurden, darunter (beliebige) Homomorphismen, inverse Homomorphismen und Residuenbildung.

Der Algorithmus zum Testen auf Überdeckung aus Kapitel 5 hat im Allgemeinen eine Laufzeit in $\mathcal{O}(|\Sigma| \cdot |S|^3)$. Nach Proposition 5.5 ist nicht mit einer Laufzeit in $o(|\Sigma| \cdot |S|^2)$ zu rechnen. Dennoch unterscheiden sich die beiden Schranken um den Faktor $|S|$ und es wäre wünschenswert, diese Lücke zu schließen.

In Kapitel 6 wurde ein auf dem Hopcroft-Verfahren basierender Algorithmus zur Berechnung der syntaktischen Halbgruppe einer gegebenen Sprache vorgestellt. Mit dem Resultat aus Proposition 5.5 folgt, dass die Laufzeit dieses Verfahrens in gewissem Sinne optimal ist. Es wäre interessant zu sehen, ob die dort angegebene untere Schranke weiter verbessert werden kann. Insbesondere ist nicht klar, ob für konstante Alphabetgröße eine Schranke von $\mathcal{O}(|S|^{2-\varepsilon})$ nachgewiesen werden kann. Zudem gibt es für den praktischen Einsatz des Algorithmus noch Optimierungsbedarf, beispielsweise bei der Initialisierung der Partition.

Ein Problem, das sich bei der Speicherung von erkennenden Homomorphismen ergibt, ist die Größe der Menge der akzeptierenden linked Pairs. Die experimentellen Ergebnisse aus Kapitel 7 lassen vermuten, dass die Anzahl an linked Pairs und die Größe der Halbgruppe bei der Umwandlung von MSO-Formeln häufig in derselben Größenordnung liegen. Dennoch zeigt das Beispiel am Ende von Abschnitt 3.1, dass die Anzahl an linked Pairs tatsächlich quadratisch in der Größe der Halbgruppe sein kann. Alternative Ansätze zur Kodierung der linked Pairs könnten hier hilfreich sein.

Eine implizite Kodierung der akzeptierenden linked Pairs ist beispielsweise bei der Konstruktion des erkennenden Homomorphismus im Beweis von Theorem 2.12 möglich. Ob ein linked Pair (R, E) akzeptierend ist kann in diesem Fall direkt aus der Matrixdarstellung von R und E abgeleitet werden. Ferner kann in dieser Repräsentation das von einem Element T erzeugte idempotente Element T^π durch den Algorithmus von Warshall effizient bestimmt werden. Diese beiden Beobachtungen erlauben es, Paare von Elementen in solchen Halbgruppen schnell auf Enthaltensein in einer vollständigen Menge zu prüfen, wodurch eine effiziente Minimierung möglich ist. Es wäre interessant zu sehen, inwiefern sich diese Art der Minimierung zur Reduktion der Größe von Büchi-Automaten eignet.

Literaturverzeichnis

[Arn85] A. Arnold. A syntactic congruence for rational ω-languages. *Theor. Comput. Sci.*, 39:333–335, 1985.

[Büc62] J. R. Büchi. On a Decision Method in Restricted Second-Order Arithmetic. In *Int. Congr. for Logic, Methodology, and Philosophy of Science*, S. 1–11. Stanford Univ. Press, 1962.

[CPP08] O. Carton, D. Perrin, J.-É. Pin. *Automata and semigroups recognizing infinite words.* HAL - CCSD, 2008.

[Hop71] J. E. Hopcroft. An N Log N Algorithm for Minimizing States in a Finite Automaton. Technischer Bericht, Stanford, CA, USA, 1971.

[Kle56] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon, J. McCarthy, Herausgeber, *Automata Studies*, Nummer 34 in Annals of Mathematics Studies, S. 3–40. Princeton University Press, 1956.

[Mic88] M. Michel. Complementation is more difficult with automata on infinite words, 1988. CNET, Paris.

[MP43] W. S. McCulloch, W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[MS72] A. R. Meyer, L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *Proc. 13th IEEE Symp. on Switching and Automata Theory*, S. 125–129. IEEE Computer Society, 1972.

[Pin86] J.-É. Pin. *Varieties of Formal Languages.* North Oxford Academic, London, 1986.

[Ram29] F. D. Ramsey. On a problem of formal logic. *Proc. of the London Math. Soc.*, 30:338–384, 1929.

[Sch56] M. P. Schützenberger. Une théorie algébrique du codage. *Séminaire Dubreil. Algèbre et théorie des nombres*, 9:1–24, 1955-1956.

[SVW87] A. P. Sistla, M. Y. Vardi, P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoret. Comput. Sci.*, 49(2-3):217–237, 1987. Twelfth international colloquium on automata, languages and programming (Nafplion, 1985).

[Tar72] R. E. Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[TFVT11] M.-H. Tsai, S. Fogarty, M. Y. Vardi, Y.-K. Tsay. State of Büchi Complementation. In *Proceedings of the 15th International Conference on Implementation and Application of Automata*, CIAA'10, S. 261–271. Springer-Verlag, Berlin, Heidelberg, 2011.

[Tho97] W. Thomas. Languages, Automata and Logic. In A. Salomaa, G. Rozenberg, Herausgeber, *Handbook of Formal Languages*, Band 3, Beyond Words, S. 389–455. Springer, Berlin, 1997.

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