Institut für Formale Methoden der Informatik
Abteilung Theoretische Informatik
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3238

Normalformenberechnung in Graph-Gruppen und Coxeter-Gruppen

Jonathan Kausch

Studiengang:	Informatik
Prüfer:	Prof. Dr. V. Diekert
Betreuer:	Prof. Dr. V. Diekert
begonnen am:	23. August 2011
beendet am:	8. Dezember 2011
CR-Klassifikation:	F.2.2, F.2.4, G.2.1

Inhaltsverzeichnis

1 Einleitung **3**

2 Grundlagen **5**
2.1 Ersetzungssysteme 5
2.2 Die freie Gruppe 7
2.3 Coxeter-Gruppen 7
2.3.1 Rechtwinklige Coxeter-Gruppen 7
2.3.2 Allgemeine Coxeter-Gruppen 8
2.4 Graph-Gruppen 10
2.4.1 Einbettung in Coxeter-Gruppen 11
2.5 LogSpace Berechnungen 11

3 Präperfekte Gruppen **13**
3.1 Coxeter-Gruppen sind präperfekt 13

4 Automatische Gruppen **15**
4.1 Rechtwinklige Coxeter-Gruppen sind shortLex automatisch 15

5 Berechnung einer Normalform **19**
5.1 In der freien Gruppe in logarithmischem Platz 19
5.2 In Graph-Gruppen und rechtwinkligen Coxeter-Gruppen 19
5.2.1 In Linearzeit 19
5.2.2 In logarithmischem Platz 22
5.3 In Coxeter-Gruppen 26
5.3.1 Berechnungskomplexität 27
5.3.2 Das Alphabet einer Geodätischen 28

6 Zusammenfassung und Ausblick **29**

1 Einleitung

Graph-Gruppen sind von besonderem Interesse in der Informatik für die Untersuchung von nebenläufigen Prozessen. Sie sind eine natürliche Erweiterung von Spurmonoiden, welche von Mazurkiewicz und Keller eingeführt wurden [12, 9]. In der Mathematik sind sie auch als rechtwinklige Artin-Gruppen bekannt [3]. Interessante Problemstellungen sind unter anderem das Konjugiertheitsproblem, das Isomorphieproblem, das Wortproblem und das Normalformenberechnungsproblem. Behandelt wird in dieser Arbeit das Normalformenberechnungsproblem, welches schwieriger ist als das Wortproblem. Allerdings ist der Anwendungsbereich des Normalformenproblems größer. Eine Normalform ist eine eindeutige Darstellung eines Elements der Gruppe. Unter dem Wortproblem versteht man die Frage, ob ein Wort das neutrale Element der Gruppe darstellt. Das Wortproblem ist für jede endlich erzeugte lineare Gruppe in logarithmischem Platz entscheidbar [10]. Ob dies auch für die Berechnung von Normalformen gilt ist noch offen. Ein Linearzeit Algorithmus zur Berechnung einer Normalform wird in [5] für Spurmonoide eingeführt. Eine weitere Klasse von Gruppen ist die Klasse der automatischen Gruppen, welche mit gewissen Automaten ausgestattet sind. Für automatische Gruppen ist das Wortproblem in quadratischer Zeit entscheidbar [7]. Ist die Gruppe darüber hinaus shortLex automatisch, so lässt sich auch eine längenlexikographische Normalform in quadratischer Zeit berechnen. Coxeter-Gruppen sind shortLex automatisch [4], und damit insbesondere auch Graph-Gruppen, denn sie lassen sich in natürlicher Weise in rechtwinklige Coxeter-Gruppen einbetten (siehe Abschnitt 2.4).

Das Ziel dieser Arbeit ist ein Algorithmus, der in logarithmischem Platz eine längenlexikographische Normalform in Graph-Gruppen bzw. rechtwinkligen Coxeter-Gruppen berechnet. Dazu wird zunächst das Alphabet der Normalform in logarithmischem Platz bestimmt. Eine Einbettung in die allgemeine lineare Gruppe liefert ein Kriterium für das Alphabet der Normalform. Mit dem Kriterium kann dann eine Geodätische in rechtwinkligen Coxeter-Gruppen berechnet werden und daraus eine längenlexikographische Normalform. Ob sich das Resultat zur Berechnung einer längenlexikographischen Normalform in logarithmischem Platz auf allgemeine Coxeter-Gruppen übertragen lässt, ist noch offen. In [1] wird ein Algorithmus zur Berechnung einer längenlexikographischen Normalform vorgestellt, der mit linear vielen arithmetischen Operationen auskommt. Allerdings lässt sich die Berechnung nicht mit ganzen Zahlen durchführen, daher wird untersucht, wie viele Bits zur Repräsentation notwendig sind. Darüber hinaus wird gezeigt, dass Coxeter-Gruppen präperfekt sind, das heißt ein kürzester Repräsentant kann mit nur längenerhaltenden und längenverkürzenden Regeln berechnet werden. Außerdem wird ein elementarer Beweis angegeben, dass sogenannte rechtwinklige Coxeter-Gruppen automatisch sind.

Der Aufbau dieser Arbeit wird im Folgenden beschrieben. Er gliedert sich in präper-

fekte Gruppen, automatische Gruppen und der Berechnung von Normalformen.

In Kapitel 2 werden die grundlegenden Definitionen eingeführt. Für die Berechnungen in den Gruppen werden Ersetzungssysteme verwendet. Die untersuchten Gruppen in dieser Arbeit sind Graph-Gruppen und Coxeter-Gruppen. Als Grundlage für die Graph-Gruppe dient die freie Gruppe. Abschließend werden in Kapitel 2 einige Eigenschaften und Werkzeuge zu Berechnungen in logarithmischem Platz gezeigt.

In Kapitel 3 werden präperfekte Gruppen eingeführt und gezeigt, dass für Coxeter-Gruppen ein präperfektes Ersetzungssystem existiert, mit dem sich Geodätische ableiten lassen.

In Kapitel 4 werden automatische Gruppen vorgestellt und es wird ein elementarer Beweis gegeben, dass rechtwinklige Coxeter-Gruppen shortLex automatisch sind.

Kapitel 5 widmet sich der Berechnung von Normalformen. Zunächst wird gezeigt, dass sich eine Normalform in der freien Gruppe in logarithmischem Platz berechnen lässt. Dann wird ein Linearzeit Algorithmus zur Berechnung einer längenlexikographischen Normalform in Graph-Gruppen und rechtwinkligen Coxeter-Gruppen vorgestellt. Anschließend wird der Algorithmus zur Berechnung der längenlexikographischen Normalform in logarithmischem Platz vorgestellt und dessen Korrektheit bewiesen. Der letzte Abschnitt des Kapitels widmet sich allgemeinen Coxeter-Gruppen. Es wird zuerst untersucht, wie viele Bits zur Repräsentation für den in [1] vorgestellten Algorithmus notwendig sind. Dann wird gezeigt, wie sich das Alphabet der längenlexikographischen Normalform in allgemeinen Coxeter-Gruppen berechnen lässt.

In Kapitel 6 wird das Ergebnis der Arbeit zusammengefasst und ein kurzer Ausblick gegeben.

2 Grundlagen

Für die Berechnung von Normalformen eignen sich Ersetzungssysteme. In diesem Kapitel werden die untersuchten Gruppen eingeführt und geeignete Ersetzungssysteme vorgestellt. Die Elemente der Gruppen werden als Wörter über einem endlichen Alphabet Σ betrachtet. Das neutrale Element wird mit 1 bezeichnet. Im Abschnitt Ersetzungssysteme wird ein System eingeführt, das festlegt, welche Wörter die gleiche Bedeutung haben. Für die Berechnung von Normalformen wird stets die längenlexikographische Normalform, also ein minimaler Repräsentant eines Elements einer Gruppe, verwendet. Dafür ist insbesondere der Begriff der Länge eines Repräsentanten von Bedeutung.

Definition 2.1: *Sei $w = a_1 \dots a_n \in \Sigma^*$. Dann bezeichnet $|w| := n$ die Länge von w.*

Der Vektorraum $\mathbb{R}^\Sigma$ wird mit $\mathbb{R}^k$ identifiziert, wobei $k := |\Sigma|$. Die Menge $\{ a \mid a \in \Sigma \}$ bildet eine Basis von $\mathbb{R}^\Sigma$. Mit LogSpace wird die Klasse der Funktionen beschrieben, die sich auf einer Turingmaschine mit Platz $\mathcal{O}(\log n)$ berechnen lassen, wobei n die Eingabegröße ist. Im Rahmen dieser Arbeit ist die Eingabegröße die Länge $|w|$ der Eingabe w.

2.1 Ersetzungssysteme

Ein Ersetzungssystem ist eine Menge S von Ersetzungsregeln der Form $l \to r$. Kommt die linke Seite l in einem Wort w vor, so darf sie durch die rechte Seite r ersetzt werden. Zwei Wörter gelten als äquivalent, falls sie sich durch Anwendung von Regeln aus dem reflexiven, symmetrischen und transitiven Abschluss von S ineinander überführen lassen.

Definition 2.2: *Sei $S \subseteq \Sigma^* \times \Sigma^*$ eine endliche Menge. Schreibe $l \to r$ für $(l, r) \in S$. Seien $w, w' \in \Sigma^*$. S definiert durch*

$$w \Longrightarrow_S w', \text{ falls } w = ulv, w' = urv \text{ und } l \to r$$

ein Ersetzungssystem auf Σ^*.

Mit $\overset{*}{\Longrightarrow}_S$ wird der reflexive und transitive Abschluss von $\Longrightarrow_S$ bezeichnet. Die Relation $\overset{*}{\Longleftrightarrow}_S$ bezeichnet den reflexiven, transitiven und symmetrischen Abschluss. Der Index S wird weg gelassen, wenn das verwendet Ersetzungssystem aus dem Kontext klar ist. Damit lässt sich über den Quotienten eine neue Struktur definieren.

Definition 2.3: *Der Quotient*

$$\Sigma^*/S := \Sigma^*/ \overset{*}{\Longleftrightarrow}_S = \{ [w]_S \mid w \in \Sigma^* \}$$

bezeichnet die Menge der Äquivalenzklassen $[w]_S$ *von* $\overset{*}{\Longleftrightarrow}_S$.

Die Relationen aus S werden auch als definierende Relationen von Σ^*/S bezeichnet. Für manche Ersetzungssysteme spielt die Reihenfolge, in der Regeln angewendet werden, eine Rolle. Ist dies nicht der Fall, so wird das Ersetzungssystem als konfluent bezeichnet.

Definition 2.4: *Ein Ersetzungssystem S heißt konfluent, falls für $w' \overset{*}{\Longleftarrow} w \overset{*}{\Longrightarrow} w''$ immer ein v mit Ableitungen $w' \overset{*}{\Longrightarrow} v \overset{*}{\Longleftarrow} w''$ existiert. Das Ersetzungssystem heißt stark konfluent, falls für $w' \Longleftarrow w \Longrightarrow w''$ immer ein v und Ableitungen $w' \overset{\leq 1}{\Longrightarrow} v \overset{\leq 1}{\Longleftarrow} w''$ existieren.*

Lemma 2.1: *Ein stark konfluentes Ersetzungssystem ist konfluent.*

Beweis. Mittels Induktion lässt sich zeigen, dass für $y \overset{\leq n}{\Longleftarrow} x \overset{\leq m}{\Longrightarrow} z$ ein v existiert mit $y \overset{\leq m}{\Longrightarrow} v \overset{\leq n}{\Longleftarrow} z$. Daraus folgt dann die Konfluenz. Für $n, m \leq 1$ gilt die Behauptung wegen der starken Konfluenz. Sei die Behauptung also für $n, m \geq 1$ gezeigt und $y \overset{\leq n+1}{\Longleftarrow} x \overset{\leq m+1}{\Longrightarrow} z$. Dann existiert auf dem Pfad zu y ein y' und zu z ein z' mit $y \Longleftarrow y' \overset{\leq n}{\Longleftarrow} x \overset{\leq m}{\Longrightarrow} z' \Longrightarrow z$. Nach Induktion existiert ein v' mit $y' \overset{\leq m}{\Longrightarrow} v' \overset{\leq n}{\Longleftarrow} z'$. Daraus lässt sich ebenfalls nach Induktion v_L und v_R ableiten mit $y \overset{\leq m}{\Longrightarrow} v_L \overset{\leq 1}{\Longleftarrow} v'$ und $v' \overset{\leq 1}{\Longrightarrow} v_R \overset{\leq n}{\Longleftarrow} z$. Wegen der starken Konfluenz existiert ein v mit $v_L \overset{\leq 1}{\Longrightarrow} v \overset{\leq 1}{\Longleftarrow} v_R$. Insgesamt ist $y \overset{\leq m+1}{\Longrightarrow} v \overset{\leq n+1}{\Longleftarrow} z$ und die Behauptung ist gezeigt. □

Bemerkung 2.1: Die Voraussetzung eines stark konfluenten Ersetzungssystems in Lemma 2.1 kann abgeschwächt werden durch: Es existiert ein $k \in \mathbb{N}$ so, dass für $y \Longleftarrow x \Longrightarrow z$ ein v existiert mit $y \overset{\leq k}{\Longrightarrow} v \overset{\leq k}{\Longleftarrow} z$.

Aufbauend auf dem Ersetzungssystem lässt sich die minimale Länge eines Repräsentanten und damit die längenlexikographische Normalform einführen.

Definition 2.5: *Die Länge eines kürzesten Repräsentanten ist durch*

$$\|w\| = \min \{\, |u| \mid u \in [w]_S \,\}$$

definiert. Es gilt stets $\|w\| \leq |w|$.

Definition 2.6: *Sei $\prec$ eine totale Ordnung auf dem Alphabet Σ. Dann induziert dies eine längenlexikographische Ordnung über Σ^*, welche wieder mit $\prec$ bezeichnet wird.*

$v \prec w$, falls $|v| < |w|$ und
$v \prec w$, falls $|v| = |w|$ und ein i existiert so, dass $v_j = w_j$ für $j < i$ und $v_i \prec w_i$

Definition 2.7:

1. *Ein Wort w heißt geodätisch, falls $|w| = \|w\|$.*

2. *Ein Wort w heißt shortLex, falls* $w = \min\left\{ u \in \Sigma^* \;\middle|\; u \overset{*}{\Longleftrightarrow} w \right\}$.

Die längenlexikographische Normalform von w wird mit $nf(w)$ bezeichnet. Ein Wort w ist in längenlexikographischer Normalform, falls es shortLex ist. Ein Wort v ist eine Geodätische von w, falls v äquivalent zu w ist und $|v| = \|w\|$.

2.2 Die freie Gruppe

Definition 2.8: *Sei $\overline{\Sigma}$ eine disjunkte Kopie von Σ , dann bezeichnet*

$$F(\Sigma) := (\Sigma \cup \overline{\Sigma})^* / \{\, a\overline{a} \to 1, \overline{a}a \to 1 \;\mid\; a \in \Sigma \,\}$$

die freie Gruppe über Σ. $F(\Sigma)$ wird mit einer Involution $\overline{a}$ ausgestattet so, dass $\overline{\overline{a}} = a$.

Bemerkung 2.2: Für die Involution gilt insbesondere $\overline{ab} = \overline{b}\overline{a}$.

Die einfachste, nicht triviale freie Gruppe ist $\mathbb{Z} \cong F(\{a\})$. Ein Isomorphismus zwischen $\mathbb{Z}$ und $F(\{a\})$ ist beispielsweise $\varphi(k) = a^k$ und $\varphi(-k) = \overline{a}^k$ für $k \geq 0$.

Lemma 2.2: *Das Ersetzungssystem $\{\, a\overline{a} \to 1 \;\mid\; a \in \Sigma \,\}$ der freien Gruppe $F(\Sigma)$ ist konfluent.*

Beweis. Die kritischen Fälle für die Konfluenz sind Überlappungen. Der einzige mögliche Fall für das Ersetzungssystem der freien Gruppe ist $a\overline{a}a$ (bzw. $\overline{a}a\overline{a}$). Es kann $a\overline{a} \to 1$ oder $\overline{a}a \to 1$ angewendet werden. In beiden Fällen ist das Ergebnis der Ersetzung jedoch a (bzw. $\overline{a}$). Somit ist das Ersetzungssystem stark konfluent und nach Lemma 2.1 konfluent. □

2.3 Coxeter-Gruppen

Die in Kapitel 5 vorgestellten Methoden für die Berechnung von Normalformen verwenden Coxeter-Gruppen als Grundlage. Die Resultate für rechtwinklige Coxeter-Gruppen können auf Graph-Gruppen übertragen werden, da sich Graph-Gruppen in rechtwinklige Coxeter-Gruppen einbetten lassen (siehe 2.4). In diesem Abschnitt werden zunächst rechtwinklige Coxeter-Gruppen und einige ihrer Eigenschaften vorgestellt, da insbesondere Theorem 5.5 auf rechtwinkligen Coxeter-Gruppen basiert. Dann werden allgemeine Coxeter-Gruppen vorgestellt. Eine ausführliche Einführung zu Coxeter-Gruppen findet sich in [1, 2].

2.3.1 Rechtwinklige Coxeter-Gruppen

Definition 2.9: *Sei Σ ein endliches Alphabet und $I \subseteq \Sigma \times \Sigma$ eine irreflexive und symmetrische Relation, dann bezeichnet*

$$C(\Sigma, I) := \Sigma^* / \left\{\, a^2 \to 1, (bc)^2 \to 1 \;\middle|\; a \in \Sigma, (b,c) \in I \,\right\}$$

die rechtwinklige Coxeter-Gruppe zu (Σ, I).

Der Bezeichner der Relation I steht für Independent, da die zweite Bedingung $(bc)^2 = 1$ äquivalent zu $bc = cb$ ist. Mit $D := \Sigma \times \Sigma \setminus I$ werden die abhängigen Erzeugenden bezeichnet. Für die Berechnungen in logarithmischem Platz ist die Einbettung der Gruppe in die allgemeine lineare Gruppe $\mathrm{GL}_k(\mathbb{C})$, $k := |\Sigma|$, von zentraler Bedeutung. Hierbei ist eine Einbettung ein injektiver Homomorphismus.

Definition 2.10: *Sei $a \in \Sigma$ und $w = a_1 \dots a_n$. Die Abbildung $\sigma_a : \mathbb{Z}^\Sigma \to \mathbb{Z}^\Sigma$ ist durch*

$$\sigma_a(b) := \begin{cases} -a, & \text{falls } b = a \\ b, & \text{falls } (a,b) \in I \\ b + 2a, & \text{sonst} \end{cases}$$

definiert. Damit ist die Abbildung $\sigma : w \mapsto \sigma_w$ durch $\sigma_w := \sigma_{a_1} \dots \sigma_{a_n}$ ein Homomorphismus von der rechtwinkligen Coxeter-Gruppe $C(\Sigma, I)$ in die allgemeine lineare Gruppe $GL_k(\mathbb{Z})$.

Lemma 2.3: *Die Abbildung σ definiert eine Einbettung von der rechtwinkligen Coxeter-Gruppe $C(\Sigma, I)$ in die allgemeine lineare Gruppe $GL_k(\mathbb{Z})$, wobei $k := |\Sigma|$.*

Beweis. Es ist zunächst zu zeigen, dass σ wohldefiniert ist. Das bedeutet für $u = w$ in $C(\Sigma, I)$ muss $\sigma(u) = \sigma(v)$ gelten. Hierfür ist zu zeigen, dass die definierenden Relationen der rechtwinkligen Coxeter Gruppe auch im Bild von σ gelten. Eine kurze Rechnung zeigt, dass $\sigma_{aa} = \mathrm{id}$ für $a \in \Sigma$ und $\sigma_{(bc)^2} = \mathrm{id}$ für $(b,c) \in I$. Die Injektivität der Abbildung wird im Beweis zu Lemma 2.6 gezeigt. □

Ein alternatives Ersetzungssystem für rechtwinklige Coxeter-Gruppen wird durch

$$S_R := \left\{ a^2 \to 1, bc \to cb \;\middle|\; a \in \Sigma, (b,c) \in I \right\}$$

erzeugt. In Kapitel 3 wird gezeigt, dass dieses Ersetzungssystem ausreicht, um eine Geodätische zu berechnen und um Geodätische ineinander zu transformieren.

2.3.2 Allgemeine Coxeter-Gruppen

Definition 2.11: *Sei $M = (m_{i,j}) \in \mathbb{N}^{k \times k}$ eine symmetrische Matrix und $m_{i,j} = 1 \Leftrightarrow i = j$. Die definierenden Relationen der Coxeter-Gruppe zu M sind $(x_i x_j)^{m_{i,j}} = 1$.*

$$C(\Sigma, M) := \Sigma^* / \left\{ (x_i x_j)^{m_{i,j}} \to 1 \;\middle|\; x_i, x_j \in \Sigma \right\}$$

Wegen $m_{i,i} = 1$ ist $x_i^2 = (x_i x_i)^1 = 1$. Rechtwinklige Coxeter-Gruppen sind offensichtlich Coxeter-Gruppen der Matrix $M = m_{i,j}$ mit $m_{i,i} = 1$ und $m_{i,j} = 2$, falls $(x_i, x_j) \in I$. Wie bereits für rechtwinklige Coxeter-Gruppen lässt sich auch für allgemeine Coxeter-Gruppen eine Einbettung in die allgemeine lineare Gruppe $\mathrm{GL}_k(\mathbb{R})$ definieren.

Definition 2.12 (Lineare Einbettung)**:** *Sei $\Sigma = \{x_1, \dots, x_k\}$ und $\sigma_a : \mathbb{R}^\Sigma \to \mathbb{R}^\Sigma$ mit*

$$\sigma_{x_i}(x_j) = \begin{cases} x_j + 2\cos(\frac{\pi}{m_{i,j}}) \cdot x_i & \text{falls } m_{i,j} \neq 0 \\ x_j + 2 \cdot x_i & \text{falls } m_{i,j} = 0 \end{cases}$$

Weiter sei $w = a_1 \dots a_n \in \Sigma^*$, dann wird durch $\sigma : C(\Sigma, M) \to GL_k(\mathbb{R}) : w \mapsto \sigma_w$ mit

$$\sigma(w) = \sigma_w := \sigma_{a_1} \dots \sigma_{a_n} \tag{2.1}$$

ein Homomorphismus der Coxeter-Gruppe in die allgemeine lineare Gruppe $GL_k(\mathbb{R})$ definiert.

Lemma 2.4: *Die Ordnung von ab in der Coxeter-Gruppe und σ_{ab} stimmt für $a, b \in \Sigma$ überein.*

Beweis. siehe [1, Kapitel 4, Proposition 4.2.1]. □

Eine besondere Eigenschaft dieses Homomorphismus ist, dass sich Aussagen über die minimale Länge eines Repräsentanten treffen lassen:

Definition 2.13: *Sei $v = \sum \lambda_a a \in \mathbb{R}^\Sigma$. Schreibe $v \leq 0$, falls $\lambda_a \leq 0$ für alle $a \in \Sigma$ und $v \geq 0$, falls $\lambda_a \geq 0$ für alle $a \in \Sigma$.*

Lemma 2.5: *Sei $a \in \Sigma$ und $w \in \Sigma^*$. Für die Länge einer Geodätischen von wa in der Coxeter-Gruppe gilt*

1. $\|wa\| > \|w\| \Leftrightarrow \sigma_w(a) \geq 0$
2. $\|wa\| < \|w\| \Leftrightarrow \sigma_w(a) \leq 0$

Beweis. siehe [1, Kapitel 4, Proposition 4.2.5]. □

Lemma 2.6: *Die Abbildung σ ist eine Einbettung der Coxeter-Gruppe $C(\Sigma, M)$ in die allgemeine lineare Gruppe $GL_k(\mathbb{R})$.*

Beweis. Es ist zunächst zu zeigen, dass σ wohldefiniert ist. Das bedeutet für $u = w$ in $C(\Sigma, M)$ muss $\sigma(u) = \sigma(v)$ gelten. Hierfür ist zu zeigen, dass die definierenden Relationen der Coxeter Gruppe auch im Bild von σ gelten. Nach Lemma 2.4 sind diese erfüllt. Es verbleibt zu zeigen, dass σ injektiv ist. Dies ist genau dann der Fall, wenn der Kern $\ker(\sigma)$ von σ nur das neutrale Element enthält. Sei dazu $w \neq 1$ in der Coxeter Gruppe. w lässt sich schreiben als $w = ua$ für ein $u \in \Sigma^*$ und $a \in \Sigma$. Dann ist nach Lemma 2.5 $\sigma_u(a) \leq 0$. Für die Identität gilt aber $\mathrm{id}(a) = a > 0$, also kann der Kern nur das neutrale Element enthalten. □

Ein alternatives Ersetzungssystem lässt sich für allgemeine Coxeter-Gruppen analog zu dem rechtwinkliger Coxeter-Gruppen angeben.

Definition 2.14: *Betrachte das Wort $(ab)^l \in \Sigma^*$, dann bezeichnet $p(l, a, b)$ den Präfix der Länge l von $(ab)^l$. Analog bezeichnet $s(l, a, b)$ den Suffix der Länge l von $(ab)^l$.*

Das alternative Ersetzungssystem wird für $\Sigma = \{x_1, \dots, x_k\}$ durch

$$S_C := \left\{ x_i^2 \to 1, s(x_i, x_j, m_{i,j}) \to s(x_j, x_i, m_{i,j}) \;\middle|\; x_i, x_j \in \Sigma \right\}$$

erzeugt. Die Regeln $s(x_i, x_j, m_{i,j}) \to s(x_j, x_i, m_{i,j})$ sind wegen M symmetrisch ebenfalls symmetrisch.

Lemma 2.7: *Das Ersetzungssystem S_C ist konfluent.*

Beweis. Die kritischen Paare sind Überlappung so, dass zwei Regeln angewendet werden können. Das sind entweder Überlappungen aaa mit der Regel $a^2 \to 1$ oder Überlappungen mit einer Regel

$$s(x_i, x_j, m_{i,j}) \to s(x_j, x_i, m_{i,j}).$$

Im Fall aaa ist das Ergebnis jeweils a. Das Resultat der Anwendung einer Regel

$$s(x_i, x_j, m_{i,j}) \to s(x_j, x_i, m_{i,j})$$

lässt sich immer rückgängig machen, da die Regel symmetrisch ist. Eine Ableitung auf ein gemeinsames Element ist also in höchstens zwei Schritten möglich. Nach Bemerkung 2.1 ist das Ersetzungssystem somit konfluent. □

2.4 Graph-Gruppen

Graph-Gruppen sind eine natürliche Erweiterung von Spurmonoiden, welche von Mazurkiewicz und Keller eingeführt wurden [12, 9]. Darüber hinaus spielen sie eine wichtige Rolle für die Informatik bei der Untersuchung von Nebenläufigkeiten.

Definition 2.15: *Sei $I \subseteq \Sigma \times \Sigma$ eine irreflexive und symmetrische Relation. Dann bezeichnet*

$$M(\Sigma, I) := \Sigma^* / \{\, ab \to ba \mid (a, b) \in I \,\}$$

das Spurmonoid (oder frei partiell kommutative Monoid) über (Σ, I) und

$$G(\Sigma, I) := F(\Sigma) / \{\, ab \to ba \mid (a, b) \in I \,\}$$

die Graph-Gruppe (oder frei partiell kommutative Gruppe) über (Σ, I). Die Elemente des Spurmonoids werden auch als Spur bezeichnet.

I wird Unabhängigkeitsrelation und $D := \Sigma \times \Sigma \setminus I$ wird Abhängigkeitsrelation genannt. Mit $I(a) := \{\, b \mid (a, b) \in I \,\}$ werden die von $a \in \Sigma$ unabhängigen Elemente bezeichnet.

Bemerkung 2.3: Die freie Gruppe $F(\Sigma)$ ist eine Graph-Gruppe, wobei die Unabhängigkeitsrelation I die leere Menge ist, also $F(\Sigma) = G(\Sigma, \emptyset)$. Die Abhängigkeitsrelation D von $F(\Sigma)$ ist also $\Sigma \times \Sigma$.

Eine Spur $w \in M(\Sigma, I)$ lässt sich eindeutig als Abhängigkeitsgraph $\mathrm{DG}(w)$ darstellen.

Definition 2.16: *Sei $w = a_1 \dots a_n \in \Sigma^*$. Der Abhängigkeitsgraph $DG(w)$ ist ein gerichteter Graph mit Knotenmenge $\{1, \dots, n\}$. Von Knoten i zu Knoten j existiert genau dann eine Kante, wenn $i < j$ und $(a_i, a_j) \in D$. Die Beschriftung von Knoten i ist a_i und wird mit $\lambda(i)$ bezeichnet.*

Der transitive Abschluss von $\mathrm{DG}(w)$ definiert eine partielle Ordnung, da der Graph azyklisch ist. Diese partielle Ordnung wird eindeutig durch ihr Hasse-Diagramm dargestellt.

Definition 2.17: *Sei $w = a_1 \dots a_n \in \Sigma^*$. Das Hasse-Diagramm ist ein gerichteter Graph mit Knotenmenge $\{1, \dots, n\}$. Von Knoten i zu Knoten j existiert genau dann eine Kante, wenn $i < j$ und $(a_i, a_j) \in D$, sowie kein k mit $i < k < j$ und $(a_i, a_k), (a_k, a_j) \in D$ existiert .*

2.4.1 Einbettung in Coxeter-Gruppen

Graph-Gruppen lassen sich leicht in rechtwinklige Coxeter-Gruppen einbetten, wenn man bedenkt, dass $ab = ba \Leftrightarrow (ab)^2 = 1$. Ein wichtiger Bestandteil rechtwinkliger Coxeter-Gruppen ist die Bedingung $a^2 = 1$, welche in Graph-Gruppen nicht erfüllt ist. Dafür werden für jedes Erzeugende a der Graph-Gruppe zwei Erzeugende a', a'' mit $a'^2 = a''^2 = 1$ in der rechtwinkligen Coxeter-Gruppe eingeführt. Mittels der Zuordnung $a \mapsto a'a''$ lässt sich eine Einbettung definieren. Das folgende Lemma formalisiert die Beschreibung.

Lemma 2.8: *Sei $G(\Sigma, I)$ eine Graph-Gruppe und Σ', Σ'' disjunkte Kopien von Σ. Die Abbildung φ definiert durch $\varphi(a) = a'a''$ und $\varphi(\overline{a}) = a''a'$ eine Einbettung in die rechtwinklige Coxeter-Gruppe $C(\Sigma' \cup \Sigma'', I_C)$, wobei*

$$I_C := \left\{ (a', b'), (a', b''), (a'', b'), (a'', b'') \;\middle|\; (a, b) \in I \right\}$$

Beweis. Die Abbildung φ ist wohldefiniert, denn die definierenden Relationen sind im Bild $\varphi(\Sigma)$ erfüllt. Für die Injektivität von φ wird eine Umkehrabbildung konstruiert. Jedes Element aus dem Bild von φ lässt sich als Element von $\{ a'a'', a''a' \mid a \in \Sigma \}^*$ schreiben. Daher lässt sich unmittelbar eine Umkehrabbildung ψ durch $\psi(a'a'') = a$ und $\psi(a''a') = \overline{a}$ angeben. Also ist φ injektiv und somit eine Einbettung. □

2.5 LogSpace Berechnungen

Ein zentraler Bestandteil dieser Arbeit sind Berechnungen in logarithmischem Platz. Dieser Abschnitt führt dafür einige notwendige Lemmata ein.

Definition 2.18: *Für das Produkt aller Primzahlen bis zu einer Zahl n setze*

$$\mu(n) := \prod_{p \in \mathbb{P}, p \leq n} p$$

Lemma 2.9: *Es gibt eine Konstante $k \in \mathbb{N}$ so, dass für das Produkt über alle Primzahlen kleiner n gilt $\mu(kn) > 2^n$.*

Beweis. Nach Hardy und Wright [8, S. 341] gibt es eine Konstante $A > 0$ so, dass $\log \mu(x) > Ax$ für alle x gilt. Falls $A \geq 1$, so kann $k = 1$ gewählt werden und die Behauptung folgt. Sei also $0 < A < 1$. Wähle $k := \lceil \frac{1}{A} \rceil$, dann ist $\mu(kn) > 2^{A(kn)} = 2^{(Ak)n} > 2^n$. □

Lemma 2.10: *Sei $n \in \mathbb{N}$ und $x \leq 2^n$. Dann ist $x = 0$ genau dann, wenn $x = 0 \mod p$ für alle Primzahlen $p \leq kn$.*

Beweis. Nach dem vorherigen Lemma 2.9 gilt $x < \mu(kn)$. Sei $x = 0 \mod p$ für alle Primzahlen $p \leq kn$. Dann folgt $x = 0$ aus dem Chinesischen Restsatz. Ist umgekehrt $x = 0$, dann gilt unmittelbar $x = 0 \mod p$. □

Lemma 2.11: *Sei $C(\Sigma, I)$ eine rechtwinklige Coxeter-Gruppe. Dann ist die Überprüfung der Koeffizienten von $\sigma_w(x)$, ob sie ungleich null sind in LogSpace möglich.*

Beweis. Setze $n := |w|$. Die Koeffizienten von $\sigma_w(x)$ sind nach Konstruktion der Abbildung durch 2^n beschränkt. Daher genügt es nach Lemma 2.10 die Koeffizienten $\mod p$ zu speichern. Da die Primzahlen $p \leq kn$ nicht bekannt sind, werden die Koeffizienten $\mod l$ für $l \leq kn$ überprüft. □

Das folgende Lemma ist ein erster Anhaltspunkt für die Berechnung einer Normalform in LogSpace. Allerdings ist es noch nicht hinreichend.

Lemma 2.12: *Das Wortproblem ist für endlich erzeugte lineare Gruppen über einem Körper K der Charakteristik 0 in LogSpace lösbar.*

Beweis. siehe [10, Theorem 5]. □

Lemma 2.13: *Das Wortproblem für Coxeter-Gruppen liegt in LogSpace.*

Beweis. Die Behauptung folgt zusammen mit Lemma 2.12 aus der linearen Einbettung σ für Coxeter-Gruppen. □

Bemerkung 2.4: Aus den beiden vorherigen Lemma folgt insbesondere, dass das Wortproblem für Graph-Gruppen (und somit auch das der freien Gruppe) in LogSpace liegt.

Das folgende Lemma ist ein wichtiges Resultat über die Hintereinanderausführung von LogSpace-Transducern.

Lemma 2.14: *Seien L_1, L_2 zwei LogSpace-Transducer und $L_i(w)$ bezeichne die Ausgabe von L_i bei Eingabe $w \in \Sigma^*$, dann kann $L_2(L_1(w))$ in logarithmischem Platz berechnet werden.*

Beweis. Das Ergebnis der Ausgabe von L_1 muss nicht zwingend in logarithmischem Platz gespeichert werden können. Allerdings liest L_2 in einem Schritt nur ein Zeichen der Eingabe $L_1(w)$. Daher wird, wenn L_2 auf den i-ten Buchstaben von $L_1(w)$ zugreifen möchte, die Ausgabe $L_1(w)$ in logarithmischem Platz neu berechnet und nur das i-te Zeichen gespeichert. Die Berechnung von L_1 und L_2 ist also weiterhin in logarithmischem Platz möglich. □

3 Präperfekte Gruppen

Ein präperfektes Ersetzungssystem ist ein konfluentes Ersetzungssystem, welches mit längenerhaltenden und längenreduzierenden Regeln auskommt. Ziel dieses Abschnitts ist es zu zeigen, dass für Coxeter-Gruppen ein präperfektes Ersetzungssystem existiert. Das besondere ist, dass sich Geodätische ausschließlich mit diesen längenerhaltenden und längenverkürzenden Regeln berechnen und ineinander transformieren lassen. Weiter lassen sich mit diesem Resultat einige Eigenschaften von Coxeter-Gruppen leicht beweisen. Eine allgemeine Einführung in präperfekte Gruppen findet sich in [6].

Definition 3.1: *Sei $\mathcal{S}$ ein konfluentes Ersetzungssystem. $\mathcal{S}$ heißt präperfekt, falls*

1. *$l \to r \in \mathcal{S}$, dann gilt $|l| > |r|$ oder*
2. *$l \to r \in \mathcal{S}$ mit $|l| = |r|$, dann ist auch $r \to l \in \mathcal{S}$.*

Eine Gruppe ist präperfekt, falls für sie ein präperfektes Ersetzungssystem existiert.

3.1 Coxeter-Gruppen sind präperfekt

Als Ersetzungssystem wird das in Abschnitt 2.3 eingeführte Ersetzungssystem S_C verwendet. Die Konfluenz dieses Ersetzungssystems wurde bereits in Lemma 2.7 gezeigt.

Definition 3.2: *a ist maximal in $w \in C(\Sigma, I)$, falls $w = ua$ in der Coxeter-Gruppe mit $\|w\| = \|u\| + 1$.*

Lemma 3.1: *Ein erzeugendes Element $a \in \Sigma$ ist genau dann maximal in $w \in \Sigma^*$, falls $\sigma_w(a) \leq 0$.*

Beweis. Sei a maximal in w, dann lässt sich nach Definition w schreiben als $w = ua$, wobei $\|ua\| > \|u\|$. Demnach ist nach Konstruktion von σ_a

$$\sigma_w(a) = \sigma_{ua}(a) = \sigma_u(-a) = -\sigma_u(a).$$

Wegen Lemma 2.5 ist $\sigma_u(a) \geq 0$, also folgt insgesamt $\sigma_w(a) \leq 0$.

Umgekehrt sei $\sigma_w(a) \leq 0$, dann ist wegen Lemma 2.5 $\|wa\| > \|w\|$. Somit existiert ein $u := wa$ mit $\|u\| = \|w\| - 1$. Mit $w = waa = ua$ in der Coxeter-Gruppe ergibt sich die Maximalität von a in w. □

Definition 3.3: *Der Suffix der Länge l von $(ab)^l$ wird mit $s(l, a, b)$ bezeichnet.*

Satz 3.2: *Coxeter-Gruppen sind präperfekt:*

1. *Seien w, w' geodätisch und $w = w'$ in der Coxeter-Gruppe. Dann lassen sich w und w' durch Anwendung von Regeln aus S_C ineinander überführen.*

2. *Sei $w \in \Sigma^*$, dann lässt sich w durch Anwenden von Regeln aus S_C in eine reduzierte Form überführen.*

Beweis. Beweis von 1 mit Induktion nach $\|w\|$: Für $\|w\| = 0$ ist die Aussage klar. Sei die Aussage gültig für Wörter der Länge höchstens $\|w\|$. Sei nun $w = ua$ und $w' = u'b$. Ohne Einschränkung sei $a \neq b$, sonst folgt die Aussage mit Induktion auf u, u'. a und b sind maximale Elemente in w und nach Lemma 3.1 somit $\sigma_w(b) = \sigma_{ua}(b) \leq 0$. Nach Konstruktion von σ gilt damit $\sigma_u(b) \leq 0$, da wegen w reduziert $\sigma_u(a) \geq 0$ gilt. Wähle $l \geq 1$ maximal so, dass $u = vs(l, a, b)$ in der Coxeter-Gruppe und $|u| = |vs(l, a, b)|$. Mit Induktion lässt sich u mit Regeln aus S_C in $vs(l, a, b)$ transformieren, das heißt $u \overset{*}{\Longrightarrow}_{S_C} vs(l, a, b)$. Sei also ohne Einschränkung $u = vs(l, a, b)$. Insgesamt ergibt sich

$$w = ua = vs(l, a, b)a = vs(l + 1, b, a)$$

Wegen der Maximalität von l ist nach Lemma 2.5 $\sigma_v(b) \geq 0$ und $\sigma_v(a) \geq 0$. Da $\sigma_{ua}(b) \leq 0$ muss nach Konstruktion von σ also $\sigma_{s(l+1,b,a)}(b) \leq 0$ gelten. Daher muss, da u reduziert ist, $l + 1 = m_{a,b}$ gelten. Somit lässt sich b durch die Regel $s(l + 1, a, b) \to s(l + 1, b, a)$ aus S_C nach hinten drehen. Mit Induktion auf u, u' ergibt sich schließlich $w \overset{*}{\Longrightarrow}_{S_C} w'$.

Beweis von 2 mit Induktion nach $|w|$: Sei $w = a_1 \dots a_n \in \Sigma^*$ und nicht reduziert. Wähle i maximal so, dass $a_1 \dots a_{i-1}$ reduziert ist. Dann ist $\sigma_{a_1 \dots a_{i-1}}(a_i) \leq 0$. Nach dem vorherigen Lemma 3.1 lässt sich ist also $a_1 \dots a_{i-1}$ schreiben als ua_i. Die Transformation kann wegen 1 mit Regeln aus S_C erfolgen. Schließlich lässt sich $a_1 \dots a_n$ unter Anwendung der Regel $a_i a_i \to 1$ transformieren in $\widetilde{w} := ua_{i+1} \dots a_n$. Mit Induktion lässt sich $\widetilde{w}$ mit Regeln aus S_C in eine Geodätische überführen. □

4 Automatische Gruppen

In diesem Abschnitt geht es um eine automatische Struktur für Gruppen. Das bedeutet, dass durch einen Automaten, also in Linearzeit, entschieden werden kann, ob ein Wort eine gewisse Form erfüllt (beispielsweise eine Normalform) und, ob ein Wort v aus w durch Multiplikation mit $a \in \Sigma$ entstanden ist. Für alle automatischen Gruppen ist das Wortproblem in quadratischer Zeit entscheidbar [7].

Definition 4.1: *Eine Gruppe G heißt* automatisch*, falls sie mit einem Automat $\mathcal{W}$ und Automaten $\mathcal{W}_a$, $a \in \Sigma \cup \{1\}$, ausgestattet ist, die folgende Eigenschaften erfüllen:*

1. *$\mathcal{W}$ akzeptiert mindestens einen Repräsentanten von $g \in G$.*
2. *Falls w, v von $\mathcal{W}$ akzeptiert werden, dann akzeptiert $\mathcal{W}_a$ das Paar (w, v) genau dann, falls $w = va$ in der Gruppe G.*

Akzeptiert $\mathcal{W}$ genau die längenlexikographischen Normalformen von G, so wird G als shortLex automatisch bezeichnet.

I bezeichnet wieder die unabhängigen Erzeugenden, also $(a, b) \in I \Rightarrow ab = ba$. Für den Beweis, dass rechtwinklige Coxeter-Gruppen ShortLex automatisch sind wird folgendes Lemma benötigt:

Lemma 4.1 (Deletion Property)**:** *Sei $C(\Sigma, I)$ eine rechtwinklige Coxeter-Gruppe und $w = a_1 \dots a_n$ geodätisch mit $\|wa\| < \|w\|$. Dann existiert $1 \leq i \leq n$ so, dass*

$$wa = a_1 \dots a_{i-1} a_{i+1} \dots a_n$$

Beweis. Im Ersetzungsprozess löscht sich a mit einem a_i. a_i ist wegen Lemma 3.1 maximal in w. Aus Theorem 3.2 folgt nun, dass a_i durch Kommutieren an das Ende von w gedreht werden kann. Angenommen es existiert ein $j > i$ mit $a_j \notin I(a)$, dann muss es ein a_k geben, welches sich mit a_j löscht. Nach Voraussetzung ist w aber geodätisch, also muss $a_j \in I(a)$ für alle $j > i$ gelten. Da $a_j \in I(a)$ für alle $j > i$ lässt sich a_i löschen ohne die Reihenfolge der a_j für $j \neq i$ zu verändern. □

Bemerkung 4.1: Das Lemma gilt für beliebige Coxeter-Gruppen. Für einen allgemeineren Beweis siehe [1, Kapitel 1, Proposition 1.4.7].

4.1 Rechtwinklige Coxeter-Gruppen sind shortLex automatisch

Für rechtwinklige Coxeter-Gruppen lässt sich ein elementarer Beweis angeben, dass sie shortLex automatisch sind. Die Kernidee ist die maximalen Elemente in der Zustandskodierung zu speichern. Damit lässt sich sicherstellen, dass das zu verifizierende Wort

geodätisch ist. Damit lediglich die längenlexikographische Normalform akzeptiert wird, werden in den Zuständen auch die maximalen Elemente der lexikographisch kleineren Wörter gespeichert.

Die Sprache $L = \{ \mathrm{nf}(x) \mid x \in \Sigma^* \}$ ist die Menge der längenlexikographischen Normalformen von Σ^*. Im Folgenden wird ein Automat konstruiert, der die Sprache L erkennt. $\mathcal{P}(\Sigma)$ ist zusammen mit dem Fehlerzustand $\mathcal{F}$ die Menge der Zustände. Der Startzustand ist die leere Menge. Die Übergangsfunktion $\delta(a, S)$ ist definiert durch

$$\delta(a,S) = \begin{cases} \mathcal{F}, & \text{falls } a \in S \text{ oder } S = \mathcal{F} \\ \{ x \in S \mid (x,a) \in I \} \cup \{ x \mid x \preceq a \}, & \text{sonst.} \end{cases}$$

Der Automat $W = (\mathcal{P}(\Sigma) \cup \{\mathcal{F}\}, \Sigma, \delta, \emptyset, \mathcal{F})$ akzeptiert ein Wort $w \in \Sigma^*$ genau dann, wenn w in längenlexikographischer Normalform ist. Für den Beweis wird die folgende Folgerung benötigt:

Lemma 4.2: *Sei $\delta(a_1 \dots a_k, \emptyset) \neq \mathcal{F}$. Dann ist $x \in \delta(a_1 \dots a_k, \emptyset)$ genau dann, wenn ein i ($1 \leq i \leq k$) existiert, so dass $a_i, \dots, a_k \in I(x)$ und $x \preceq a_{i-1}$.*

Beweis. Folgt direkt aus der Definition von δ. □

Satz 4.3: *Der Automat $W = (\mathcal{P}(\Sigma) \cup \{\mathcal{F}\}, \Sigma, \delta, \emptyset, \mathcal{P}(\Sigma))$ akzeptiert genau die Sprache $L = \{ nf(x) \mid x \in \Sigma^* \}$.*

Beweis. Sei $w = a_1 \dots a_n$ in längenlexikographischer Normalform. Es ist zu zeigen, dass der Automat w akzeptiert. Angenommen der Automat akzeptiert w nicht, dann existiert $1 \leq k \leq n$ so, dass $\delta(a_1 \dots a_k, \emptyset) =: S \neq \mathcal{F}$ und $\delta(a_1 \dots a_{k+1}, \emptyset) = \mathcal{F}$. Nach Konstruktion ist $a_{k+1} \in S$, aber wegen w geödätisch nicht maximal in $a_1 \dots a_k$ (Lemma 3.1). Nach dem vorherigen Lemma 4.2 existiert ein i so, dass $a_i, \dots, a_k \in I(a_{k+1})$. Insbesondere ist, da w geodätisch, $a_{k+1} \preceq a_{i-1}$. Dann aber ist

$$w = a_1 \dots a_{i-2} a_{k+1} a_{i-1} \dots a_k a_{k+2} \dots a_n \prec a_1 \dots a_{i-2} a_{i-1} \dots a_k a_{k+1} \dots a_n = w,$$

was einen Widerspruch zu w in längenlexikographischer Normalform darstellt. Also akzeptiert der Automat w.

Für die Umkehrung sei w nicht in längenlexikographischer Normalform. Es ist zu zeigen, dass der Automat w nicht akzeptiert. Sei dazu zunächst w nicht geodätisch und k minimal so, dass $\|a_1 \dots a_k a_{k+1}\| < \|a_1 \dots a_k\|$. Wegen Lemma 3.1 ist a_{k+1} maximal in $a_1 \dots a_k$. Also existiert ein i mit $a_i = a_{k+1}$. Nach Wahl von k ist $a_1 \dots a_k$ geodätisch, daher folgt aus der Deletion Property $a_i \dots a_k = a_{i+1} \dots a_k a_i$. Somit ist $a_{i+1}, \dots, a_k \in I(a_i) = I(a_{k+1})$. Dann aber ist nach Lemma 4.2 $a_{k+1} \in S := \delta(a_1 \dots a_k, \emptyset)$ und der Automat akzeptiert w nicht.

Sei also nun w geodätisch, aber nicht in längenlexikographischer Normalform. Wähle k minimal so, dass $a_1 \dots a_k$ nicht in längenlexikographischer Normalform ist. Dann gibt es ein $i < k$ mit $a_i \prec a_k$ und $a_i \dots a_k = a_k a_i \dots a_{k-1}$. Da w geodätisch ist, ist $a_i, \dots, a_{k-1} \in I(a_k)$. Nach Konstruktion des Automaten ist daher $a_k \in \delta(a_1 \dots a_{k-1}, \emptyset)$ und der Automat akzeptiert w nicht. □

Das nächste Ziel ist ein Automat W_a, der ein Paar (w, v) genau dann akzeptiert, falls $wa = v$ in der rechtwinkligen Coxeter Gruppe. Die Idee für den Automaten ist, zu überprüfen, ob $v = \mathrm{nf}(wa)$ in Σ^*. Das folgende Lemma beschreibt die möglichen Fälle für die Normalform von wa:

Lemma 4.4: *Sei $w = a_1 \dots a_n$ in längenlexikographischer Normalform, dann können für die Normalform von wa nur drei Fälle eintreten:*

$$nf(wa) = \begin{cases} a_1 \dots a_n a, & \textit{falls } wa \textit{ shortLex} \quad (4.1a) \\ a_1 \dots a_{k-1} a a_k \dots a_n, & \textit{falls } wa \textit{ geodätisch} \quad (4.1b) \\ a_1 \dots a_{k-1} a_{k+1} \dots a_n, & \textit{falls } wa \textit{ nicht geodätisch} \quad (4.1c) \end{cases}$$

Beweis. Der letzte Fall, wa nicht geodätisch, ergibt sich aus der Deletion Property 4.1. Beachte, dass in diesem Fall $a_{k-1} \neq a \neq a_{k+1}$. Der zweite Fall folgt analog und der erste Fall folgt direkt aus der Definition von shortLex. □

Das Alphabet, auf dem der Automat W_a arbeitet, ist $\Sigma^2 := (\Sigma \cup \{\$\}) \times (\Sigma \cup \{\$\}) \setminus (\$, \$)$. Wörter (w, v) werden mit \$ aufgefüllt, falls sie eine unterschiedliche Länge haben. Die Zustandsmenge ist der Fehlerzustand $\mathcal{F}$ und der Akzeptanzzustand $\mathcal{A}$ zusammen mit $\mathcal{P}(\Sigma) \cup \mathcal{P}(\Sigma) \times \{G, N\} \times \Sigma$. Der Startzustand ist wieder die leere Menge. Die Übergangsfunktion $\delta_a((s,t), \mathcal{X})$ wird im Folgenden definiert, wobei der neue Zustand $\delta_a((s,t), \mathcal{X})$ mit $\mathcal{X}'$ bezeichnet wird.

1. Falls $\mathcal{X} \subseteq \mathcal{P}(\Sigma)$:
   - (i) $\mathcal{X}' = \delta(t, \mathcal{X})$ — falls $s = t$
   - (ii) $\mathcal{X}' = \mathcal{A}$ — falls $\{s,t\} = \{\$, a\}$ und $\delta(a, S) \neq \mathcal{F}$
   - (iii) $\mathcal{X}' = (\delta(t, \mathcal{X}), G, s)$ — falls $t = a$ und $\delta(t, \mathcal{X}) \neq \mathcal{F}$
   - (iv) $\mathcal{X}' = (\delta(t, \mathcal{X}), N, t)$ — falls $s = a$ und $\delta(t, \mathcal{X}) \neq \mathcal{F}$
   - (v) $\mathcal{X}' = \mathcal{F}$ — falls $s, t \neq a$ oder $\delta(t, \mathcal{X}) = \mathcal{F}$

2. Falls $\mathcal{X} = (S, G, u) \in \mathcal{P}(\Sigma) \times \{G, N\} \times \Sigma$:
   - (i) $\mathcal{X}' = \mathcal{F}$, — falls $t \neq u$, $s \notin I(a)$ oder $\delta(t, S) = \mathcal{F}$
   - (ii) $\mathcal{X}' = (\delta(t, \mathcal{X}), G, s)$ — falls $t = u$ und $s \in \Sigma$
   - (iii) $\mathcal{X}' = \mathcal{A}$ — falls $(s,t) = (\$, t)$

3. Falls $\mathcal{X} = (S, N, u) \in \mathcal{P}(\Sigma) \times \{G, N\} \times \Sigma$:
   - (i) $\mathcal{X}' = \mathcal{F}$, — falls $s \neq u$, $t \notin I(a)$ oder $\delta(t, S) = \mathcal{F}$
   - (ii) $\mathcal{X}' = (\delta(t, \mathcal{X}), N, t)$ — falls $s = u$ und $t \in \Sigma$
   - (iii) $\mathcal{X}' = \mathcal{A}$ — falls $(s,t) = (s, \$)$

4. Falls $\mathcal{X} = \mathcal{A}$ oder $\mathcal{X} = \mathcal{F}$:
   - (i) $\mathcal{X}' = \mathcal{F}$

Der 2. Fall beschreibt die Übergänge, falls wa geodätisch, aber nicht längenlexikographisch ist. Der 3. Fall beschreibt die Übergänge, falls wa nicht geodätisch ist. Fall 3 ist symmetrisch zu Fall 2, denn falls $wa = v$ in der rechtwinkligen Coxeter-Gruppe und wa ist nicht geodätisch, dann ist $va = waa = w$ in der rechtwinkligen Coxeter-Gruppe und va ist geodätisch.

Satz 4.5: *Der Automat* $W_a = \big(\{\mathcal{F}, \mathcal{A}\} \cup \mathcal{P}(\Sigma) \cup (\mathcal{P}(\Sigma) \times \{G, N\} \times \Sigma), \Sigma^2, \delta_a, \emptyset, \mathcal{A}\big)$ *akzeptiert genau die Sprache*

$$L_a := \{\, (w, v) \mid w, v \textit{ lexikographisch und } wa = v \textit{ in der Coxeter-Gruppe} \,\}.$$

Beweis. Zeige, dass (w, v) genau dann akzeptiert wird, wenn $(w, v) \in L_a$. Der Automat akzeptiert nach Konstruktion nicht, falls $v, w \notin L$. Sei also für den restlichen Beweis $w, v \in L$. Nach Konstruktion ist der Automat symmetrisch in der Eingabe (w, v). Sei $w = s_1 \dots s_n$ und $v = t_1 \dots t_m$. Setze $\mathcal{X}_i := \delta_a((s_1 \dots s_i, t_1 \dots t_i), \emptyset)$ auf den Zustand, in dem sich der Automat nach dem Lesen von (s_i, t_i) befindet. Wähle j maximal so, dass $s_i = t_i$ für alle $i < j$. Dann gilt $\mathcal{X}_{j-1} \subseteq \mathcal{P}(\Sigma)$. Ist nun $j = n$ oder $j = k$, so tritt Regel (iii) von Fall 1 ein, welche Fall (a) der Normalform von wa oder va entspricht, und der Automat akzeptiert genau dann, wenn $(w, v) \in L_a$. Ist weder $s_j = a$, noch $t_j = a$, so ist keiner der Fälle von $\mathrm{nf}(wa)$ möglich und $wa = v$ kann nicht gelten. Durch (v) lehnt der Automat die Eingabe ab. Ohne Einschränkung sei nun $t_j = a$. Der Automat befindet sich also nach dem Lesen von (s_j, t_j) in Fall 2. Angenommen der Automat akzeptiert nach dem Lesen von (s_k, t_k) die Eingabe nicht, dann gilt entweder $t_k \neq s_{k-1}$ oder $s_k \notin I(a)$. Für $j < i < k$ gilt nach Konstruktion des Automaten

$$\begin{aligned} s_j \dots s_{i-1} &= t_{j+1} \dots t_i \\ s_j, \dots, s_{k-1} &\in I(a) \end{aligned}$$

Falls $s_k \notin I(a)$, so kann wegen Lemma 4.2 nicht $wa = v$ gelten. Sei also $s_{k-1} \neq t_k$. Dann aber kann die Normalform $\mathrm{nf}(wa)$ nicht mit v übereinstimmen und somit ist $wa = v$ nicht möglich.

Akzeptiert der Automat, so lässt sich wa wegen $s_i \in I(a)$ für $i > j$ wie in Fall (b) von $\mathrm{nf}(wa)$ schreiben. Also stimmt v mit $\mathrm{nf}(wa)$ überein und es gilt $wa = v$. □

Satz 4.6: *Rechtwinklige Coxeter-Gruppen sind shortLex automatisch.*

Beweis. Die beiden vorherigen Lemmata liefern die notwendigen Automaten W und W_a für $a \in \Sigma$. Es fehlt noch der Automat W_1. Dieser ergibt sich direkt, da W nur die längenlexikographische Normalform akzeptiert. Für den Automaten W_1 sei $w = a_1 \dots a_n$ und $v = b_1 \dots b_m$. Der Automat W_1 akzeptiert genau dann, wenn $a_i = b_i$ für $1 \leq i \leq n, m$ und $n = m$. □

5 Berechnung einer Normalform

5.1 In der freien Gruppe in logarithmischem Platz

Die Berechnung einer Normalform in der freien Gruppe in logarithmischem Platz lässt sich fast unmittelbar auf das Wortproblem zurückführen. Das Wortproblem der freien Gruppe liegt nach Bemerkung 2.4 in LogSpace.

Satz 5.1: *Sei $F(\Sigma)$ die freie Gruppe über Σ und $w \in \Sigma^*$. Dann lässt sich in logarithmischem Platz eine Normalform berechnen.*

Beweis. Sei $w = a_1 \dots a_n$. Der Algorithmus arbeitet in Phasen und startet in Phase $i = 1$. In Phase i wird a_i der Eingabe gelesen und ein maximales j mit $a_j = \overline{a_i}$ und $a_i \dots a_j = 1$ berechnet. Der Test, ob $a_i \dots a_j = 1$ lässt sich wegen Bemerkung 2.4 in logarithmischem Platz durchführen. Existiert ein solches j, so wird in Phase $j + 1$ fortgefahren, ansonsten wird a_i ausgegeben und in Phase $i + 1$ fortgefahren. Sei w_i der Inhalt des Ausgabebandes zu Beginn von Phase i. Der Algorithmus erfüllt die folgenden Invarianten.

1. $w_i = a_1 \dots a_{i-1}$ in der freien Gruppe.
2. w_i ist reduziert.

Invariante 1 ist nach Konstruktion immer erfüllt, da nur die 1 gelöscht wird. Angenommen w_i ist nicht reduziert. Sei dazu $w_i = \widetilde{a}_1 \dots \widetilde{a}_{n_i}$, wobei $n_i = |w_i|$. Es gibt Positionen $1 \leq j < k \leq n_i$ mit $\widetilde{a}_j \dots \widetilde{a}_k = 1$. $\widetilde{a}_j$ wurde aber nur ausgegeben, falls es kein Teilwort gibt, das der 1 entspricht. Da nur Teilwörter, die der 1 entsprechen gelöscht wurden, muss es für $\widetilde{a}_j$ auch vorher bereits ein passendes Teilwort gegeben haben. Also muss w_i reduziert sein. □

Der Algorithmus bildet die Basis für Theorem 5.5, in dem ein Algorithmus präsentiert wird, der eine Geodätische in einer rechtwinkligen Coxeter-Gruppe in logarithmischem Platz berechnet. Allerdings ist es dann nicht mehr ausreichend zu testen, ob das Teilwort der 1 entspricht.

5.2 In Graph-Gruppen und rechtwinkligen Coxeter-Gruppen

5.2.1 In Linearzeit

Die Berechnung einer längenlexikographischen Normalform ist nach [5] für Spurmonoide in Linearzeit möglich. Durch eine minimale Anpassung des Algorithmus lässt sich auch

eine Normalform für rechtwinklige Coxeter-Gruppen in Linearzeit berechnen, da sich in der längenlexikographischen Normalform zwei aufeinander folgende a löschen. Wegen Lemma 2.8 ist es schließlich auch möglich eine längenlexikographische Normalform in Graph-Gruppen zu berechnen.

Definition 5.1: *Sei $F \subseteq \Sigma$ mit $(a, b) \in I$ für alle $a, b \in F$. Dann bezeichnet*

$$[F] = \prod_{a \in F} a$$

einen elementaren Schritt.

Die Reihenfolge, in der die $a \in F$ multipliziert werden, spielt für das Produkt wegen $(a, b) \in I$ für alle $a, b \in F$ keine Rolle. Für die Bestimmung der Normalform wähle eine Cliquen Überdeckung $\{ \Sigma_i \mid 1 \leq i \leq k \}$ von (Σ, D) so, dass

$$\pi : M(\Sigma, I) \hookrightarrow \prod_{i=1}^{k} \Sigma_i^*, \qquad \Sigma = \bigcup_{i=1}^{k} \Sigma_i, \qquad D = (\Sigma \times \Sigma) \setminus I = \bigcup_{i=1}^{k} \Sigma_i \times \Sigma_i$$

und π eine Einbettung ist. Die Existenz einer passenden Cliquen Überdeckung ergibt sich aus dem Einbettungstheorem [5, Korollar 1.4.7]. Für $a \in \Sigma$ bezeichne component(a) die Index-Menge $\{ i \mid a \in \Sigma_i \}$. Damit lässt sich eine Zerlegung eines Repräsentanten einer Spur in ein k-Tupel realisieren.

```
function pi(s : string) : tuple;
begin
    var i : index;
    var a : char;
    var t : tuple := 1;

    while s ≠ 1 do
        a := first(s); s := rest(s);
        foreach i ∈ component(a) do
            t[i] := t[i]a;
        end
    end

    return t;
end
```

Um die längenlexikographische Normalform zu berechnen, werden sukzessive die minimalen Elemente bestimmt. Das lexikographisch kleinste Element wird dann an die Normalform angehängt. Der folgende Algorithmus bestimmt die minimalen Elemente des Tupels t. Da im späteren Verlauf bereits einige minimale Elemente m bekannt sind, werden diese aus Effizienzgründen berücksichtigt. Die Grundidee des Algorithmus ist, dass

ein Element $a \in \Sigma$ minimal ist genau dann, wenn $a = \text{first}([j])$ für alle $j \in \text{component}(a)$.

```
function min(m : step, t : tuple) : step;
begin
   var B : boolean;
   var a : char;
   /* die minimalen Elemente m sind bereits bekannt                    */
   var F : step := m;
   var i,j : index;
   var I,J : index-set;
   /* neue minimale Elemente müssen in I liegen                        */
   I := { i | t[i] ≠ 1 and i ∉ component(a) for all a ∈ m };
   while I ≠ ∅ do
      choose i ∈ I; I := I \ {i};
      a := first(t[i]); J := component(a) \ {i};
      while J ≠ ∅ and B do
         choose j ∈ J;
         if j ∈ I and a = first(t[j]) then
            /* Komponente j erfüllt die Voraussetzung und muss
               nicht weiter getestet werden                             */
            I := I \ {j}; J := J \ {j};
         else
            /* a kann nicht minimal sein                                */
            B := false;
         end
      end
      if B then
         F := F ∪ {a};
      else
         B := true;
      end
   end
   return F;
end
```

Die Bestimmung der minimalen Elemente läuft in konstanter Zeit, da die Zahl k der Komponenten fest ist. Daher ergibt sich ein Linearzeit Algorithmus für die längenlexikographische Normalform. Tritt a^2 in der längenlexikographischen Normalform auf, so wird es gekürzt. Die Konfluenz des Ersetzungssystems liefert die Eindeutigkeit der Normalform. Zur besseren Lesbarkeit wird a löschen, durch a^{-1} beschrieben, obwohl $a^{-1} = a$

in Coxeter-Gruppen.

```
function lex-NF(t : tuple) : string;
begin
    var a : char;
    var s : string;
    var F : step := min(t);
    while F ≠ ∅ do
        /* wähle aus den minimalen Elementen den lexikographisch
           kleinsten Buchstaben                                    */
        a := lexfirst(F);
        if a = last(s) then
            /* a kann gelöscht werden                             */
            s := sa^{-1};
        else
            s := sa;
        end
        t := a^{-1}t;
        F := min(F \ {a}, t);
    end
    return s;
end
```

5.2.2 In logarithmischem Platz

In diesem Abschnitt wird ein neuer Algorithmus vorgestellt, der bei Eingabe $w \in \Sigma^*$ die längenlexikographische Normalform von w in rechtwinkligen Coxeter-Gruppen und Graph-Gruppen in logarithmischem Platz berechnet. Dies zeigt nach meinem Kenntnisstand erstmals, dass das Problem der Normalformenberechnung für rechtwinklige Coxeter-Gruppen und Graph-Gruppen in LogSpace liegt.

Definition 5.2: *Ein Wort w heißt a-reduziert, falls in keiner Ableitung die Regel $a^2 \to 1$ angewendet werden kann.*

Das folgende Lemma charakterisiert die Eigenschaft a-reduziert und spielt eine wesentliche Rolle für die Berechnung einer Normalform in logarithmischem Platz. Mit $\alpha(v)$ werden die in v vorkommenden Buchstaben bezeichnet.

Definition 5.3: *Eine Spur $u \in M(\Sigma, I)$ ist ein Faktor der Spur $v \in M(\Sigma, I)$, falls $v \in M(\Sigma, I)uM(\Sigma, I)$.*

Definition 5.4: *Eine Spur w ist eine Coxeter-Spur, falls w keinen Faktor a^2 enthält.*

Lemma 5.2: *Eine Spur w ist genau dann a-reduziert, falls w keinen Faktor ava mit*

$\alpha(nf(v)) \subseteq I(a)$ *enthält.*

Beweis. Enthält w einen Faktor ava mit $\alpha(\mathrm{nf}(v)) \subseteq I(a)$, dann ist w offensichtlich nicht a-reduziert. Sei nun w nicht a-reduziert. Lässt sich die Regel $a^2 \to 1$ anwenden, so muss w einen Faktor ava mit $\alpha(v) \subseteq I(a)$ enthalten und es ist nichts weiter zu zeigen. Lässt sich die Regel nicht anwenden, ist entweder w bereits reduziert oder die Regel $b^2 \to 1$ kann für ein $b \in \Sigma$ angewendet werden. Der erste Fall, w reduziert, ist ein Widerspruch zur Voraussetzung w nicht a-reduziert. Also kann die Regel $b^2 \to 1$ angewendet werden und w lässt sich schreiben als $xbvby$ mit $\alpha(v) \subseteq I(b)$. Das Wort $w' := xvy$ ist nicht a-reduziert, also lässt sich Induktion auf das Wort $w' = xvy$ anwenden. Sei $av'a$ ein Faktor von w'. Ist $av'a$ ein Faktor von x, v oder y, so ist $av'a$ ein Faktor von w und es ist nichts weiter zu zeigen. Sei also $av'a$ kein Faktor von x, v und y und

$$xvy = a_1 \dots a_{n_x} a_{n_x+1} \dots a_{n_v} a_{n_v+1} \dots a_{n_y}$$

mit $av'a = a_{i-1} a_i \dots a_j a_{j+1}$. Es gibt drei mögliche Fälle für den Faktor in

$$w = xbvby = a_1 \dots a_{n_x} b a_{n_x+1} \dots a_{n_v} b a_{n_v+1} \dots a_{n_y}.$$

1. $i \leq n_x$ und $j \geq n_v$: b lässt sich kürzen und mit der Voraussetzung für v' ergibt sich

$$\widehat{\alpha}(a_i \dots a_{n_x} b a_{n_x+1} \dots a_{n_v} b a_{n_v+1} \dots a_j) = \widehat{\alpha}(a_i \dots a_j) = \widehat{\alpha}(v') \subseteq I(a).$$

2. $i \leq n_x$ und $j < n_v$: Es ist $n_{x+1} \leq j + 1 \leq n_v$, sonst wäre v' ein Faktor von x. Also ist wegen $a \in \alpha(v) \subseteq I(b)$ auch $a \in I(b)$. Schließlich ist

$$\begin{aligned} \widehat{\alpha}(a_i \dots a_{n_x} b a_{n_x+1} \dots a_j) &= \widehat{\alpha}(a_i \dots a_{n_x} a_{n_x+1} \dots a_j b) \\ &= \widehat{\alpha}(v'b) \subseteq \widehat{\alpha}(v') \cup \{b\} \subseteq I(a). \end{aligned}$$

3. $i > n_x$ und $j > n_v$: Folgt analog wie im 2. Fall durch $a \in \alpha(v)$.

□

Lemma 5.3: *Sei w eine Spur und $wd = udv$ mit $u, v \in \Sigma^*$, $(d, v) \in I$ und d ist das einzige maximale Element von ud. Weiter sei $\sigma_w(d) = \sum_{b \in \Sigma} \lambda_b b$, dann ist*

$$\lambda_b \neq 0 \Leftrightarrow b \in \alpha(nf(u)).$$

Beweis. Wenn $a \notin \alpha(\mathrm{nf}(u))$, dann ist $\lambda_a = 0$ nach Konstruktion der Abbildung σ_u. Für die Umkehrung sei $\widehat{u}$ die Normalform von u und $\widehat{u} = au'$. Für $a \in \Sigma$ bezeichne i_a die minimale Position von a in u'. Nach Konstruktion von σ ist $\sigma_{u'}(d) = \sigma_w(d) = \sum_{c \in \Sigma} \lambda_c c$. Es gilt die stärkere Behauptung: Falls $i_a > i_b$ und $(a, b) \in D$, dann gilt $\lambda_a > \lambda_b > 0$. Beweis der Behauptung mit Induktion. Für $|u| = 1$ ist die Behauptung

nach Voraussetzung erfüllt, denn d ist das einzige maximale Element von ud. Betrachte nun

$$\begin{aligned}
\sigma_u(d) &= \sigma_{\widehat{u}}(a) = \sigma_{au'}(d) \\
&= \sigma_a\left(\sum_{c\in\Sigma}\lambda_c c\right) \\
&= \sum_{c\in\Sigma}\lambda_c\sigma_a(c) \\
&= \sum_{a\neq d\in D(a)}\lambda_d(d+2a) + \sum_{c\in I(a)}\lambda_c c - \lambda_a a \\
&= \sum_{a\neq d\in D(a)}\lambda_d d + \sum_{c\in I(a)}\lambda_c c + \left(\sum_{a\neq d\in D(a)} 2\lambda_d - \lambda_a\right) a \\
&=: \sum_{c\in\Sigma}\mu_c c
\end{aligned}$$

Es hat sich nur der Koeffizient für a verändert, also gilt die Behauptung für $c \neq a$ nach Induktion. Falls $a \notin \alpha(v)$, so ist $\mu_a = \sum_{a\neq d\in D(a)} 2\lambda_d d > \lambda_b = \mu_b > 0$ für $(a,b) \in D$ und $b \in \alpha(v)$. Sei also $a \in \alpha(u')$. Dann ist wegen der Induktionsvoraussetzung $\lambda_d > \lambda_a$ für alle d mit $i_d < i_a$. Da u' reduziert ist, existiert mindestens ein solches d. Also ist $\mu_a > 0$. Wegen $\lambda_b > \lambda_a$ für $i_a > i_b$ und $(a,b) \in D$ ist $\mu_a > \sum_{a\neq d\in D(a)} \lambda_d d > \lambda_b = \mu_b > 0$. □

Bemerkung 5.1: Der Beweis von Lemma 5.3 zeigt zusätzlich, dass $\lambda_b > 0$, falls $b \in \alpha(\mathrm{nf}(u))$.

Lemma 5.4: *Sei $C(\Sigma, I)$ eine rechtwinklige Coxeter-Gruppe und $w \in \Sigma^*$. Dann lässt sich in logarithmischem Platz das Alphabet $\alpha(nf(w))$ der Normalform von w berechnen.*

Beweis. Sei x ein neuer Buchstabe, der noch nicht in Σ vorkommt. Es wird nur die neue Regel $x^2 = 1$ eingeführt. Dann ist insbesondere $(a,x) \notin I$ für alle $a \in \Sigma$. Betrachte $\sigma_w(x) = \sum_{b\in\Sigma}\lambda_b b$. Nach dem vorherigen Lemma 5.3 ist $\lambda_b \neq 0$ genau dann, wenn $b \in \alpha(\mathrm{nf}(w))$. Dies lässt sich nach Lemma 2.11 in logarithmischem Platz überprüfen. □

Satz 5.5: *Sei $C(\Sigma, I)$ eine rechtwinklige Coxeter-Gruppe und $w \in \Sigma^*$. Dann lässt sich in logarithmischem Platz eine Coxeter-Spur für w berechnen.*

Beweis. Wir konstruieren einen Algorithmus, der bei Eingabe $w \in \Sigma^*$ ein Wort $u \in \Sigma^*$ berechnet, das für ein festes $a \in \Sigma$ die folgenden Eigenschaften erfüllt.

1. $u = w$ in der Coxeter-Gruppe $C(\Sigma, I)$.
2. u ist a-reduziert.
3. u ist b-reduziert, falls w b-reduziert ist.

Zusammen mit Lemma 2.14 ergibt sich eine LogSpace Maschine, die bei Eingabe $w \in \Sigma^*$ ein Wort $u \in \Sigma^*$ berechnet, welches geodätisch ist.

Sei also $w = a_1 \dots a_n$ und $a \in \Sigma$ fest gewählt. Der Algorithmus arbeitet in Phasen. In Phase i wird a_i gelesen. Ist $a_i \neq a$, so gebe a_i aus und fahre mit Phase $i+1$ fort. Ist $a_i = a$ so berechne ein $j > i$ mit $a_j = a$ und $\alpha(\mathrm{nf}(a_{i+1} \dots a_{j-1})) \subseteq I(a)$ Dies ist nach Lemma 5.4 in logarithmischem Platz möglich. Falls kein solches $j > i$ existiert gebe a aus und fahre mit Phase $i+1$ fort. Existiert dieses j gebe $a_i \dots a_j$ aus, aber überspringe dabei alle a. Fahre danach mit Phase $j+1$ fort.

Zum Beweis der Korrektheit bezeichne w_{j-1} den Inhalt des Ausgabebands zu Beginn von Phase j und dem Ende von Phase i. Die Ausgabe w_{j-1} erfüllt die folgende Invariante

1. w_{j-1} ist a-reduziert.
2. $w_{j-1} = a_1 \dots a_{j-1}$ in der Coxeter-Gruppe $C(\Sigma, I)$.
3. w_{j-1} ist b-reduziert, falls $a_1 \dots a_{j-1}$ b-reduziert ist.

Beweis der Invariante über Induktion. Für $j = 1$ ist $w_{j-1} = w_0 = 1$, also ist die Invariante erfüllt. Wurde in Phase i ein a ausgegeben, so ist $w_{j-1} = w_{i-1}a$. Angenommen w_{j-1} ist nicht a-reduziert und sei $w_{j-1} = \widetilde{a}_1 \dots \widetilde{a}_{n_j}$. Dann existiert nach Lemma 5.2 ein k, wobei $1 \leq k < n_j$, mit $\widetilde{a}_k = a$ und $\alpha(\mathrm{nf}(\widetilde{a}_{k+1} \dots \widetilde{a}_{n_j-1})) \subseteq I(a)$. a_k wurde in Phase i_k ausgegeben. Nach Induktion gilt $w_{i_k-1} = a_1 \dots a_{i_k-1}$ und daher $a_{i_k} \dots a_j = \widetilde{a}_k \dots \widetilde{a}_{n_j}$. Somit ist

$$\alpha(\mathrm{nf}(a_{i_k} \dots a_j)) = \alpha(\mathrm{nf}(\widetilde{a}_k \dots \widetilde{a}_{n_j})) \subseteq I(a).$$

Dann aber hätte der Algorithmus in Phase i_k alle a zwischen i_k und j gelöscht und direkt mit Phase $j+1$ fort gefahren. Also muss w_{j-1} a-reduziert sein. Wurde in Phase i kein a ausgegeben, so folgt aus Lemma 5.2 und w_{i-1} a-reduziert direkt, dass w_{j-1} a-reduziert ist. Damit ist 1 gezeigt. 2 folgt nach der Konstruktion von w_{j-1} direkt aus Lemma 5.2. Für 3 sei $a_1 \dots a_{j-1}$ b-reduziert und in Phase i wurde ein b ausgegeben. Angenommen w_{j-1} ist nicht b-reduziert, dann gibt es nach Lemma 5.2 ein k_b, wobei $1 \leq k_b < n_j$, mit $\widetilde{a}_{k_b} = b$ und $\alpha(\mathrm{nf}(\widetilde{a}_{k_b+1} \dots \widetilde{a}_{n_j-1})) \subseteq I(b)$. Durch den Algorithmus wurden nur a gelöscht. Da $a_1 \dots a_{j-1}$ b-reduziert ist muss also $(a, b) \in I$ gelten. Seien k_a, l_a mit $1 \leq k_a < k_b < l_a < n_j$, und $a_{k_a} = a_{l_a} = a$ die Positionen, für die sich a reduzieren lässt. Da $a_1 \dots a_{j-1}$ b-reduziert ist und $(a, b) \in I$, ist $b \in \alpha(\mathrm{nf}(a_{k_a+1} \dots a_{l_a+1}))$. Dann aber hätte der Algorithmus die beiden a nicht gelöscht. Wurde in Phase i kein b ausgegeben, so folgt w_{j-1} ist b-reduziert direkt aus w_{i-1} ist b-reduziert und 3 ist gezeigt. □

Lemma 5.6: *Sei w eine Coxeter-Spur. Dann lässt sich in logarithmischem Platz ein Hasse-Diagramm für w berechnen.*

Beweis. Sei $w = a_1 \dots a_n$. Zunächst wird $(a_1, \dots, a_n)$ als Beschriftung für die Knoten $(1, \dots, n)$ des Graphen ausgegeben. Dann arbeitet der Algorithmus alle Knoten-Paare (i, j) ab. Ein Knoten-Paar (i, j) wird als Kante ausgegeben, falls

1. $i < j$ mit $(a_i, a_j) \in D$ und

2. $(a_i, a_k) \in D \Rightarrow (a_k, a_j) \notin D$ für $i < k < j$

Die Berechnung der Kanten liegt in LogSpace, da dafür nur die Zähler i, j und k gespeichert werden müssen. □

Lemma 5.7: *Sei $H = (V, E)$ ein Hasse-Diagramm zur Coxeter-Spur w und $\lambda : V \mapsto \Sigma$ die dazugehörige Beschriftung, dann lässt sich in logarithmischem Platz die längenlexikographische Normalform von w in der rechtwinkligen Coxeter-Gruppe berechnen.*

Beweis. Es werden sukzessive die minimalen Elemente ausgegeben. Dazu wird in logarithmischem Platz für jeden Buchstaben $a \in \Sigma$ ein Zähler z_a gespeichert. Alle Zähler z_a werden mit $\square$ initialisiert. Ein Buchstabe a ist minimal, falls ein Knoten i existiert, der mit $\lambda(i) = a$ beschriftet ist und keine eingehenden Kanten hat. Setze also $z_{\lambda(i)} := i$ für jeden Knoten i, der keine eingehenden Kanten hat. Die Belegung der z_a ist eindeutig, da für $i \neq j$ mit $\lambda(i) = \lambda(j) = a$ wegen $(a, a) \in D$ die Kante (i, j) im Hasse-Diagramm existiert. Für die Ausgabe der Normalform wird folgendes solange wiederholt, bis für jeden Knoten seine Beschriftung ausgegeben wurde.

> Gebe den minimalen Buchstaben a mit $z_a \neq \square$ aus und setze $z_{\lambda(z_a)} := j$ für jede Kante $(\lambda(z_a), j)$, sowie $z_a := \square$.

Das Weitersetzen der Zähler entspricht dem Löschen des Knotens aus dem Hasse-Diagramm. Da im Hasse-Diagramm eine Kante (i, j) nur dann existiert, falls es kein k mit $i < k < j$ und $(i, k), (k, j) \in E$ gibt ist die Belegung der z_a im Verlauf eindeutig. Nach Konstruktion ist das ausgegebene Wort die längenlexikographische Normalform. □

Satz 5.8: *Sei G eine rechtwinklige Coxeter-Gruppe oder eine Graph-Gruppe. Dann lässt sich bei Eingabe $w \in \Sigma^*$ in logarithmischem Platz die längenlexikographische Normalform von w berechnen.*

Beweis. Falls G eine Graph-Gruppe ist, so lässt sie sich nach Lemma 2.8 in eine rechtwinklige Coxeter-Gruppe einbetten. Es genügt also rechtwinklige Coxeter-Gruppen zu betrachten. Das Theorem folgt mit Lemma 2.14 direkt aus Theorem 5.5 und den beiden Lemmata 5.6 und 5.7. □

5.3 In Coxeter-Gruppen

Björner und Brenti stellen in [1] einen Algorithmus zur Berechnung einer Normalform für beliebige Coxeter-Gruppen vor. Dieser Algorithmus kommt mit linear vielen mathematischen Operationen aus. In diesem Abschnitt wird der Algorithmus zunächst vorgestellt und die zu berechnenden Koeffizienten genauer untersucht. Danach wird eine obere Schranke für die Speicherkomplexität der reellwertigen Koeffizienten bestimmt.

Für dieses Kapitel sei $a^* : \mathbb{R}^\Sigma \to \mathbb{R}$ mit $a^*(b) = \delta_{a,b}$. Die Menge $\{ a^* \mid a \in \Sigma \}$ bildet eine Basis des Dual Raums von $\mathbb{R}^\Sigma$. Wir identifizieren den Dual Raum $\left(\mathbb{R}^\Sigma\right)^*$ mit $\mathbb{R}^\Sigma$.

Definition 5.5 (Duale Einbettung)**:** *Mit* $\sigma^* : \left(\mathbb{R}^\Sigma\right)^* \to \left(\mathbb{R}^\Sigma\right)^*$ *wird die duale Einbettung bezeichnet, wobei* $\sigma^*(w) := \sigma^*_{a_1} \dots \sigma^*_{a_n}$ *mit* $w = a_1 \dots a_n$*. Die* b^**-Koordinate von* $\sigma^*_a(p)$ *ist definiert als*

$$(\sigma^*_a(p))_b := \begin{cases} p_b + 2\cos\left(\frac{\pi}{m_{a,b}}\right) p_a, & \text{falls } m_{a,b} \neq 0 \\ p_b + 2p_a, & \text{falls } m_{a,b} = 0 \end{cases}$$

Bemerkung 5.2: Das Wachstum der Faktoren vor $\cos\left(\frac{\pi}{m_{a,b}}\right)$ in den Koeffizienten von $\sigma^*_{a_1 \dots a_n}(p)$ ist durch 2^n beschränkt.

Zur Berechnung der Normalform wähle $p := \sum_{a \in \Sigma} a^* = (1, 1, \dots, 1)$ und $\widehat{w}_0 := 1$. Berechne nun $p'_0 := \sigma^*_{w^{-1}}(p)$. Wiederhole Folgendes, bis $p'_i = p$.

> Wähle $a \in \Sigma$ minimal mit $(p'_i)_{a^*} < 0$. Setze $\widehat{w}_i := \widehat{w}_{i-1} \cdot a$ und aktualisiere $p'_i := \sigma^*_a(p'_{i-1})$. Fahre mit Runde $i + 1$ fort.

Das berechnete $\widehat{w} := \widehat{w}_i$ ist die Normalform $\mathrm{nf}(w)$ von w. Zur Korrektheit und Termination der Berechnung siehe [1, Kapitel 4.3]

5.3.1 Berechnungskomplexität

Für den Algorithmus muss in jeder Runde entschieden werden, welche Einträge des Vektors p'_i negativ sind. Betrachte für die Analyse der Komplexität N-te Einheitswurzeln, wobei $N := \mathrm{kgV}\{\, 2m_{a,b} \mid a, b \in \Sigma \,\}$. Für diesen Abschnitt sei $N \geq 3$, da ansonsten die Berechnung von p'_i in $\mathbb{Z}$ möglich ist. Es gibt N verschiedene N-te Einheitswurzeln $e^{2\pi i j/N}$, $1 \leq j \leq N$. Der Kosinus lässt sich durch Einheitswurzeln ausdrücken, wobei i die imaginäre Einheit bezeichnet.

$$\cos\left(\frac{\pi}{m_{a,b}}\right) = \frac{e^{\pi i/m_{a,b}} - e^{-\pi i/m_{a,b}}}{2}$$

Wegen der Wahl von N als kgV, lässt sich jeder der Cosinuse als N-te Einheitswurzel ausdrücken. Das Produkt zweier N-ter Einheitswurzeln ist wieder eine N-te Einheitswurzel, also hat ein Eintrag des Vektors p'_i die Form

$$\sum_{j=1}^{n} \frac{\lambda_j}{2^{k_j}} e^{2\pi i j/N}, \text{ wobei } |\lambda_j| \leq 2^n \text{ und } 0 \leq k_j \leq n. \tag{5.1}$$

Da für den Algorithmus nur entschieden werden muss, ob Gleichung 5.1 negativ ist kann durch Durchmultiplizieren mit 2^n der Bruch eliminiert werden. Die dadurch entstandenen Faktoren werden wieder mit λ_a bezeichnet.

Lemma 5.9: *Sei* $|\lambda_j| \leq 2^n$*, dann gilt für die Linearform der* N*-ten Einheitswurzeln*

$$2^{(n+d)d} > \left| \sum_{j=1}^{N} \lambda_j e^{2\pi i j/N} \right| > \frac{1}{2^{(n+d)d}}, \text{ wobei } d = \log(N).$$

Beweis. Die erste Ungleichung folgt sofort aus $|e^{2\pi ij/N}| \leq 1$. Für die zweite siehe [11, Theorem 3]. □

Die Einträge des Vektors p'_i sind reellwertig, daher genügt es wegen $e^{i\varphi} = \cos(\varphi)+i\sin(\varphi)$ die Summe

$$\sum_{i=1}^{n} \lambda_j \cos(2\pi ij/N), \text{ wobei } -2^{2n} \leq \lambda_j \leq 2^{2n} \tag{5.2}$$

auszuwerten. Wegen Lemma 5.9 muss der Cosinus auf einen Fehler kleiner $2^{-(2n+d)(d+1)}$ approximiert werden. Für die Darstellung der Summe sind also insgesamt höchstens $(2n+d)(d+1) \in O(n)$ Bits notwendig. In der Praxis bedeutet diese Genauigkeit, dass obiger Algorithmus bestenfalls eine quadratische Laufzeit hat.

5.3.2 Das Alphabet einer Geodätischen

Das Alphabet einer Geodätischen und somit insbesondere das der längenlexikographischen Normalform lässt sich in logarithmischem Platz berechnen. Dies lies sich elementar für rechtwinklige Coxeter-Gruppen zeigen. Für allgemeine Coxeter Gruppen sei $x \notin \Sigma$ ein neuer Buchstabe mit der einzigen neuen Relation $x^2 = 1$. Damit ist insbesondere die Ordnung von ax unendlich und es gilt stets $\|wx\| > \|w\|$ für $w \in \Sigma$.

Lemma 5.10: *Sei $w \in C(\Sigma, M)$ und $\sigma_w(x) = x + \sum_{b\in\Sigma} \lambda_b b$. Dann ist $\lambda_b \neq 0$ genau dann, wenn b im Alphabet einer Geodätischen vorkommt.*

Beweis. Sei w ohne Einschränkung geodätisch. Kommt $b \in \Sigma$ nicht im Alphabet der Geodätischen vor, so ist $\lambda_b = 0$ nach Konstruktion von σ_w. Die Umkehrung folgt mit Induktion. Ist $|w| = 1$, dann folgt die Behauptung sofort aus der Konstruktion der Einbettung. Sei also $w = ua$, dann ist $\|ua\| > \|u\|$ und

$$\sigma_w(x) = \sigma_{ua}(x) = \sigma_u(x + 2a) = \sigma_u(x) + 2\sigma_u(a).$$

Nach Induktion ist $\lambda_b \neq 0$ für $a \neq b \in \Sigma$. Nach Lemma 2.5 ist $\sigma_u(a) \geq 0$, ebenso ist $\sigma_u(x) \geq 0$. Kommt a in u vor, so ist $\lambda_a > 0$ durch $\sigma_u(x)$. Sei also a nicht im Alphabet von u, dann ist $\sigma_u(a) = a + \sum_{a\neq c\in\Sigma} \mu_c c$ und daher $\lambda_a \geq 2$. □

Bemerkung 5.3: Die Einbettung σ ist unabhängig vom gewählten Repräsentanten. Das Alphabet verschiedener Geodätischer zu $w \in \Sigma^*$ stimmt also überein. Allerdings muss die Häufigkeit der Buchstaben verschiedener Geodätischer nicht übereinstimmen.

Lemma 5.11: *Das Alphabet $\alpha(nf(w))$ der längenlexikographischen Normalform von w lässt sich in logarithmischem Platz berechnen.*

Beweis. Nach Lemma 5.10 muss überprüft werden, ob $\sigma_w(x) \neq 0$. In [10] wird gezeigt, wie sich die Berechnung der Koeffizienten von $\sigma_w(x)$ in einer endlich erzeugten linearen Gruppe über $\mathbb{Z}[X]$ ausführen lassen. Für die Überprüfung, ob die Koeffizienten ungleich null sind, genügt es daher dies $\mod m$ für $m \leq n^k$ für ein festes $k \in \mathbb{N}$ zu überprüfen. □

6 Zusammenfassung und Ausblick

Coxeter-Gruppen besitzen nach Kapitel 3 ein präperfektes Ersetzungssystem. Das bedeutet, dass sich Geodätische immer mit längenerhaltenden und längenverkürzenden Regeln ableiten lassen. Außerdem sind Coxeter-Gruppen shortLex automatisch. In Kapitel 4 wurde ein elementarer Beweis, dass rechtwinklige Coxeter-Gruppen shortLex automatisch sind, angegeben.

Das wesentliche Ergebnis dieser Arbeit ist ein Algorithmus, der in Graph-Gruppen und rechtwinkligen Coxeter-Gruppen die längenlexikographische Normalform in logarithmischem Platz berechnet. Die Korrektheit des Algorithmus wurde in Kapitel 5 bewiesen. Für den Algorithmus wurde ein Kriterium eingeführt, mit dem sich das Alphabet der Normalform aus der Einbettung in die allgemeine lineare Gruppe ablesen lässt. Die Berechnung des Alphabets über diese Einbettung ist in logarithmischem Platz möglich. Weiter wurde ein Algorithmus vorgestellt, um in Coxeter-Gruppen die längenlexikographische Normalform mit linear vielen arithmetischen Operationen zu bestimmen. Diese Operationen lassen sich allerdings nicht ausschließlich mit ganzen Zahlen berechnen, daher wurde untersucht, wie viele Bits zur Repräsentation der Koeffizienten notwendig sind. Es stellte sich heraus, dass $O(n)$ Bits genügen. Abschließend wurde gezeigt, wie sich das Alphabet der längenlexikographischen Normalform in allgemeinen Coxeter-Gruppen berechnen lässt.

Für allgemeine Coxeter-Gruppen ist es offen, ob sich die Normalform in logarithmischem Platz berechnen lässt. Auch ist nicht bekannt, ob die Berechnung einer Normalform in automatischen bzw. shortLex automatischen Gruppen in logarithmischem Platz möglich ist. Hier besteht weiterer Untersuchungsbedarf.

Literaturverzeichnis

[1] Anders Björner and Francesco Brenti. *Combinatorics of Coxeter Groups*. Springer, 2005.

[2] Nicolas Bourbaki. *Lie groups and Lie algebras. Chapters 4–6*. Elements of Mathematics (Berlin). Springer-Verlag, Berlin, 2002. Translated from the 1968 French original by Andrew Pressley.

[3] E. Brieskorn and K. Saito. Artin-Gruppen und Coxeter-Gruppen. *Invent. Math.*, 17:245–271, 1972.

[4] Brigitte Brink and Robert B. Howlett. A finiteness property and an automatic structure for Coxeter groups. *Math. Ann.*, 296:179–190, 1993.

[5] Volker Diekert. *Combinatorics on Traces*. Number 454. 1990.

[6] Volker Diekert, Andrew J. Duncan, and Alexei G. Myasnikov. Geodesic rewriting systems and pregroups. In O. Bogopolski, I. Bumagin, O. Kharlampovich, and E. Ventura, editors, *Combinatorial and Geometric Group Theory*, Trends in Mathematics, pages 55–91. Birkhäuser, 2010.

[7] David B. A. Epstein, James W. Cannon, Derek F. Holt, Silvio V. F. Levy, Michael S. Paterson, and William P. Thurston. *Word Processing in Groups*. Jones and Bartlett, Boston, 1992.

[8] G. H. Hardy and E. M. Wright. *Introduction to the Theory of Numbers*. Oxford University Press, 1975.

[9] Robert M. Keller. Parallel program schemata and maximal parallelism I. Fundamental results. *Journal of the ACM*, 20(3):514–537, 1973.

[10] R. J. Lipton and Y. Zalcstein. Word problems solvable in logspace. *Journal of the ACM*, 24(3):522–526, 1977.

[11] B. Litow. On sums of roots of unity. *Automata, Languages and Programming*, 37, 2010.

[12] Antoni Mazurkiewicz. Trace theory. In W. Brauer et al., editors, *Petri Nets, Applications and Relationship to other Models of Concurrency*, number 255, pages 279–324, 1987.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Jonathan Kausch)