Solving Algorithmic Problems in Baumslag-Solitar Groups and their Extensions Using Data Compression

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Jürn-Jochen Laun

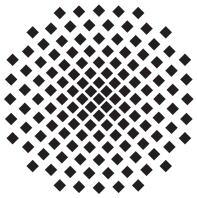
aus Filderstadt

Hauptberichter: Prof. Dr. rer. nat. habil. Volker Diekert

Mitberichter: Prof. Dr. rer. nat. habil. Markus Lohrey

Prof. Alexander Ushakov PhD

Tag der mündlichen Prüfung: 11. Dezember 2012



Universität Stuttgart
Institut für Formale Methoden der Informatik (FMI)

2012

Contents

List of Figures

Abstract

This thesis deals with algorithmic group theory and the application of data compression techniques in this area. Elements of the Baumslag-Solitar groups can be represented by comparatively short sequences of generators while their canonical normal forms are exponentially longer. As a consequence, algorithms that solve the word problem by computing such normal forms have to apply compression of the same order to their working data in order to satisfy polynomial time and space constraints. A common way to do this is to write numbers in binary. Going in the opposite direction, the problem of finding a shortest representation of a group element, a so-called geodesic, is also of interest. For example, geodesics can be used to make statements about growth inside groups. In Chapter 2, geodesic normal forms for the Baumslag-Solitar group $\mathrm{BS}(p, q)$ are defined and an algorithm is developed that computes them in polynomial time if p divides q. For arbitrary p and q, a partial solution is given which includes the horocyclic subgroup. Experimental results suggest that the horocyclic growth series of $\mathrm{BS}(2, 3)$ is irrational.

In some extensions of the Baumslag-Solitar groups, for example Higman's group and the Baumslag-Gersten group, the discrepancy between the lengths of geodesics and normal forms cannot even be described by an elementary function. This makes conventional approaches to the word problem infeasible. Recently, a data structure has been introduced that makes integers of this magnitude manageable. With the help of these so-called "power circuits", the word problem for the Baumslag-Gersten group $\mathrm{BG}(1, 2)$ has been proved to be polynomial-time solvable. In Chapter 3, the crucial reduction procedure for power circuits is improved, thereby decreasing the best known upper time bound for the word problem for $\mathrm{BG}(1, 2)$ from $\mathcal{O}(n^7)$ to $\mathcal{O}(n^3)$. At the same time, power circuits are generalized so as to allow bases other than 2. This makes the data structure apt for the generalized Baumslag-Gersten groups $\mathrm{BG}(1, q)$ with $q \geq 2$.

In Chapter 4, power circuits are used to solve the word problem for Higman's group $\mathrm{H}_4(1, 2)$, an amalgamated product of four copies of $\mathrm{BS}(1, 2)$.

For this group, a time bound of $\mathcal{O}(n^6)$ is proved. Again, the generalized notion of power circuit allows an extension of the solution to a larger class of groups $\mathrm{H}_4(1,q)$. The algorithm also works for an even more generalized version $\mathrm{H}_f(1,q)$ of Higman's group, where an arbitrary number $f \geq 4$ of copies of $\mathrm{BS}(1,q)$ are amalgamated. The time complexity of the algorithm remains $\mathcal{O}(n^6)$ and does not depend on f.

Preliminary work for this thesis has been published in journals and conference proceedings or submitted for publication [DL11, DLU12, DLU13, Lau12].

Zusammenfassung

Diese Arbeit handelt von der Anwendung von Datenkompressionstechniken in der algorithmischen Gruppentheorie. In Gruppen, welche durch Erzeuger und Relationen gegeben sind, haben Elemente grundsätzlich viele verschiedene Darstellungen als Wörter über den Erzeugenden. Bedingt durch die Konstruktion der jeweiligen Gruppe gibt es jedoch häufig eine kanonische Darstellung für jedes Element, die sogenannte Normalform. In manchen Gruppen, wie den Baumslag-Solitar-Gruppen $\mathrm{BS}(p, q)$, sind diese Normalformen exponentiell länger als die kürzestmögliche Darstellung. Dies führt dazu, dass Algorithmen zur Lösung des Wortproblems dieser Gruppen ihre Eingabe in sehr kompakter Form erhalten und daher besonders effizient arbeiten müssen, um polynomielle Zeit- und Platzschranken einzuhalten. Sollen solche Algorithmen dennoch Normalformen berechnen, so müssen sie diese ebenfalls exponentiell komprimieren. Im Falle der Baumslag-Solitar-Gruppen kann diese Kompression beispielsweise durch Verwendung der binären Zahlendarstellung erfolgen.

Von Interesse ist auch die umgekehrte Problemstellung, eine kürzeste Darstellung eines gegebenen Gruppenelements – eine sogenannte Geodätische – zu finden. Mit Hilfe Geodätischer lassen sich beispielsweise Aussagen über das Wachstum in Gruppen treffen. In Kapitel 2 werden geodätische Normalformen für $\mathrm{BS}(p, q)$ definiert und Algorithmen vorgestellt, die diese in Polynomialzeit berechnen, sofern p ein Teiler von q ist. Für beliebige Werte von p und q wird eine Teillösung für bestimmte Gruppenelemente, darunter die horozyklischen, vorgestellt. Auch auf die Spezialfälle negativer Parameter p und q wird eingegangen. Experimentelle Ergebnisse auf der Grundlage der entwickelten Algorithmen legen nahe, dass die Wachstumsreihe der Baumslag-Solitar-Gruppe $\mathrm{BS}(2, 3)$ nicht rational ist.

In bestimmten Erweiterungen der Baumslag-Solitar-Gruppen, wie zum Beispiel der Baumslag-Gersten-Gruppe, tritt das Problem der komprimierten Eingabe in verschärfter Form auf. Hier ist der Längenunterschied zwischen Geodätischen und den zugehörigen kanonischen Normalformen durch eine nicht-elementare Funktion gegeben. Dies macht Standardverfahren zur Lö-

sung des Wortproblems undurchführbar. Lange Zeit galt daher die Baumslag-Gersten-Gruppe $\mathrm{BG}(1,2)$ als Kandidat für eine Einrelatorgruppe mit sehr schwierigem Wortproblem. Vor Kurzem wurde jedoch von Myasnikov, Ushakov und Won gezeigt, dass das Wortproblem dieser Gruppe in Linearzeit lösbar ist. Dies geschah durch Verwendung einer besonderen Datenstruktur, den sogenannten „Power Circuits“. Hierbei handelt es sich um arithmetische Schaltkreise mit der Potenzierungsfunktion $(n_1, \ldots, n_k) \mapsto 2^{n_1+\ldots+n_k}$ als Gatter. Power Circuits erlauben eine sehr kompakte Darstellung extrem großer Zahlen, wie sie bei Lösung des Wortproblems in $\mathrm{BG}(1,2)$ auftreten. Die notwendigen arithmetischen Operationen lassen sich mit Power Circuits effizient durchführen. Die eigentliche Schwierigkeit besteht darin, zwei mittels Power Circuits dargestelle Zahlen zu vergleichen. Hierzu werden die Schaltkreise in eine Normalform überführt, ein Prozess, der als Reduktion bezeichnet wird. Der Reduktionsalgorithmus von Myasnikov et al. benötigt kubische Zeit. In Kapitel 3 wird dieser Algorithmus optimiert, was in einer nur noch quadratischen Laufzeit resultiert. Diese und weitere Verbesserungen schlagen sich in einer kürzeren Laufzeit des Algorithmus zur Lösung des Wortproblems in $\mathrm{BG}(1,2)$ nieder: anstelle von $\mathcal{O}(n^7)$ wird $\mathcal{O}(n^3)$ erreicht.

Gleichzeitig werden Power Circuits dahingehend verallgemeinert, dass anstelle der Basis 2 beliebige Basen $q \geq 2$ zugelassen werden. Um dies zu erreichen, müssen Konzepte wie die Kompaktheit binärer Summen, die für $q = 2$ noch sehr einfach sind, verallgemeinert und die zugehörigen Algorithmen angepasst werden. Das Ergebnis ist eine Datenstruktur, die in allen Belangen – insbesondere der Effizienz – den ursprünglichen Power Circuits ebenbürtig ist, jedoch mit beliebigen Basen $q \geq 2$ umgehen kann. Dies hat unmittelbare Auswirkungen auf gruppentheoretische Fragestellungen. Anstelle von $\mathrm{BG}(1,2)$ kann das Wortproblem nun in den verallgemeinerten Baumslag-Gersten-Gruppen $\mathrm{BG}(1,q)$ effizient gelöst werden.

Eine Gruppe mit aus algorithmischer Sicht ähnlichen Eigenschaften wie $\mathrm{BG}(1,2)$ ist die Higman-Gruppe $\mathrm{H}_4(1,2)$. Anders als die Baumslag-Gersten-Gruppe entsteht diese aus der Baumslag-Solitar-Gruppe $\mathrm{BS}(1,2)$ nicht durch eine HNN-Erweiterung, sondern durch Amalgamierung von vier Kopien von $\mathrm{BS}(1,2)$. Auch in dieser Gruppe tritt ein nicht-elementarer Längenunterschied zwischen Geodätischen und kanonischen Normalformen auf. Mit Hilfe von Power Circuits lässt sich, wie in Kapitel 4 gezeigt wird, das Wortproblem der Higman-Gruppe in $\mathcal{O}(n^6)$ Zeit lösen. Dies ist allerdings weniger offensichtlich als im Fall von $\mathrm{BG}(1,2)$ und erfordert eine detailliere Auseinandersetzung mit der Gruppenstruktur.

Die Erweiterung von Power Circuits auf beliebige Basen $q \geq 2$ ermöglicht auch bei der Higman-Gruppe eine Verallgemeinerung: das Wortproblem der Gruppe $\mathrm{H}_4(1,q)$, welche auf $\mathrm{BS}(1,q)$ aufbaut, ist ebenfalls in $\mathcal{O}(n^6)$ lösbar.

Die Higman-Gruppe erlaubt noch eine zweite Form der Verallgemeinerung: anstelle von vier Kopien der $\mathrm{BS}(1,q)$ kann eine beliebige Anzahl $f \geq 4$ amalgamiert werden. Dies liefert die Klasse von Gruppen $\mathrm{H}_f(1,q)$. In Kapitel 4 wird gezeigt, dass auch hier das Wortproblem mit Zeitaufwand $\mathcal{O}(n^6)$ gelöst werden kann, unabhängig von f.

Im Anhang wird eine Implementierung von Power Circuits vorgestellt, die im Rahmen eines studentischen Projektes im Jahr 2011 erstellt wurde. Diese als Machbarkeitsstudie ausgelegte Projektarbeit demonstriert die Verwendbarkeit der Datenstruktur in realen Anwendungen.

Teile dieser Arbeit wurden in Zeitschriften und Konferenzbänden veröffentlicht oder zur Veröffentlichung eingereicht (gemäß §2 (4) der Promotionsordnung):

- Volker Diekert and Jürn Laun. On Computing Geodesics in Baumslag-Solitar Groups. *International Journal of Algebra and Computation*, 21(1-2):119–145, 2011

- Volker Diekert, Jürn Laun, and Alexander Ushakov. Efficient algorithms for highly compressed data: The Word Problem in Higman's group is in P. In *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, volume 14 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 218–229, 2012

- Volker Diekert, Jürn Laun, and Alexander Ushakov. Efficient algorithms for highly compressed data: The word problem in Higman's group is in P. *International Journal of Algebra and Computation*, 2013. To appear

- Jürn Laun. Efficient algorithms for highly compressed data: The Word Problem in Generalized Higman Groups is in P. *Preprint*, 2012. `http://arxiv.org/abs/1006.2570`

Chapter 1

Introduction

Group theory has been a source of difficult computational problems for more than a century. Most famously, Max Dehn is credited with bringing some of them to the attention of mathematicians in 1911. The word problem, asking whether a given product of group generators equals the identity element of a group, is the most basic and most important of them. Since this was well before formal notions of computability were established, it is not surprising that the first results were positive ones. For example, Dehn gave an algorithm solving the word problem for fundamental groups of certain types of manifolds. Later, Gromov showed that Dehn's algorithm can be applied to much larger classes of groups. In the 1950s, the first negative result appeared, when Novikov (and later but independently, Boone) found groups with undecidable word problems.

Apart from mere decidability, more refined questions about complexity are of interest. Lots of groups have a polynomial time solvable word problem, a prominent example being hyperbolic (or more general, automatic) groups. For many others, no efficient algorithm is known.

For some groups, the difficulty of finding efficient algorithms arises from a phenomenon which might be described as "compression": in principle, we know an effective way of solving the problem (usually some standard technique like Britton reductions in HNN extensions), but during the computation, some words or numbers become too large (e.g. exponential in the input size or even worse). For example, an algorithm might require computing normal forms of some kind, but in the particular group, a short input might have a very long normal form.

The Baumslag-Solitar groups are the most basic example of groups having large compression in this sense. Standard techniques like Britton's Lemma yield normal forms for these groups, thus allowing conceptually simple solutions for most algorithmic problems. However, the normal form of a word

can be exponentially longer than the word itself. Thanks to the relatively mild compression (only exponential), this is easily remedied in the case of the word problem. Representing certain group elements by binary numbers, we can compress normal forms to acceptable sizes and manipulate them using integer arithmetic.

Another way of solving the word problem efficiently in a group with compression is to choose shorter normal forms or even ones with minimal length. These so-called geodesic normal forms are interesting in their own right. However, in most cases, geodesic normal forms are very hard to compute. Solving this problem for as many Baumslag-Solitar groups as possible is the main concern of Chapter 2.

The situation becomes worse, when we use Baumslag-Solitar groups as building blocks for more complicated groups. Two historically important examples are Higman's Group and the Baumslag-Gersten group. Both allow not only exponential but non-elementary compression. Immediately, the word problem becomes non-trivial, as even binary integer variables of polynomial bit-length cannot hold such numbers. The solution comes from a more potent data structure, called "power circuits", which was invented recently by Myasnikov, Ushakov, and Won, who used it to solve the word problem for the Baumslag-Gersten group in polynomial time. In Chapter 3, we present an enhanced version of this data structure and we give more efficient algorithms operating on them. This allows us to accelerate their algorithm and to apply it to a generalized class of Baumslag-Gersten groups in Chapter 4. We also prove polynomial time decidability of a generalization of Higman's group.

1.1 Algorithms and Complexity

A significant part of this work is about efficient algorithms, which is why we need to fix some conventions regarding the details of computing machinery and time measurement.

Unless stated otherwise, all computations are done on a deterministic Random Access Machine (RAM) which can perform arithmetic operations of reasonably long numbers in one step. The choice of "reasonable" length determines the actual time complexity on other models of computation such as Turing machines. In this thesis we require the bit-length of numbers to be linear in the input size. Arithmetic operations on longer numbers have to be counted extra in time analysis. This means that algorithms designed for a RAM can be simulated by a deterministic Turing machine at polynomial additional cost.

We express time and space complexity using the common Landau symbols $\mathcal{O}$, o, Ω, ω, and Θ (see e.g. [CLRS09], Chap. 3.1). Sometimes, when the exact bound is less important or depends on technical details of the implementation, we use the $\tilde{\mathcal{O}}$ notation, usually pronounced "soft Oh". This symbol ignores logarithmic factors. It is defined by

$$f \in \tilde{\mathcal{O}}(g) \quad :\Leftrightarrow \quad \exists k : f \in \mathcal{O}\left(g \cdot (\log^k(g+1)+1)\right).$$

1.2 Rewriting Systems

Rewriting systems are an important tool in algorithmic group theory and many other areas (cf. the introductory chapters of the books [BO93, Jan88]). The basic idea is to turn equivalences (for instance of strings that determine the same group element) into directed rules while preserving certain properties. We start with an alphabet Σ (possibly infinite) and the free monoid Σ^* generated by Σ. A rewriting system $\mathcal{S}$ is a set of pairs $(\ell, r) \in \Sigma^* \times \Sigma^*$. Instead of $(\ell, r) \in \mathcal{S}$ we usually write $\ell \underset{\mathcal{S}}{\longrightarrow} r$ or $\ell \longrightarrow r$ if the system is understood. Rewriting systems are sometimes called "string rewriting systems", "semi-Thue systems", or "reduction systems", although the latter often refers to the more general notion where the elements need not be strings.

The relation $\underset{\mathcal{S}}{\longrightarrow}$ gives rise to the rewriting (or reduction) relation $\underset{\mathcal{S}}{\Longrightarrow}$ defined by

$$x\ell y \underset{\mathcal{S}}{\Longrightarrow} xry \quad \text{if and only if } \ell \underset{\mathcal{S}}{\longrightarrow} r \text{ and } x, y \in \Sigma^*.$$

As usual, the reflexive transitive closure of $\underset{\mathcal{S}}{\Longrightarrow}$ is denoted $\overset{*}{\underset{\mathcal{S}}{\Longrightarrow}}$. The symmetric reflexive transitive closure $\overset{*}{\underset{\mathcal{S}}{\Longleftrightarrow}}$ is a congruence relation. It defines the quotient monoid $\Sigma^* / \overset{*}{\underset{\mathcal{S}}{\Longleftrightarrow}}$.

A string that contains no left-hand side of any rule in $\mathcal{S}$ is called irreducible (with respect to $\mathcal{S}$). The set of all irreducible elements is denoted $\mathrm{IRR}(\mathcal{S})$.

There are several important properties that rewriting systems may possess:

- $\mathcal{S}$ is called *locally confluent* if for all $u, v', v'' \in \Sigma^*$ with $v' \underset{\mathcal{S}}{\Longleftarrow} u \underset{\mathcal{S}}{\Longrightarrow} v''$ there is an element $w \in \Sigma^*$ with $v' \overset{*}{\underset{\mathcal{S}}{\Longrightarrow}} w \overset{*}{\underset{\mathcal{S}}{\Longleftarrow}} v''$.
- $\mathcal{S}$ is called *confluent* if for all $u, v', v'' \in \Sigma^*$ with $v' \overset{*}{\underset{\mathcal{S}}{\Longleftarrow}} u \overset{*}{\underset{\mathcal{S}}{\Longrightarrow}} v''$ there is an element $w \in \Sigma^*$ with $v' \overset{*}{\underset{\mathcal{S}}{\Longrightarrow}} w \overset{*}{\underset{\mathcal{S}}{\Longleftarrow}} v''$.

- $\mathcal{S}$ is said to have the *Church-Rosser* property if for all $v', v'' \in \Sigma^*$ with $v' \overset{*}{\underset{\mathcal{S}}{\Longleftrightarrow}} v''$ there is an element $w \in \Sigma^*$ with $v' \overset{*}{\underset{\mathcal{S}}{\Longrightarrow}} w \overset{*}{\underset{\mathcal{S}}{\Longleftarrow}} v''$.
- $\mathcal{S}$ is called *terminating* (or "noetherian") if every infinite chain

$$u_1 \overset{*}{\underset{\mathcal{S}}{\Longrightarrow}} u_2 \overset{*}{\underset{\mathcal{S}}{\Longrightarrow}} \dots$$

  eventually becomes stationary (i.e., there is some n_0 with $u_i = u_j$ for all $i, j \geq n_0$).

Obviously, the Church-Rosser property implies confluence which in turn implies local confluence. In fact, confluence and Church-Rosser are equivalent ([BO93], Lem. 1.1.7). Local confluence and termination together imply confluence ([BO93], Thm. 1.1.13).

If $\mathcal{S}$ is confluent, the irreducible elements $\mathrm{IRR}(\mathcal{S})$ are unique representatives of their respective congruence classes in Σ^*. If $\mathcal{S}$ is terminating, each element of $\Sigma^* / \overset{*}{\underset{\mathcal{S}}{\Longleftrightarrow}}$ has an irreducible representative. Rewriting systems that are both confluent and terminating (sometimes called "convergent") are of particular importance as they provide a unique irreducible representative (normal form) for each class.

In general, it is undecidable whether a given system is confluent, locally confluent, or terminating, even if it is finite ([Jan88], Chap. 2.1). For specific systems, a common strategy is to prove local confluence and termination. Local confluence can be tested by looking at all possible overlaps of the left-hand sides of the rules.

For termination, there is an obvious sufficient condition: a system $\mathcal{S}$ is called *length-reducing* if for every rule $(\ell, r) \in \mathcal{S}$ we have $|\ell| > |r|$. The following lemma is folklore (e.g. [DDM10], Lem. 3.1), but we give a complete proof since we will rely heavily on it later.

Lemma 1.1. *Let $\mathcal{S}$ be a length-reducing rewriting system. Then for any word w we find a word $\hat{w} \in \mathrm{IRR}(\mathcal{S})$ with $w \overset{*}{\underset{\mathcal{S}}{\Longrightarrow}} \hat{w}$ using at most $|w| \cdot (C+1)$ tests for reducibility (at most $|w|$ reduction steps are actually conducted, obviously), where*

$$C = \max\left\{ \frac{|r|}{|\ell| - |r|} \,:\, (\ell, r) \in \mathcal{S} \right\}.$$

Proof. We work with a word uv, where $u \in \mathrm{IRR}(\mathcal{S})$ and $w \overset{*}{\underset{\mathcal{S}}{\Longrightarrow}} uv$. At the beginning we have $u = 1$ and $v = w$ and we gradually enlarge u while shortening v until we end up with $u = \hat{w}$ and $v = 1$. We will show that in every step the number

$$p(u, v) := |uv| \cdot C + |v|,$$

which is initially $|w|\cdot(C+1)$, decreases at least by one. This gives the bound on the number of tests.

Let a be the first letter of v and $v = av'$. If ua does not contain the left-hand side of any rule in $\mathcal{S}$, we can replace u by $u' := ua$ and v by v'. This does not change $|uv|$, but $|v|$ is decreased by one, so $p(u', v') = p(u, v) - 1$.

If ua does contain the left-hand side of some rule $(\ell, r) \in \mathcal{S}$, then due to irreducibility of u, ℓ must involve a. We define u' by $ua = u'\ell$ and we get the situation shown in Figure 1.1 (a). Replacing ℓ by r and setting $v'' := rv'$ we arrive at Figure 1.1 (b), where u' (being a subword of u) is irreducible, and we have

$$p(u', v'') - p(u, v) \le (|r| - |\ell|) \cdot C + |r| - 1 \le -1.$$

□

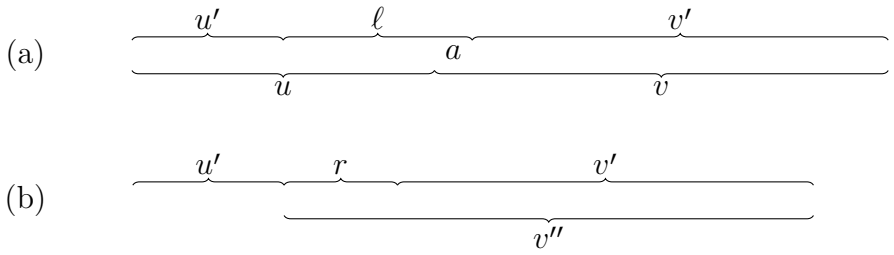


Figure 1.1: Word problem for length-reducing rewriting systems

1.3 Groups, Amalgams, and HNN Extensions

In algorithmic group theory, groups are usually given via a set X of generators and a set $\mathcal{R} \subseteq (X \cup X^{-1})^*$ of relators which are sequences of generators and their inverses. The thus presented group is defined by

$$\langle X \mid \mathcal{R} \rangle := F(X) / \lhd \mathcal{R} \rhd,$$

where $F(X)$ is the free group with basis X and $\lhd \mathcal{R} \rhd$ is the normal subgroup of $F(X)$ generated by $\mathcal{R}$. For example, $\langle \{a, b\} \mid \{aba^{-1}b^{-1}\} \rangle$ denotes the group $\mathbb{Z} \times \mathbb{Z}$. It is common to relax the notation and write $\langle a, b \mid ab = ba \rangle$ instead.

Elements of the group $\langle X \mid \mathcal{R} \rangle$ are written as strings $w \in (X \cup X^{-1})^*$. This notation is ambiguous, for instance aba, a^2b, and $b^{-1}b^2a^2$ denote the same element in $\langle a, b \mid ab = ba \rangle$. Whenever it is crucial to distinguish equality of strings from equality of group elements, we write $=$ for the former and $\sim$ for the latter.

The most important algorithmic problem that arises from the representation of groups by generators and relators is the word problem: given a word $w \in (X \cup X^{-1})^*$, determine whether $w \sim 1$ in $G = \langle X \mid \mathcal{R} \rangle$. The word problem was formulated by Dehn in his 1912 paper [Deh12] (where it was more aptly called the "identity problem"), along with other algorithmic problems in group theory such as the conjugacy problem ("transformation problem") and the isomorphism problem. All three problems are undecidable for arbitrary (finitely presented) groups [Nov55, Boo58], although many classes of groups are known where some or all of these problems are solvable.

If $\langle X \mid \mathcal{R} \rangle$ is a finite presentation ($|X|, |\mathcal{R}| < \infty$), the word problem is recursively enumerable, since every element in the subgroup $\lhd \mathcal{R} \rhd$ of $F(X)$ is a finite product of conjugates of relators from $\mathcal{R}$. The Dehn function $\mathrm{Dehn}_{X,\mathcal{R}}(n)$ gives the least upper bound of the number of relators that are needed for words of length $\leq n$. The Dehn function is recursive if and only if the word problem is solvable. Although there are some further ties between the complexity of the word problem and the growth of the Dehn function (e.g. [Ger93], Thm. 3.1), the gap between the two can be quite large. For example, the Baumslag-Gersten group has a non-elementary Dehn function, but the word problem is solvable in polynomial time, as we will see in Section 4.3.

We use two major methods of constructing new groups from existing ones: HNN extensions and amalgamated products. HNN extensions (named after Graham Higman, Bernhard H. Neumann, and Hanna Neumann [HNN49]) start with a group G, an unused symbol t called "stable letter", and an isomorphism $\varphi : H_1 \tilde{\to} H_2$ of two subgroups H_1, H_2 of G. In the HNN extension

$$\mathrm{HNN}(G, t, H_1 \tilde{\to} H_2) := G * \langle t \rangle / \lhd t h_1 t^{-1} = \varphi(h_1) \,:\, h_1 \in H_1 \rhd,$$

the subgroups are conjugated via t. The most important property of HNN extensions is that the original group G embeds in $\mathrm{HNN}(G, t, H_1 \tilde{\to} H_2)$. This follows from Britton's Lemma:

Theorem 1.2. *(Britton's Lemma, cf. [LS01], Chap. IV Sect. 2) If w is a non-empty alternating sequence of elements $g \in G \setminus \{1\}$ and powers of letters t^n ($n \in \mathbb{Z} \setminus \{0\}$) and if $w \sim 1$ in $\mathrm{HNN}(G, t, H_1 \tilde{\to} H_2)$, then w contains either a subword tgt^{-1} with $g \in H_1$ or a subword $t^{-1}gt$ with $g \in H_2$.* □

This subword can be replaced by $\varphi(g)$ or $\varphi^{-1}(g)$. Such a replacement is called a "Britton reduction" and often denoted by the symbol $\underset{\mathcal{B}}{\Longrightarrow}$. In other words, Britton's Lemma states that a non-empty Britton-reduced sequence does not equal the identity in an HNN extension.

We formalize the notion of Britton reduction by giving a rewriting system $\mathcal{B}$ over $(G \setminus \{1\} \cup \{t, t^{-1}\})^*$:

$$\begin{aligned} tt^{-1}, t^{-1}t &\longrightarrow 1 \\ g_1g_2 &\longrightarrow g && \text{if } g_1g_2 \sim g \text{ in } G \\ tgt^{-1} &\longrightarrow \varphi(g) && \text{if } g \in H_1 \\ t^{-1}gt &\longrightarrow \varphi^{-1}(g) && \text{if } g \in H_2 \end{aligned}$$

On the right-hand sides of these rules, we identify the group identity with the empty string. It follows from Britton's Lemma that this system is confluent on the subset of strings representing the identity element of the group. However, it is not globally confluent and in particular, group elements other that 1 may have more than one irreducible representation.

Since the rewriting system $\mathcal{B}$ is length-reducing, we obtain an algorithm for the word problem in HNN extensions, given that we have effective access to H_1, H_2, and φ:

Proposition 1.3. *If the subgroup membership problems of H_1 and H_2 in G are solvable and we can effectively compute $\varphi : H_1 \to H_2$ and $\varphi^{-1} : H_2 \to H_1$, then the word problem in* $\mathrm{HNN}(G, t, H_1 \tilde{\to} H_2)$ *is solvable.* □

The subgroup membership problem of a subgroup $H \leq G$ asks whether a given element of G is in fact in the subgroup H.

There are also confluent rewriting systems for HNN extensions. If C_1 and C_2 are right coset representatives of H_1 and H_2 in G with $1 \in C_1 \cap C_2$, we can define the following rules:

$$\begin{aligned} tt^{-1}, t^{-1}t &\longrightarrow 1 \\ g_1g_2 &\longrightarrow g && \text{if } g_1g_2 \sim g \text{ in } G \\ g_1tg_2 &\longrightarrow g_1'tc_1 && \text{if } g_2 = h_1c_1 \text{ with } h_1 \in H_1,\ c_1 \in C_1, \text{ and } g_1' \sim g_1\varphi(h_1) \\ g_1t^{-1}g_2 &\longrightarrow g_1't^{-1}c_2 && \text{if } g_2 = h_2c_2 \text{ with } h_2 \in H_2,\ c_2 \in C_2, \text{ and } g_1' \sim g_1\varphi^{-1}(h_2) \end{aligned}$$

One can easily show that this system is both locally confluent and terminating, and hence confluent. The rules of this second system leave Britton-reduced strings Britton-reduced, which proves Britton's Lemma. The normal forms with respect to this system start with an element from G, followed by an alternating sequences of representatives from $C_1 \cup C_2$ and letters $t^{\pm 1}$, where t is followed by an element from C_1 and t^{-1} by one from C_2.

Amalgamated products (also called "free products with amalgamation") are the second important construction. We start with two groups G_1, G_2

that contain isomorphic subgroups, i.e., $G_1 \overset{\iota_1}{\hookleftarrow} H \overset{\iota_2}{\hookrightarrow} G_2$. The amalgamated product of G_1 and G_2 with respect to H is defined by

$$G_1 *_H G_2 := G_1 * G_2 / \lhd\, \iota_1(h) = \iota_2(h) \,:\, h \in H \,\rhd\,.$$

Like with HNN extensions, the groups G_1 and G_2 embed into $G_1 *_H G_2$. There is an analogon of Britton's Lemma:

Theorem 1.4. *If a non-empty alternating sequence w of elements from $G_1 \setminus \{1\}$ and $G_2 \setminus \{1\}$ equals 1 in the amalgamated product $G_1 *_H G_2$, then one of the elements in the sequence is in $\iota_1(H)$ or $\iota_2(H)$.* □

This element, say g, can be replaced by $\iota_2(\iota_1^{-1}(g))$ or $\iota_1(\iota_2^{-1}(g))$ and then merged with its neighbor(s), making the sequence again alternating but shorter. There is no specific name for this operation, although one might arguably call it a Britton reduction as well. Like with Britton reductions for HNN extensions, we get a terminating rewriting system over $(G_1 \setminus \{1\} \cup G_2 \setminus \{1\})^*$ that is confluent on the subset of sequences representing the group identity:

$$\begin{aligned} g'g'' \longrightarrow g \qquad & \text{if } g', g'' \in G_1 \text{ or } g', g'' \in G_2 \text{ and in both cases } g'g'' \sim g \\ & \text{or if } g' = \iota_1(h) \text{ and } g \sim \iota_2(h)g'' \in G_2 \\ & \text{or if } g' = \iota_2(h) \text{ and } g \sim \iota_1(h)g'' \in G_1 \\ & \text{or if } g'' = \iota_1(h) \text{ and } g \sim g'\iota_2(h) \in G_2 \\ & \text{or if } g'' = \iota_2(h) \text{ and } g \sim g'\iota_1(h) \in G_1 \end{aligned}$$

Again, this leads to an algorithm for the word problem:

Proposition 1.5. *If the subgroup membership problems for $\iota_1(H)$ in G_1 and for $\iota_2(H)$ in G_2 are solvable and we can effectively compute $\iota_2 \circ \iota_1^{-1}$ and $\iota_1 \circ \iota_2^{-1}$, then the word problem in $G_1 *_H G_2$ is solvable.* □

Finally, there is also a confluent rewriting system for $G_1 *_H G_2$. Let C_1 and C_2 be right coset representatives of $\iota_1(H)$ in G_1 and $\iota_2(H)$ in G_2, respectively.

$$\begin{aligned} g'g'' &\longrightarrow g && \text{if } g', g'' \in G_1 \text{ or } g', g'' \in G_2 \text{ and in both cases } g'g'' \sim g \\ g_1g_2 &\longrightarrow g_1'c_2 && \text{if } g_2 \sim \iota_2(h)c_2 \text{ with } c_2 \in C_2 \text{ and } g_1' \sim g_1\iota_1(h) \in G_1 \\ g_2g_1 &\longrightarrow g_2'c_1 && \text{if } g_1 \sim \iota_1(h)c_1 \text{ with } c_1 \in C_1 \text{ and } g_2' \sim g_2\iota_2(h) \in G_2 \end{aligned}$$

Chapter 2

Baumslag-Solitar Groups

The Baumslag-Solitar groups $BS(p,q)$ have a simple description, yet they feature characteristics that render them interesting, especially as counterexamples. In this chapter we explore their basic properties, so we can build upon them in later chapters. In addition, we investigate the problem of finding geodesics. From the results, we deduce statements about the growth series of these groups.

Definition 2.1. *For any two integers $p, q \in \mathbb{Z}$, the group*

$$\mathrm{BS}(p,q) = \langle a, t \mid ta^p t^{-1} = a^q \rangle.$$

is called the Baumslag-Solitar group*.*

If $p = 0$ or $q = 0$, then $\mathrm{BS}(p,q)$ is isomorphic to the free product of $\mathbb{Z}$ and another cyclic group. We exclude these rather uninteresting cases from further consideration. Other special cases are $\mathrm{BS}(1,1) = \mathbb{Z}\times\mathbb{Z}$ and $\mathrm{BS}(1,-1)$, which is the fundamental group of the Klein bottle.

The mapping $a \mapsto a$, $t \mapsto t^{-1}$ defines an isomorphism $\mathrm{BS}(p,q) \tilde{\to} \mathrm{BS}(q,p)$. Furthermore, $\mathrm{BS}(p,q) = \mathrm{BS}(-p,-q)$. This allows us to fix the following convention:

Remark 2.2. *From now on, for any Baumslag-Solitar group $\mathrm{BS}(p,q)$, we assume that $0 < p \leq |q|$.*

Note that when working with words over the alphabet $\{a, a^{-1}, t, t^{-1}\}$, the above isomorphism is a letter-by-letter substitution that can be computed in linear time and is length-preserving. Thus, the restriction $0 < p \leq |q|$ remains justified when we discuss algorithmic problems such as the word problem or the geodesic search problem.

The Baumslag-Solitar groups were introduced by Gilbert Baumslag and Donald Solitar in [BS62]. They used $\mathrm{BS}(2,3)$ as an example of a finitely generated one-relator group that is non-Hopfian. In fact, $\mathrm{BS}(p,q)$ is residually finite if and only if $|p| = 1$ or $|p| = |q|$ (see [BS62] with a correction in [Mes72]) and in this case also Hopfian. In the non-residually finite case, $\mathrm{BS}(p,q)$ is Hopfian if and only if p and q have the same set of prime divisors.

The group $\mathrm{BS}(p,q)$ is one of the most basic examples of an HNN extension:

$$\mathrm{BS}(p,q) = \mathrm{HNN}(\langle a\rangle, t, \langle a^p\rangle \tilde{\to} \langle a^q\rangle)$$

Therefore, Britton's Lemma applies, so any (freely reduced) word over $\{a, a^{-1}, t, t^{-1}\}$ that equals 1 in the group has to contain a subword of the form $ta^{n\cdot p}t^{-1}$ or $t^{-1}a^{n\cdot q}t$. For example, if $2 \le p \le |q|$ and r divides p, then

$$[a, ta^{p/r}t^{-1}] = ata^{p/r}t^{-1}a^{-1}ta^{-p/r}t^{-1} \neq 1 \text{ in } \mathrm{BS}(p,q),$$

so the mapping

$$\varphi : \mathrm{BS}(p,q) \to \mathrm{BS}(p,q);\ a \mapsto a^r,\ t \mapsto t$$

has a non-trivial kernel. Yet, if r is a prime dividing only p but not q, then $a \in \varphi(\mathrm{BS}(p,q))$, which proves the above statement about non-Hopficity.

An important special case is $\mathrm{BS}(1,q)$. This group is isomorphic to the semi-direct product $\mathbb{Z}[1/q] \rtimes \mathbb{Z}$, where $\mathbb{Z}[1/q]$ is the (additive) group of fractions whose denominators are powers of q. Since the projection of $\mathbb{Z}[1/q] \rtimes \mathbb{Z}$ onto its second component has the abelian group $\mathbb{Z}[1/q]$ as its kernel, $\mathrm{BS}(1,q)$ is metabelian. Baumslag-Solitar groups with $p > 1$ are not solvable.

None of the Baumslag-Solitar groups $\mathrm{BS}(p,q)$ are hyperbolic (apart from the excluded cases $p \cdot q = 0$). If $p \neq |q|$, $\mathrm{BS}(p,q)$ is not even automatic since its Dehn function is exponential [Ger91]. Moreover, a group that contains a Baumslag-Solitar group cannot be hyperbolic. In particular, Dehn's algorithm cannot be used to solve the word problem. However, since we have Britton's Lemma, there is no need for that.

Throughout this chapter, we regard p and q as constants. The time bounds stated in the results are not uniform with respect to these parameters.

2.1 The Word Problem

Writing $\mathrm{BS}(p,q)$ as an HNN extension immediately proves decidability of the word problem. However, the resulting algorithm uses exponential space. For instance, starting with the word $t^n a t^{-n}$ in $\mathrm{BS}(1,2)$, the computed normal form is a^{2^n}. Storing the exponents of a's in binary circumvents this problem and results in a polynomial time solution of the word problem:

Theorem 2.3. *The word problem in* $\mathrm{BS}(p,q)$ *can be solved in linear time (on a RAM;* $\tilde{\mathcal{O}}(n^2)$ *time on a Turing machine).*

More precisely, given a word $w \in \{a, a^{-1}, t, t^{-1}\}^*$*, it takes at most linear time (*$\tilde{\mathcal{O}}(|w|^2)$ *time on a Turing machine) to find a Britton-reduced word* $\hat{w}$ *with* $w \overset{*}{\underset{\mathcal{B}}{\Longrightarrow}} \hat{w}$.

Proof. We take the Britton rewriting system $\mathcal{B}$ for $\mathrm{BS}(p,q)$ over the infinite alphabet $\Sigma = \{t, t^{-1}, a^n : n \in \mathbb{Z} \setminus \{0\}\}$ as defined in Section 1.3. Note that a^n is regarded as a single letter, needing $\mathcal{O}(\log n)$ space on the tape of a Turing machine. The input string consists only of the letters t, t^{-1}, a^1, a^{-1}. The rules of $\mathcal{B}$ are:

$$\begin{aligned} a^m a^n &\longrightarrow a^{m+n} \\ t a^{n\cdot p} t^{-1} &\longrightarrow a^{n\cdot q} \\ t^{-1} a^{n\cdot q} t &\longrightarrow a^{n\cdot p} \qquad (\text{for } m, n \in \mathbb{Z}) \end{aligned}$$

For convenience, we identify a^0 with the empty word. As stated in Theorem 1.2, the system $\mathcal{B}$ is not confluent, but every word that equals 1 in the group $\mathrm{BS}(p,q)$ reduces to the empty word.

By Lemma 1.1, we need at most $2n$ tests for reducibility to solve the word problem, if we start with an input word of length n. The sum of the exponents is bounded by $|q/p|^n \subseteq 2^{\mathcal{O}(n)}$, so their bit-length in binary notation remains linear. □

In $\mathrm{BS}(p,q) = \langle a, t \mid t a^p t^{-1} = a^q \rangle$, the subgroup generated by a is called the horocyclic subgroup. The HNN construction of $\mathrm{BS}(p,q)$ implies that $\langle a \rangle$ is isomorphic to $\mathbb{Z}$. From Theorem 2.3 we obtain:

Corollary 2.4. *The subgroup membership problem for the horocyclic subgroup is solvable in linear time (*$\tilde{\mathcal{O}}(n^2)$ *time on a Turing machine). Moreover, given a word* $w \in \{a, a^{-1}, t, t^{-1}\}^*$*, we can compute the number* α *(in binary notation) such that* $w \sim a^\alpha$ *within the same time bound.*

This follows simply from the fact that a^α is the unique Britton-reduced form for horocyclic elements. In fact, if $w \sim a^\alpha$, then $w a^{-\alpha} \sim 1$, and all Britton reductions must occur inside w. □

Recently, Elder, Elston, and Ostheimer have proposed a logspace algorithm for the word problem in solvable Baumslag-Solitar groups $\mathrm{BS}(1,q)$ [EEO12].

2.2 Geodesics and Growth

Elements of a group G presented by $\langle X \mid \mathcal{R} \rangle$ are expressed by strings from $(X \cup X^{-1})^*$. Among the possibly many words representing the same group element, those with minimal length are called geodesics (with respect to the generating set X). They describe shortest paths from the origin of the Cayley graph of G (w.r.t. X) to the node corresponding to the group element in question. In general, geodesics of a given group element $g \in G$ are far from being unique. However, their length, called the geodesic length of g, is well-defined for fixed X.

If the word problem for a group has time complexity $t(n)$, then the problem of deciding whether a given word is geodesic is in co-NTIME($t(2n)$), for one can guess a shorter word and verify that it represents the same group element. Using Theorem 2.3 we get co-NTIME(n) for the Baumslag-Solitar groups. The geodesic search problem which, given a group element of BS(p, q), asks for a corresponding geodesic word (or all of them), has an obvious solution in linear space: enumerate all words in order of increasing length and look for the first one that represents the group element.

More efficient algorithms are not as easy to obtain. The first result in this area was by Elder [Eld10], who presented a linear time algorithm for the geodesic search problem for the solvable groups BS$(1, q)$. In [DL11], Diekert and the author extended these results by showing that the problem is solvable for BS(p, q) in polynomial time if $p \mid q$ (see Section 2.7). It is open whether this remains true for arbitrary p and q. There are partial results by Freden et al. [FKS11] concerning the horocyclic subgroup $\langle a \rangle \leq$ BS(p, q) (see also Section 2.4).

Apart from being an interesting subject on their own, geodesic words are connected to other important properties of groups. For a group G, finitely generated by X, define

$$B_n(G, X) := |\{g \in G : g \text{ has geodesic length } n \text{ (w.r.t. } X)\}|.$$

This is the cardinality of the sphere consisting of the group elements at distance n from the identity in the Cayley graph. If we define two sequences $(a_n)_{n\in\mathbb{N}}$ and $(b_n)_{n\in\mathbb{N}}$ to be equivalent if there is a number C such that $a_{n/C} \leq b_n \leq a_{n\cdot C}$ for all n, the equivalence class of $(B_n(G, X))_{n\in\mathbb{N}}$ is called the growth rate of G (see [Mil68]). The growth rate does not depend on the set X. The growth rate of $G = \langle X \mid \mathcal{R} \rangle$ is naturally bounded by $(2 \cdot |X|)^n$. Apart from groups with exponential (e.g. BS$(2, 3)$) or polynomial growth, there are groups with an intermediate growth rate, as was shown by Grigorchuk [Gri83].

The formal power series

$$S_{G,X}(z) = \sum_{n\geq 0} B_n(G,X) z^n$$

is called the growth series of G (w.r.t. X). If $\mathcal{N} \subseteq (X \cup X^{-1})^*$ is a set of geodesic normal forms (a set of geodesic words containing exactly one for each group element), then $S_{G,X}(z) = \sum_{n\geq 0} |\mathcal{N} \cap (X \cup X^{-1})^n| \, z^n$. The case where $S_{G,X}(z)$ is rational (i.e., the quotient of two polynomials) is of particular interest, since then all the information about growth inside the group is encoded in a finite object. According to a theorem by Chomsky and Schützenberger, the growth series is rational if some set of geodesic normal forms is an unambiguous linear context-free language [CS63, Kui70].

Example 2.5. *Both for* BS$(1,1)$ *and for* BS$(1,-1)$, $\{t^m a^n : m,n \in \mathbb{Z}\}$ *is a set of geodesic normal forms. There is one word of length* 0 *and for* $n \geq 1$ *there are* $4n$ *words of length* n *(*$a^{\pm n}$, $t^{\pm n}$, *and* $t^{\pm i}a^{\pm(n-i)}$ *for* $0 < i < n$*), so the growth series of these groups is*

$$S_{\mathrm{BS}(1,\pm 1),\{a,t\}}(z) = 1 + \sum_{n\geq 1} 4nz^n = 1 + \frac{4z}{(1-z)^2}.$$

The growth series is also rational and known for the other residually finite cases BS$(1,q)$ ([CEG94] and Corollary 2.11; rationality independently by [Bra74]) and BS(q,q) ([EJ92] and Theorem 2.41 and Corollary 2.11) and for some other groups, mostly arising from geometry (see Section 0 of [EJ92]). For other Baumslag-Solitar groups, such as BS$(2,3)$, the growth series is unknown, although there have been attempts leading to partial results, see [FKS11] and Section 2.4.1.

2.3 The Structure of BS(p,q)

In order to make words over the generators a and t of BS(p,q) and their inverses a^{-1} and t^{-1} easier to read, we use T as a shorthand for t^{-1}. Elements of the horocyclic subgroup $\langle a \rangle \simeq \mathbb{Z}$ are identified with the corresponding integer. For instance, we write:

$$taaaat^{-1}a^{-1}a^{-1} = ta^4Ta^{-2} = t4T(-2)$$

Words containing the letters t and T as well as integers, may be regarded as mere abbreviations or, alternatively, as elements of the group $\langle t \rangle * \mathbb{Z}$, which has BS$(p,q)$ as a quotient. When it comes to measuring the length $|w|$ of a word w, in order to sustain consistency, the length of an integer is defined

as its absolute value. For example, $|t4T(-2)| = 1 + |4| + 1 + |-2| = 8$. We use greek letters for horocyclic elements. The Britton rewriting system from Section 2.1 translates to a system for $\langle t \rangle * \mathbb{Z}$, which we also call $\mathcal{B}$.

The Cayley graph of $\mathrm{BS}(p, q)$ is made of "bricks" corresponding to the relator $ta^p t^{-1} a^{-q}$ (see Figure 2.1). Starting at the line that corresponds to the

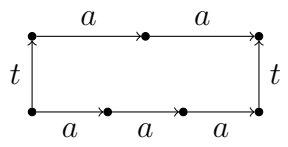


Figure 2.1: A "brick", a 2-cell in the Cayley complex of $\mathrm{BS}(2, 3)$

horocyclic subgroup $\langle a \rangle$, we glue the a^q side of a brick to every a^q segment of that line. These bricks naturally fall into q classes, each of which forms a "row" of bricks. The same can be done with the a^p side, resulting in a further p such rows. The part of the Cayley graph constructed so far is depicted in Figure 2.2. In all pictures, we draw the t edges going in an upward direction and the a edges sideways. In order to construct the whole Cayley graph, we have to repeat this procedure recursively for the lines $a^i t \langle a \rangle$ $(0 \le i < q)$ and $a^i T \langle a \rangle$ $(0 \le i < p)$. Thus, topologically, the Cayley complex is the direct product of $\mathbb{R}$ and an infinite tree in which all nodes have degree $p + q$.

For every infinite simple upward-going path in this tree starting at 1 and given by the word $w = \alpha_0 t \alpha_1 t \ldots$, the set $\bigcup_{i \ge 0} \alpha_0 t \alpha_1 \ldots \alpha_{i-1} t \langle a \rangle \subseteq \mathrm{BS}(p, q)$ is called a sheet of the Cayley graph. If $w = t^\omega$, this is called the main sheet (cf. Sect. 2 of [FKS11]). The main sheet of $\mathrm{BS}(2, 3)$ is depicted in Figure 2.3. This picture can be interpreted as an embedding of the sheet graph (endowed with the word metric) into the Poincaré half-plane model of the hyperbolic plane. The line corresponding to $\langle a \rangle$ becomes a horocycle, which explains the naming of this subgroup.

Words over the generators of $\mathrm{BS}(p, q)$ describe paths in the Cayley graph. Throughout this chapter we regularly use a graphical representation of words, which, while reflecting some of the structure of the Cayley graph, flattens this path to a two-dimensional image with t edges going up, T edges down, and edges labeled a^n going to the right. An example is shown in Figure 2.4. Sometimes horocyclic elements are contracted to points.

Being an HNN extension, $\mathrm{BS}(p, q)$ has a confluent rewriting system (see Section 1.3, with coset representatives $C_1 = \{0, \ldots, p-1\}$, $C_2 = \{0, \ldots, q-1\}$).

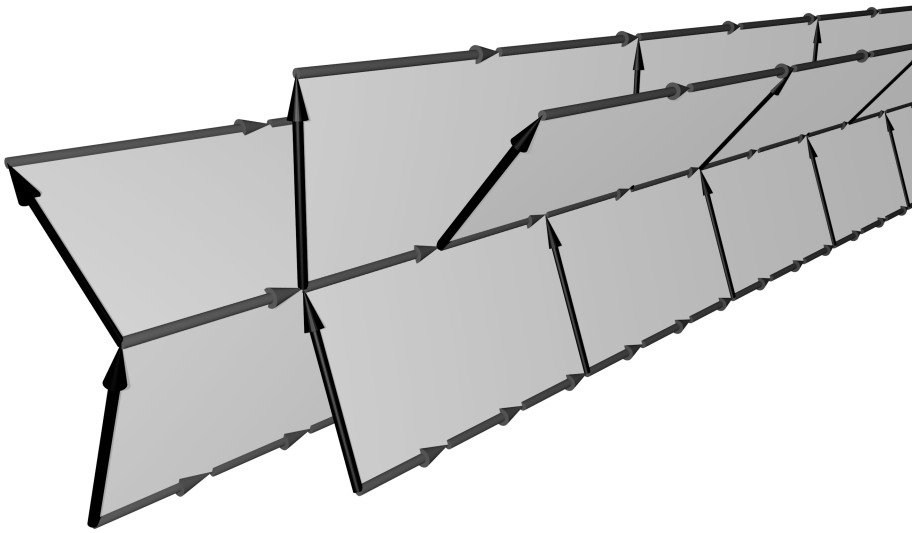


Figure 2.2: A clipping from the Cayley graph of BS$(2,3)$

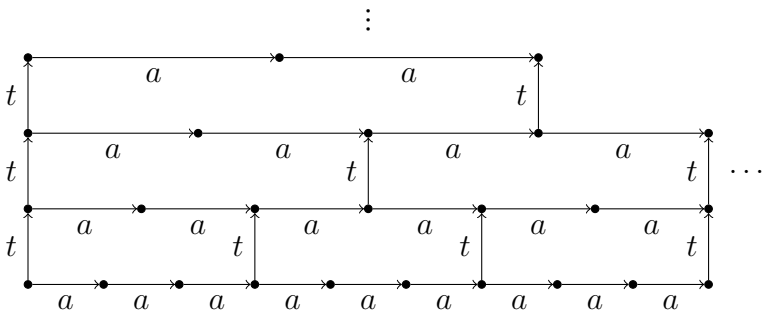


Figure 2.3: The main sheet of BS$(2,3)$

The rules are:

$$
\begin{aligned}
tT, Tt &\longrightarrow 0 \\
\alpha\beta &\longrightarrow (\alpha+\beta) \\
\alpha t\beta &\longrightarrow (\alpha+\mu\cdot q)t\rho && \text{if } \beta = \mu\cdot p+\rho \text{ with } 0\le\rho<p \\
\alpha T\beta &\longrightarrow (\alpha+\mu\cdot p)T\rho && \text{if } \beta = \mu\cdot q+\rho \text{ with } 0\le\rho<q
\end{aligned}
$$

We identify $0 \in \mathbb{Z} \simeq \langle a\rangle$ with the empty word (except for the left-hand side of the second rule which is supposed to be length-reducing). When applying one

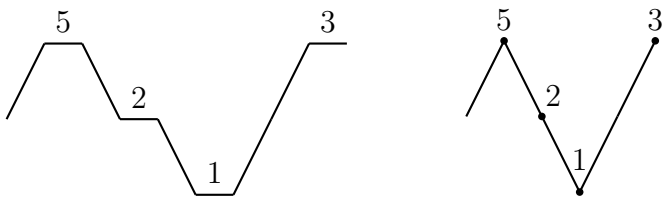


Figure 2.4: Graphical representations of the word $t5T2T1tt3$

of these rules to a Britton-reduced word, the result is also Britton-reduced. Aside from proving Britton's Lemma, this shows that the sequence of the letters t and T in a Britton-reduced word is uniquely determined by the group element. Furthermore, we obtain:

Proposition 2.6. *Let $w_1, w_2 \in \{t, T\} * \mathbb{Z}$ be Britton-reduced words with $w_1 \sim w_2$ in $\mathrm{BS}(p, q)$. Then we get from w_1 to w_2 via finitely many replacements of the form*

$$(\alpha \pm q)t\beta \longleftrightarrow \alpha t(\beta \pm p) \quad \text{and} \quad (\alpha \pm p)T\beta \longleftrightarrow \alpha T(\beta \pm q).$$

□

The graphical representation of words introduced above motivates the following notions.

Definition 2.7. *Let w be a word and $w \overset{*}{\underset{\mathcal{B}}{\Longrightarrow}} \hat{w}$ with $\hat{w}$ Britton-reduced. We call w*

- horocyclic*, if $\hat{w}$ contains neither t nor T,*
- *a* slope*, if $\hat{w}$ contains no t,*
- *a* hill*, if in $\hat{w}$ no T occurs before any t,*
- *a* valley*, if the number of t's equals the number of T's and in every prefix of $\hat{w}$, the number of t's does not exceed the number of T's.*

The height *of a slope or a valley is the number of T's in $\hat{w}$. The height of a hill is the number of t's or T's, whichever is larger.*

By Proposition 2.6, these definitions do not depend on the choice of $\hat{w}$. Examples of Britton-reduced slopes, hills, and valleys are shown in Figure 2.5. In [Eld10], Slopes are called words of type N and hills are called words of type PN.

The following proposition is rather obvious but nevertheless a key observation for the computation of geodesics.

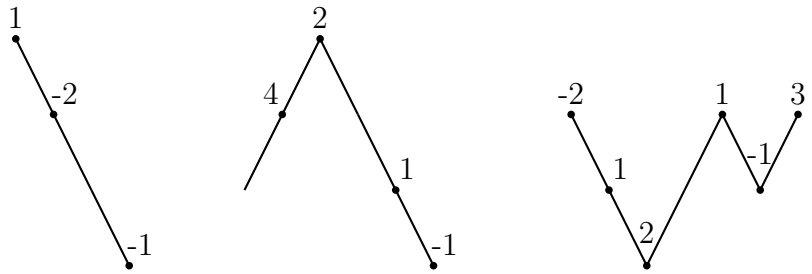


Figure 2.5: A slope, a hill, and a valley

Proposition 2.8.

(i) Let w be geodesic and $w \overset{*}{\underset{\mathcal{B}}{\Longrightarrow}} \hat{w}$ with

$$\hat{w} = \alpha_0 t^{\varepsilon_1} \alpha_1 \dots \alpha_{n-1} t^{\varepsilon_n} \alpha_n$$

Britton-reduced. If we choose some geodesic word $w_i \sim \alpha_i$ for each α_i, then

$$\tilde{w} = w_0 t^{\varepsilon_1} w_1 \dots w_{n-1} t^{\varepsilon_n} w_n$$

is geodesic, too.

(ii) If $w, w_i \in \mathcal{N}$ for some set $\mathcal{N}$ of geodesic normal forms compatible with subwords (i.e., $uv \in \mathcal{N}$ implies $u, v \in \mathcal{N}$; for instance shortlex normal forms), then $\tilde{w} = w$.

Proof. Let $w = v_0 t^{\varepsilon_1} v_1 \dots v_{n-1} t^{\varepsilon_n} v_n$ where $v_i \overset{*}{\underset{\mathcal{B}}{\Longrightarrow}} \alpha_i$ for $0 \le i \le n$. Being subwords of w, all v_i are geodesic (or normal forms). Thus, $|v_i| = |w_i|$ (or $v_i = w_i$) and $|w| = |\tilde{w}|$ (or $w = \tilde{w}$). □

Definition 2.9. *For each word $w = \alpha_0 t^{\varepsilon_1} \alpha_1 \dots \alpha_{n-1} t^{\varepsilon_n} \alpha_n$, we define*

$$\|w\| = n + \sum_{i=0}^{n} [\textit{geodesic length of the horocyclic element } \alpha_i].$$

Note that the triangle inequality holds for $\|\cdot\|$.

By now, we already have a coarse recipe for finding a geodesic of a word w: All Britton-reduced words corresponding to w can be obtained by taking an arbitrary Britton-reduced word and shifting a's around, as in Proposition 2.6. Among these words, look for one that minimizes $\|\cdot\|$. Finally, replace all horocyclic integers in that word by geodesics, which according to Proposition 2.8 results in a geodesic for w.

Following this outline, we first examine the horocyclic subgroup in Section 2.4. After that, in Sections 2.6 and 2.7, we deal with ever larger classes of words, using the notions from Definition 2.7.

Before we start, let us make some general observations on geodesic normal forms for $\mathrm{BS}(p,q)$. Although for some purposes (such as determining the growth of these groups) any set of geodesic normal forms will do, we attempt to get such ones that are as "natural" as possible. By that we mean geodesic normal forms that have a simple and straightforward characterization (for example being lexicographically minimal). If, however, the particular choice of geodesic normal forms is not important, we can use the following reduction to get rid of the case $q < 0$.

Let Ξ be the mapping that sends each word

$$w = \alpha_0 t^{\varepsilon_1} \alpha_1 t^{\varepsilon_2} \alpha_2 t^{\varepsilon_3} \alpha_3 \dots \alpha_{k-1} t^{\varepsilon_k} \alpha_k \qquad (\alpha_i \in \mathbb{Z}, \varepsilon_i \in \{\pm 1\})$$

to

$$\Xi(w) = \alpha_0 t^{\varepsilon_1} (-\alpha_1) t^{\varepsilon_2} \alpha_2 t^{\varepsilon_3} (-\alpha_3) \dots ((-1)^{k-1} \alpha_{k-1}) t^{\varepsilon_k} ((-1)^k \alpha_k).$$

Lemma 2.10. *Ξ gives rise to a bijective mapping $\mathrm{BS}(p,q) \to \mathrm{BS}(p,-q)$ (also denoted Ξ). For horocyclic elements $w \sim \alpha \in \mathbb{Z} \simeq \langle a \rangle \leq \mathrm{BS}(p,q)$, we have $\Xi(w) \sim \alpha \in \mathrm{BS}(p,-q)$. Geodesics are mapped to geodesics.*

Proof. The mapping Ξ commutes with Britton reductions. Thus, for horocyclic words $w \sim \alpha \in \mathrm{BS}(p,q)$, we have $\Xi(w) \sim \alpha \in \mathrm{BS}(p,-q)$.

Let w_1 and w_2 be non-horocyclic words, representing the same element of $\mathrm{BS}(p,q)$, and let $w_i \overset{*}{\underset{\mathcal{B}}{\Longrightarrow}} \hat{w}_i$ with $\hat{w}_i$ Britton-reduced ($i \in \{1,2\}$). Since Ξ is compatible with Britton reductions, we have $\Xi(w_i) \sim \Xi(\hat{w}_i)$. It remains to show that $\Xi(\hat{w}_1) = \Xi(\hat{w}_2)$. By Proposition 2.6, we get from $\hat{w}_1$ to $\hat{w}_2$ by a succession of replacements of the forms $(\alpha \pm q)t\beta \longleftrightarrow \alpha t(\beta \pm p)$ and $(\alpha \pm p)T\beta \longleftrightarrow \alpha T(\beta \pm q)$. Mimicking these replacements on $\Xi(\hat{w}_1)$ with $(\alpha \mp q)t\beta \longleftrightarrow \alpha t(\beta \pm p)$ and $(\alpha \pm p)T\beta \longleftrightarrow \alpha T(\beta \mp q)$ (which are the correct replacements in $\mathrm{BS}(p,-q)$), yields exactly $\Xi(\hat{w}_2)$.

Since Ξ is an involution, the same argument applies to Ξ^{-1}. It is also length-preserving and hence maps geodesics to geodesics. □

Note that Ξ does not respect lexicographic order. For example, the word

$$w = t^6 4TT1T1TT(-2)T(-1) \sim 50$$

in $\mathrm{BS}(2,3)$ is geodesic and minimal with respect to the lexicographic order that we will formally introduce in Section 2.4. Yet,

$$\Xi(w) = t^6 4TT1T(-1)TT2T(-1),$$

which represents 50 in $\mathrm{BS}(2,-3)$, is not lexicographically minimal, since $50 \sim t^6 4TT1T1T1T(-1)T(-1)$ is smaller with respect to that order. However, if $\mathcal{N}$ is a set of geodesic normal forms for $\mathrm{BS}(p,q)$, then $\Xi(\mathcal{N}) = \{\Xi(w) : w \in \mathcal{N}\}$ is a set of geodesic normal forms for $\mathrm{BS}(p,-q)$. This shows:

Corollary 2.11. *The growth series of* $\mathrm{BS}(p,q)$ *and* $\mathrm{BS}(p,-q)$ *coincide. The same is true of the growth series of their horocyclic subgroups.*

In particular, the growth series of $\mathrm{BS}(1,-q)$ *and* $\mathrm{BS}(p,-p)$ *are rational and known [CEG94, EJ92].* □

2.4 Horocyclic Elements and Slopes

The importance of the horocyclic subgroup for the computation of geodesics has been highlighted by Freden et al. who for $p \mid q$ showed the rationality of its growth series and how to compute it [FKS11]. Yet, for our later treatment of the whole group $\mathrm{BS}(p,q)$, we need to know more about geodesic normal forms, which is why we follow the slightly different approach of [DL11].

We postpone the case $p = |q|$ to Section 2.8 and assume until then $0 < p < |q|$. This is a generalization of [DL11], where $q > 0$ is assumed.

Lemma 2.12. *([DL11], Lem. 1) Let w be a geodesic word corresponding to some horocyclic element $\alpha \in \mathbb{Z} \simeq \langle a \rangle \leq \mathrm{BS}(p,q)$, where $0 < p < |q|$. Let k be the number of t's in w.*

(i) w has the form

$$w = \beta_k t \dots \beta_1 t \alpha_0 T \alpha_1 \dots T \alpha_k.$$

(ii) The words

$$w_1 = t^k \alpha_0 T(\alpha_1 + \beta_1) \dots T(\alpha_k + \beta_k) \text{ and}$$
$$w_2 = (\alpha_k + \beta_k) t \dots (\alpha_1 + \beta_1) t \alpha_0 T^k$$

are also geodesic representations of α.

(iii) If $|\alpha| \geq 2|q|$, there are geodesics that contain the letter t (which means we can assume $k \geq 1$ in (ii)).

Proof. We prove (iii) first. Choose $\gamma, \delta \in \mathbb{Z}$ such that $\alpha = \gamma \cdot q + \delta$, $|\gamma| \geq 2$, $|\delta| < |q|$, and $\mathrm{sign}(\delta) = \mathrm{sign}(\alpha)$ (where $\mathrm{sign} : \mathbb{Z} \to \{-1, 0, +1\}$ denotes the sign of an integer). Then $\alpha \sim t(\gamma \cdot p)T\delta$, and the length of this word is at most $|\alpha|$.

For (i) and (ii) we use induction on k. For $k = 0$, there is nothing to do. Let $k \geq 1$. We distinguish two cases to prove (i):

1.) Assume that $w = uv$ for two shorter horocyclic geodesic words u and v. By induction, we can choose geodesics u_2 (ending with T if possible) and v_1 (beginning with t if possible) and replace $w = uv$ with u_2v_1. Having the same length as uv, this word is also geodesic and thus cannot be shortened, so at least one of the subwords u_2, v_1 does not contain any t or T.

2.) If w is not a product of two non-empty horocyclic subwords, then we have either $w = tuT$ or $w = Tut$ for some geodesic word u. In the first case, (i) follows by induction. If $w = Tut$ and $k \geq 2$, using induction, we choose a geodesic $u_1 \sim u$ beginning with the letter t. The word $Tu_1t \sim w$ has the same length as w, but is not geodesic, leading to a contradiction.
If $w = Tut$ and $k = 1$, then, by Britton's Lemma, $u = (\gamma \cdot q)$ for some $\gamma \in \mathbb{Z}$. But $w = T(\gamma \cdot q)t \sim (\gamma \cdot p)$ which is shorter, again contradicting the assumption that w is geodesic.

Finally, (ii) follows from (i), since every subword $t\beta_\ell \dots \alpha_\ell T$ of w is horocyclic and thus commutes with the horocyclic elements $\beta_{\ell+1}$ and $\alpha_{\ell+1}$. □

Lemma 2.12 can be paraphrased in more geometric terms: geodesics of horocyclic words do not branch into different sheets of the Cayley graph. In this sense, for horocyclic elements, the pictures introduced in Section 2.3 are almost exact depictions of the relevant cutouts of the Cayley graph. The word w_1 uses only the main sheet.

In order to define a geodesic normal form, we introduce an order on strings.

Definition 2.13. *The order $<_s$ is defined on the alphabet $\{a, a^{-1}, t, t^{-1}\}$ via $t <_s t^{-1} <_s a <_s a^{-1}$ and extended to the shortlex (sometimes called length-lexicographical) order on $\{a, a^{-1}, t, t^{-1}\}^*$. More precisely, $u <_s v$ if and only if either $|u| < |v|$ or $|u| = |v|$ and u is lexicographically smaller than v (i.e., $u = xby$ and $v = xcz$ with $x, y, z \in \{a, a^{-1}, t, t^{-1}\}^*$, $b, c \in \{a, a^{-1}, t, t^{-1}\}$, and $b <_s c$).*

The shortlex normal form $\mathrm{snf}(g)$ *of an element $g \in \mathrm{BS}(p, q)$ is a word over $\{a, a^{-1}, t, t^{-1}\}$ such that $\mathrm{snf}(g) \sim g$ and among all such words, $\mathrm{snf}(g)$ is minimal with respect to $<_s$.*

Shortlex normal forms are, by definition, geodesic normal forms. Note that the isomorphism $a \mapsto a$, $t \mapsto t^{-1}$ that was used for the assumption $0 < p \leq |q|$ (see Remark 2.2) is not compatible with the shortlex order $<_s$. However, the same algorithms for computing shortlex normal forms still work, if we change $t <_s t^{-1}$ to $t^{-1} <_s t$. This does no harm, as $t <_s t^{-1} <_s a <_s a^{-1}$ was

an arbitrary choice in the first place. For many practical purposes (such as computing growth) any geodesic normal form suffices.

If w is the shortlex normal form of a horocyclic word, then w equals the word w_1 from Lemma 2.12 (ii).

The next proposition generalizes Proposition 3 of [DL11] to negative q.

Proposition 2.14. *Let $w \sim \alpha \in \mathbb{Z} \simeq \langle a \rangle \leq \mathrm{BS}(p,q)$ $(0 < p < |q|)$ be a horocyclic word.*

(i) We can compute in linear time ($\tilde{\mathcal{O}}(|w|^2)$ time on a Turing machine) a number $\ell \in \Theta(\log|\alpha|)$ and a slope u such that

$$\alpha \sim t^{\ell} \underbrace{\alpha_0 t^{-1} \nu_1 \dots \nu_{\ell-1} t^{-1} \nu_{\ell}}_{u}$$

and $|\alpha_0| < 2|q|$ and $|\nu_i| < |q|$ for $1 \leq i \leq \ell$.

(ii) $\mathrm{snf}(\alpha) = t^{\ell}\mathrm{snf}(u)$

Proof. Using Theorem 2.3 we compute the number $\alpha \sim w$ (in binary notation) within the time bound. We proceed by computing a string similar to $\mathrm{snf}(\alpha)$, but not quite geodesic. If $|\alpha| \geq 2|q|$, write $\alpha = \mu \cdot q + \nu$ with $\mu, \nu \in \mathbb{Z}$ and $|\nu| < |q|$, where we choose the remainder ν to have the same sign as α (if ν is non-zero). We have $\alpha \sim t(\mu \cdot p)T\nu$. If $|\mu \cdot p| \geq 2|q|$, we repeat this procedure with $\mu \cdot p$ in place of α. Iterating this, we end up with

$$\alpha \sim v := t^{\ell}\alpha_0 T\nu_1 \dots \nu_{\ell-1} T\nu_{\ell},$$

where

- $\ell \in \Theta(\log|\alpha|)$,
- $\alpha_i := \mu_i \cdot q + \nu_i \sim t^i \alpha_0 T\nu_1 \dots \nu_{i-1} T\nu_i$ for $0 < i \leq \ell$,
  $\alpha_i = \mu_{i+1} \cdot p$ for $0 \leq i < \ell$ and $\alpha_\ell = \alpha$,
- $|\alpha_0| < 2|q|$, and
- $|\nu_i| < |q|$ and either $\nu_i = 0$ or $\mathrm{sign}(\nu_i) = \mathrm{sign}(\alpha_i) = \mathrm{sign}(q) \cdot \mathrm{sign}(\alpha_{i-1})$ for $0 < i \leq \ell$.

This proves (i). The word v is depicted in Figure 2.6.

For (ii) we have to show that $\mathrm{snf}(\alpha)$ starts with t^{ℓ}. Define v_i to be the subword $t^i\alpha_0 T\nu_1 \dots \nu_{i-1} T\nu_i \sim \alpha_i$. We prove by induction on i, that for every α' with $|\alpha'| \geq |\alpha_i|$, the word $\mathrm{snf}(\alpha_i)$ starts with t^i. Since $\alpha = \alpha_\ell$, this completes the proof.

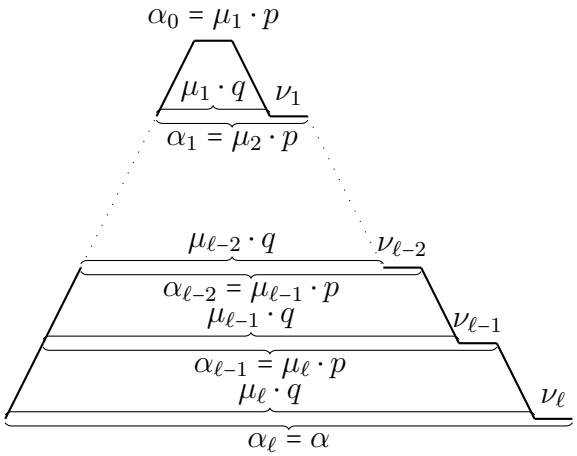


Figure 2.6: The precomputed word from Proposition 2.14

For $i = 0$ there is nothing to do, so let $i \geq 1$. Since $|\alpha'| \geq |\alpha_i| \geq 2|q|$, it follows from Lemma 2.12 that

$$\mathrm{snf}(\alpha') = t^{i'}\gamma_0 T \gamma_1 \dots \gamma_{i'-1} T \gamma_{i'}$$

for some $i' \geq 1$ and some $\gamma_0, \dots, \gamma_{i'} \in \mathbb{Z}$. Since $\mathrm{snf}(\alpha')$ is geodesic, $|\gamma_j| < |q|$ for $1 \leq j \leq i'$, or else the word could be shortened by replacing the subword $\gamma_{j-1} T \gamma_j$ by $(\gamma_{j-1} \pm p) T (\gamma_j \mp q)$. We have $\alpha' = \mu' \cdot q + \gamma_{i'}$ for some $\mu' \in \mathbb{Z}$. From $|\alpha'| \geq |\alpha_i|$, $\alpha_i = \mu_i \cdot q + \nu_i$, and the choice of ν_i we deduce $|\mu'| \geq |\mu_i|$, hence $|\mu' \cdot p| \geq |\alpha_{i-1}|$. By induction, $\mathrm{snf}(\mu' \cdot p)$ begins with t^{i-1}, thus $\mathrm{snf}(\alpha')$ starts with t^i. □

The underlying idea of the next proposition is that the shortlex normal form of a horocyclic element α does not differ much from the word $t^\ell u$ computed in Proposition 2.14. Basically, the only point where we might have gone wrong there, was the choice of ν_i based on the sign instead of the length of the resulting word. In terms of complexity, however, the main work is done. Starting from the already computed slope u, we find the shortlex normal form in linear time (even on a Turing machine) using a dynamic programming approach. We start with a lemma about the shape of shortlex normal forms of slopes.

Lemma 2.15. *Let u be a slope of height ℓ in $\mathrm{BS}(p, q)$ with $0 < p < |q|$. Then*

$$\mathrm{snf}(u) = t^k \beta_{k+\ell} T \beta_{k+\ell-1} \dots \beta_1 T \beta_0$$

with $|\beta_{k+\ell}| < 2|q|$ and $|\beta_i| < |q|$ for $0 \leq i < k + \ell$.

Proof. Performing Britton reductions on $\mathrm{snf}(u)$, we get a Britton-reduced slope $w = \gamma T \beta_{\ell-1} \dots \beta_1 T \beta_0$ with the same height as u. By Proposition 2.8, we have

$$\mathrm{snf}(u) = \mathrm{snf}(\gamma) T \,\mathrm{snf}(\beta_{\ell-1}) \dots \mathrm{snf}(\beta_1) T \,\mathrm{snf}(\beta_0).$$

For $i < \ell$, $\mathrm{snf}(\beta_i)$ contains no t, or else, by Lemma 2.12 (ii), it would start with a t, making Tt a subword of $\mathrm{snf}(u)$. Hence, $\mathrm{snf}(\beta_i) = \beta_i$. Furthermore, if $|\beta_i| \geq |q|$, the subword $\beta_{i+1} T \beta_i$ of $\mathrm{snf}(u)$ could be replaced by the shorter word $(\beta_{i+1} \pm p) T (\beta_i \mp q)$. Thus, $|\beta_i| < q$ for $i < \ell$. The rest follows by applying Lemma 2.12 (ii) to γ. □

Proposition 2.16. *([DL11], Prop. 4 and Cor. 7) Let $0 < p < |q|$.*

(i) There is a linear time algorithm (on a Turing machine) which, given a slope

$$u = \alpha_0 T \alpha_1 \dots \alpha_{\ell-1} T \alpha_\ell$$

with $|\alpha_0| < 2|q|$ and $|\alpha_i| < |q|$ for $1 \leq i \leq \ell$, computes $\mathrm{snf}(u)$.

(ii) The set of shortlex normal forms of slopes is regular.

Proof. We give the proof in two parts. First, we describe a polynomial (but not linear) time algorithm for finding $\mathrm{snf}(u)$. In the second part, we improve this algorithm in such a way that it needs only a constant amount of memory which leads us to (i) and (ii). A reader who is only interested in polynomial time results, can skip the second part.

Let r be the smallest positive integer such that

$$r \geq p \cdot \frac{r + |q| - 1}{|q|} + |q| - 1.$$

For each $0 \leq i \leq \ell$ and $\gamma \in \mathbb{Z}$ we define the word

$$u(i, \gamma) := \alpha_0 T \alpha_1 \dots \alpha_{i-1} T \gamma.$$

We will show how to compute $\mathrm{snf}(u(i, \gamma))$ for $|\gamma| \leq r$, assuming we already know all the words $\mathrm{snf}(u(i-1, \gamma'))$ with $|\gamma'| \leq r$. Since $u = u(\ell, \alpha_\ell)$ and $|\alpha_\ell| \leq |q| - 1 \leq r$, we obtain $\mathrm{snf}(u)$ after the ℓ-th iteration.

Precompute $\mathrm{snf}(u(0, \gamma)) = \mathrm{snf}(\gamma)$ for $|\gamma| \leq r + |q|$. This can be done beforehand and the result hard-wired into the algorithm. Now, let $i \geq 1$. According to Lemma 2.15, the word $\mathrm{snf}(u(i, \gamma))$ ends with $T\rho$ for some $\rho \in \mathbb{Z}$ with $|\rho| < |q|$ and, by Proposition 2.6, $\rho \equiv \gamma \bmod q$. Thus, we have

$$\mathrm{snf}(u(i, \gamma)) = \min \Big\{ \mathrm{snf}(u(i-1, \alpha_{i-1} + \rho')) T \rho \quad : \quad \rho \equiv \gamma \bmod q, \; |\rho| < |q|, \; \rho' = p \cdot \frac{\gamma - \rho}{q} \Big\},$$

where the minimum is taken with respect to the shortlex order $<_s$. Note that there are at most two choices for ρ. Because of

$$|\rho' + \alpha_{i-1}| \leq p \cdot \frac{|\gamma| + |\rho|}{|q|} + |\alpha_{i-1}| \leq p \cdot \frac{r + |q| - 1}{|q|} + |\alpha_{i-1}| \leq \begin{cases} r & \text{if } i > 1, \\ r + |q| & \text{if } i = 1, \end{cases}$$

the word $\mathrm{snf}(u(i-1, \alpha_{i-1} + \rho'))$ is among those computed in the previous iteration. This completes the description of the algorithm. A demonstration of the algorithm can be found in Example 2.18.

The algorithm described above needs quadratic time on a Turing machine. In each of the linearly many iterations, we have to compute a constant number of geodesics, each of which has linear length. For each computation we have to compare two words. All the necessary arithmetic operations only concern integers bounded by a constant.

In order to achieve linear time, we observe that

$$\big|\ |\mathrm{snf}(u(i,\gamma))| - |\mathrm{snf}(u(i,\gamma'))|\ \big| \leq 2r$$

for $|\gamma|, |\gamma'| \leq r$. This is true in general: multiplying an element by a generator can change the geodesic length at most by one. If we split each word $\mathrm{snf}(u(i,\gamma))$ into

$$\mathrm{snf}(u(i,\gamma)) = v_{i,\gamma}\, w_{i,\gamma}$$

with $|v_{i,\gamma}| = \min\{|\mathrm{snf}(u(i,\delta))| : |\delta| \leq r\}$, we have $|w_{i,\gamma}| \leq 2r$. Now we modify the algorithm in such a way that after the i-th iteration, instead of all the words $\mathrm{snf}(u(i,\gamma))$ we only store the following information:

- $w_{i,\gamma}$ for $|\gamma| \leq r$ and
- the lexicographical order of the $v_{i,\gamma}$ (this coincides with the shortlex order since they all have the same length).

This information only takes a constant amount of space. If, during the minimum search, we have to compare $\mathrm{snf}(u(i-1,\gamma))T\rho$ and $\mathrm{snf}(u(i-1,\delta))T\tau$, we first compare their lengths by looking at the lengths of $w_{i-1,\gamma}T\rho$ and $w_{i-1,\delta}T\tau$. If they are equal, we check the lexicographic order of $v_{i-1,\gamma}$ and $v_{i-1,\delta}$. If $v_{i-1,\gamma} = v_{i-1,\delta}$, we finally compare $w_{i-1,\gamma}T\rho$ and $w_{i-1,\delta}T\tau$. All of this takes constant time and uses only the available data.

The length of the shortest word among the $\mathrm{snf}(u(i,\gamma))$ ($|\gamma| \leq r$) strictly increases in each iteration, since a factor $T\rho$ is appended. After each iteration, we cut prefixes of the same length off the computed words in order to get new suffixes $w_{i,\gamma}$ with length at most $2r$. The prefixes that have been

cut off are used to update the lexicographic order. In order to permit reconstruction of the word afterwards, we print them out, along with the positions $\rho' + \alpha_{i-1}$ where the minimum was attained. In other words, the output of the algorithm consists of a sequence $(c_i)_i$ of tuples $c_i = (x^i_\gamma, \tau^i_\gamma)_\gamma$, such that $\mathrm{snf}(u(i,\gamma)) = \mathrm{snf}(u(i-1,\tau^i_\gamma))x^i_\gamma$ for each $|\gamma| \le r$. In the last tuple c_ℓ, we write all the remaining $w_{\ell,\gamma}$.

An application of this algorithm to a concrete word can be found in the second part of Example 2.18. There we visualize the output as a matrix, where the columns are the tuples c_i. The result $\mathrm{snf}(u) = \mathrm{snf}(u(\ell,\alpha_\ell))$ can be recovered from right to left by starting at $x^\ell_{\alpha_\ell}$ and following the pointer $\tau^\ell_{\alpha_\ell}$ to the appropriate entry in the previous column and so on.

Having reduced the space needed by the algorithm to a constant, every iteration only takes constant time. This proves (i). If, as in (ii), the task is only to recognize shortlex normal forms instead of computing them, we can omit the second step of going backwards through the output. Instead, we monitor which of the potential outputs correspond to the input string. This one-pass Turing machine translates to a finite automaton. □

Combining Propositions 2.14 and 2.16 (i), we get:

Theorem 2.17. *(cf. [DL11], Cor. 5) Let $0 < p < |q|$. The shortlex normal form of a horocyclic word in $\mathrm{BS}(p,q)$ can be computed in linear time ($\tilde{\mathcal{O}}(n^2)$ time on a Turing machine).* □

Example 2.18. *In order to illustrate the quite technical constructions leading to Theorem 2.17, we apply them to an example. Let $p = 2$, $q = 3$ and $\alpha = 22$. The algorithm from Proposition 2.14 yields*

$$22 \sim t^3 \underbrace{4T2T2T1}_{u}.$$

We have $r = 10$. Figure 2.7 shows the result of the simpler version of the algorithm from Proposition 2.16. The first column contains the precomputed shortlex normal forms of $u(0,\gamma) = \gamma$ for $|\gamma| \le r + |q| = 13$. As an example of how to compute the other entries, let us take a closer look at $\mathrm{snf}(u(2,4)) = \mathrm{snf}(4T2T4)$. The choices for $\rho \equiv 4 \bmod 3$ are 1 and -2, resulting in

$$\begin{aligned}\mathrm{snf}(4T2T4) &= \min\{\mathrm{snf}(4T4)T1,\ \mathrm{snf}(4T6)T(-2)\}\\ &= \min\{4TT1T1,\ 4T2TT(-2)\}\\ &= 4TT1T1.\end{aligned}$$

Having computed all the other entries of the table in a similar way, we get $\mathrm{snf}(u) = t4TT1T(-1)T1$ and the final result:

$$\mathrm{snf}(22) = t^3\mathrm{snf}(u) = tttt4TT1T(-1)T1 = ttttaaaaTTaTATa$$

γ	$\mathrm{snf}(u(0,\gamma))$	$\mathrm{snf}(u(1,\gamma))$	$\mathrm{snf}(u(2,\gamma))$	$\mathrm{snf}(u(3,\gamma))$
−13	$tt(-4)T(-2)T(-1)$			
−12	$tt(-4)T(-2)T$			
−11	$tt(-4)TT(-2)$			
−10	$tt(-4)TT(-1)$	$(-2)T(-1)$	$TT2$	$2T1TT2$
−9	$tt(-4)TT$	$(-2)T$	$T2T$	$2T1T2T$
−8	$t(-4)T(-2)$	$T(-2)$	$T2T1$	$2T1T2T1$
−7	$t(-4)T(-1)$	$T(-1)$	$T2T2$	$4TT1T(-1)$
−6	$t(-4)T$	T	$2T1T$	$4TT1T$
−5	−5	$T1$	$2T1T1$	$4TT1T1$
−4	−4	$T2$	$2T1T2$	$4TT1T2$
−3	−3	$2T$	$4TT$	$4T2TT$
−2	−2	$2T1$	$4TT1$	$4T2TT1$
−1	−1	$2T2$	$4TT2$	$4T2TT2$
0		$4T$	$4T2T$	$t4TT1T(-1)T$
1	1	$4T1$	$4T2T1$	**t4TT1T(−1)T1**
2	2	$4T2$	$t4TT1T(-1)$	$t4TT1T1T(-1)$
3	3	$t4TT$	$t4TT1T$	$t4TT1T1T$
4	4	$t4TT1$	$t4TT1T1$	$t4TT1T1T1$
5	5	$t4TT2$	$t4TT1T2$	$t4T2TTT(-1)$
6	$t4T$	$t4T2T$	$t4T2TT$	$t4T2TTT$
7	$t4T1$	$t4T2T1$	$t4T2TT1$	$t4T2TTT1$
8	$t4T2$	$tt4TT1T(-1)$	$t4T2TT2$	$t4T2TTT2$
9	$tt4TT$	$tt4TT1T$	$tt4TT1T(-1)T$	$t4T2TT2T$
10	$tt4TT1$	$tt4TT1T1$	$tt4TT1T(-1)T1$	$t4T2TT2T1$
11	$tt4TT2$			
12	$tt4T2T$			
13	$tt4T2T1$			

Figure 2.7: Results of the algorithm for the slope $u = 4T2T2T1$

The output of the linear time version of the algorithm is shown in Figure 2.8. After the words $\mathrm{snf}(u(1,\gamma))$ *are computed in the same way as before, the first letter of each word is cut off and printed out, giving the column* $i = 1$*. After the second iteration, a further three letters are cut off and printed, along with the indices indicating where the minimum was attained. For example, the tuple* $(3,2)$ *at* $i = 2, \gamma = 1$ *tells us, that*

$$\mathrm{snf}(u(2,1)) = \mathrm{snf}(4T2T1) = [\textit{entry at } i = 1, \gamma = 2]\, 3\, w_{2,1} = 4\, w_{2,1}.$$

γ $i=1$	$i=2$	$i=3$
−10 (−1)	$(T2, -6)$	$(TT2, -6)$
−9 (−1)	$(2T, -4)$	$(T2T, -4)$
−8 T	$(2T, -4)$	$(T2T1, -4)$
−7 T	$(2T, -4)$	$(TT1T(-1), -2)$
−6 T	$(1T1, -2)$	$(TT1T, -2)$
−5 T	$(1T1, -2)$	$(TT1T1, -2)$
−4 T	$(1T1, -2)$	$(TT1T2, -2)$
−3 1	$(3, 0)$	$(T2TT, 0)$
−2 1	$(3, 0)$	$(T2TT1, 0)$
−1 1	$(3, 0)$	$(T2TT2, 0)$
0 1	$(3, 2)$	$(1TT1T(-1)T, 2)$
1 1	$(3, 2)$	$(\mathbf{1TT1T(-1)T1}, \mathbf{2})$
2 1	$(\mathbf{3}, \mathbf{4})$	$(1TT1T1T(-1), 4)$
3 t	$(3, 4)$	$(1TT1T1T, 4)$
4 **t**	$(3, 4)$	$(1TT1T1T1, 4)$
5 t	$(3, 4)$	$(1T2TTT(-1), 6)$
6 t	$(3, 6)$	$(1T2TTT, 6)$
7 t	$(3, 6)$	$(1T2TTT1, 6)$
8 t	$(3, 6)$	$(1T2TTT2, 6)$
9 t	$(t2, 8)$	$(1T2TT2T, 8)$
10 t	$(t2, 8)$	$(1T2TT2T1, 8)$

Figure 2.8: Output of the linear time algorithm for the slope $u = 4T2T2T1$

Corollary 2.19. *([DL11], Cor. 7) If $0 < p < |q|$ and p divides q, then the set of shortlex normal forms of horocyclic elements in $\mathrm{BS}(p, q)$ is a deterministic linear context-free language. The growth series of the horocyclic subgroup of $\mathrm{BS}(p, q)$ is rational.*

Proof. The language of shortlex normal forms of horocyclic elements is the intersection of the following sets:

1.) $\{w : \text{the number of } t\text{'s in } w \text{ equals the number of } T\text{'s}\}$

2.) $\{t^k \alpha_0 T \alpha_1 \dots T \alpha_k : p \text{ divides } \alpha_i \text{ for } 0 \le i < k\}$

3.) $\{t^k \mathrm{snf}(u) : u \text{ is a slope}\}$

The first language is recognized by a deterministic one-turn pushdown automaton (see [Har78], Sect. 5.7). The second one is regular and ensures

horocyclicity. This is where we need the assumption $p \mid q$. The regularity of the third language was shown in Proposition 2.16 (ii).

Having an unambiguous linear context-free grammar for a set of geodesic normal forms, the rationality of the growth series follows from the Chomsky-Schützenberger theorem [CS63, Kui70]. □

Corollary 2.19 was first proved in [FKS11]. However, Proposition 2.16 is stronger than the corresponding result there, since it also applies to $p \nmid q$. We need the assumption $p \mid q$ only to check horocyclicity. For this, however, $p \mid q$ is necessary, see Theorem 7.2 in [FKS11]. In addition to that, we have given a linear time algorithm for computing geodesic normal forms of horocyclic elements. For $p = 1$, this has been done before in [Eld10]. Corollary 6.3 in [FKS11] gives a quadratic time acceptor for another set of geodesic normal forms for the horocyclic subgroup of $\mathrm{BS}(p, q)$.

2.4.1 Horocyclic Growth in $\mathrm{BS}(2, 3)$

The first case where Corollary 2.19 does not apply is $p = 2, q = 3$. It is still unknown whether the growth series of the horocyclic subgroup of $\mathrm{BS}(2, 3)$ is rational or not. Usually, irrationality is conjectured [EJ92, FKS11]. In [FKS11], Freden et al. base this conjecture on some undisclosed experimentation. Using the algorithm developed in Section 2.4, we show:

Result 2.20. *If the growth series of the horocyclic subgroup of* $\mathrm{BS}(2, 3)$ *is rational, its total degree is at least* 70*.*

The total degree of a rational function r is

$$\mathrm{tdeg}(r) = \min\{\max\{\deg f, \deg g\} \,:\, f, g \text{ polynomials}, r = f/g\}.$$

In order to get this result, we use two lemmas:

Lemma 2.21. *If some growth series* $S(z) = \sum_{n\geq 0} B_n z^n$ *is rational and its total degree is smaller than* k*, then the following system of linear equations has a solution:*

$$\begin{pmatrix} B_k & B_{k-1} & \dots & B_2 & B_1 \\ B_{k+1} & B_k & & & B_2 \\ \vdots & & \ddots & & \vdots \\ B_{2k-2} & & & B_k & B_{k-1} \\ B_{2k-1} & B_{2k-2} & \dots & B_{k+1} & B_k \end{pmatrix} \cdot \begin{pmatrix} 1 \\ q_1 \\ \vdots \\ q_{k-2} \\ q_{k-1} \end{pmatrix} = 0 \tag{2.1}$$

Proof. Assume that $S(z) = {}^{p(z)}\!/_{q(z)}$ for two polynomials p, q with $\deg p, \deg q < k$. We write

$$p(z) = \sum_{i=0}^{k-1} p_i z^i \qquad \text{and} \qquad q(z) = 1 + \sum_{i=1}^{k-1} q_i z^i.$$

The equation $S(z) = {}^{p(z)}\!/_{q(z)}$ can be rewritten as $S(z) \cdot q(z) = p(z)$. Comparing the coefficients of $z^k, \ldots, z^{2k-1}$ leads to Equation 2.1. □

Lemma 2.22. *Let w be a geodesic word representing the horocyclic element $\alpha > 0$ in* BS$(2,3)$. *Then*

$$\alpha \leq 8 \cdot (3/2)^{\lfloor |w|/2 \rfloor - 1}.$$

Proof. According to Lemma 2.12, we have $w = \beta_k t \ldots \beta_1 t \alpha_0 t^{-1} \alpha_1 \ldots t^{-1} \alpha_k$ with $\alpha_0 \geq 2$, so the number of t's in w is bounded by $k = \lfloor |w| - 2/2 \rfloor$. Furthermore, we have $\alpha_0 \leq 4$ (since $\alpha_0 = 6$ could be replaced by $t4t^{-1}$) and $\alpha_i < 3$ for $i > 0$. Thus,

$$\alpha \leq \left(\ldots \left(\left(4 \cdot \frac{3}{2} + 2 \right) \cdot \frac{3}{2} + 2 \right) \cdot \frac{3}{2} + 2 \ldots \right) \cdot \frac{3}{2} + 2 \leq 8 \cdot (3/2)^k.$$

□

Algorithm 5.1 in the appendix is an implementation (for $p = 2, q = 3$) of the procedure developed in Section 2.4. It computes the geodesic lengths of horocyclic elements up to $8 \cdot (3/2)^{k-2}$ for $k = 70$ and then counts how many of each length between 0 and $2k - 1 = 139$ occur. Since the geodesic length of the horocyclic element α is the same as that of $-\alpha$ (replace a by a^{-1} in the geodesic), only geodesics of positive integers need to be computed, and we get the coefficients B_n $(n \geq 1)$ of the growth series by doubling these numbers. Finally, the unsolvability of the equation in Lemma 2.21 is verified using the computer algebra system Maple.

Although Result 2.20 by no means proves irrationality, it might be considered (albeit weak) evidence, particularly by contrast with the comparatively small degrees of the horocyclic growth series of other Baumslag-Solitar groups:

Result 2.23.

group	*horocyclic growth series*	*total degree*	
BS$(1,3)$	$\dfrac{1 + 2z + z^2 - 2z^3 - 2z^4}{1 - z^2 - 2z^3}$	4	
BS$(2,4)$	$\dfrac{1 + 2z + z^2 - 2z^7 - 2z^8}{1 - z^2 - 2z^6}$	8	*(cf. [FKS11], Sect. 5)*
BS(q,q)	$\dfrac{1+z}{1-z}$	1	

2.5 Peak Normal Forms

In this section we introduce a geodesic normal form for all elements of the group $\mathrm{BS}(p,q)$. It differs from the shortlex normal form used for horocyclic elements in that it is more symmetric. For example, take a Britton-reduced hill

$$w = \underbrace{\alpha_k t \alpha_{k-1} \dots \alpha_1 t}_{u} \, \alpha_0 \, \underbrace{T\beta_1 \dots \beta_{\ell-1} T \beta_\ell}_{v}.$$

For a reasonably symmetric normal form of w, one would expect that the subwords corresponding to u and v^{-1} are both normal forms of the same type (for example shortlex normal forms). This is the case for the Britton peak normal form which we are going to introduce shortly. We start by defining the peak of a Britton-reduced word.

Definition 2.24. *Let $w = \alpha_0 t^{\varepsilon_1} \alpha_1 \dots \alpha_{n-1} t^{\varepsilon_n} \alpha_n$ be a Britton-reduced word. We define the* level *of a position $0 \le i \le n$ by*

$$\mathrm{lev}(0) = 0 \qquad \textit{and} \qquad \mathrm{lev}(i+1) = \mathrm{lev}(i) + \varepsilon_{i+1}.$$

The peak *of w is the maximum (rightmost) position among those of maximum level.*

The motivation for the notion of level is obvious from the graphical representation. An example can be found in Figure 2.9. The peak is actually a property of the group element rather than the particular word w, since the peaks of any two Britton-reduced words $w_1 \sim w_2$ coincide. The choice of the peak as the rightmost among all potential peaks is somewhat arbitrary, yet a simple way of avoiding ambiguity.

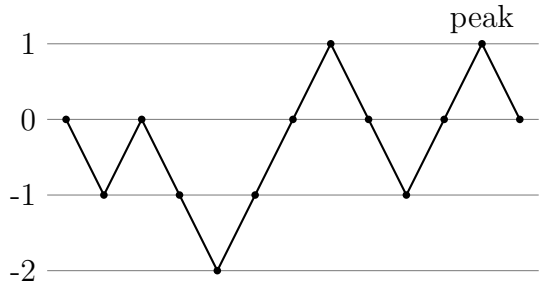


Figure 2.9: Levels and peak of a Britton-reduced word

In the next definition, we use the norm

$$\|\alpha_0 t^{\varepsilon_1} \alpha_1 \dots \alpha_{n-1} t^{\varepsilon_n} \alpha_n\| = n + \sum_{i=0}^{n} |\mathrm{snf}(\alpha_i)|$$

from Definition 2.9.

Definition 2.25. *Let* $w_1 = \alpha_0 t^{\varepsilon_1} \alpha_1 \dots \alpha_{k-1} t^{\varepsilon_k} \alpha_k$ *and* $w_2 = \beta_0 t^{\varepsilon_1} \beta_1 \dots \beta_{\ell-1} t^{\varepsilon_\ell} \beta_\ell$ *be Britton-reduced words with* i *and* j *being the peaks of* w_1 *and* w_2*, respectively. Let* $w_1 = u_1 \alpha_i v_1$ *and* $w_2 = u_2 \beta_j v_2$ *be the factorizations of* w_1 *and* w_2 *into the parts to the left and to the right of their peaks. We define* $w_1 <_p w_2$ *if and only if*

(i) $\|w_1\| < \|w_2\|$ *or*

(ii) $\|w_1\| = \|w_2\|$ *and* $u_1 <_s u_2$ *or*

(iii) $\|w_1\| = \|w_2\|$ *and* $u_1 = u_2$ *and* $v_1^{-1} <_s v_2^{-1}$ *(v^{-1} denotes the word v read from right to left with every letter replaced by its inverse) or*

(iv) $\|w_1\| = \|w_2\|$ *and* $u_1 = u_2$ *and* $v_1 = v_2$ *and* $\alpha_i <_s \beta_j$*.*

The Britton peak normal form *of* w *(*$\mathrm{Bpnf}(w)$*) is the minimal word with respect to* $<_p$ *among all Britton-reduced words* w' *with* $w' \sim w$*.*

If $\mathrm{Bpnf}(w) = \alpha_0 t^{\varepsilon_1} \alpha_1 \dots \alpha_{k-1} t^{\varepsilon_k} \alpha_k$*, then*

$$\mathrm{pnf}(w) := \mathrm{snf}(\alpha_0) t^{\varepsilon_1} \mathrm{snf}(\alpha_1) \dots \mathrm{snf}(\alpha_{k-1}) t^{\varepsilon_k} \mathrm{snf}(\alpha_k)$$

is called the peak normal form *of* w*.*

Both Britton peak normal forms and peak normal forms are normal forms in the sense that there is exactly one such word for each group element. If Condition (iv) is omitted, $<_p$ becomes a partial order on Britton-reduced words, but it remains a total order inside the equivalence classes given by the group relator, thus defining the same normal forms. By definition, peak normal forms are geodesics (cf. Proposition 2.8).

If $w = \alpha_0 t^{\varepsilon_1} \alpha_1 \dots \alpha_{n-1} t^{\varepsilon_n} \alpha_n$ is a Britton peak normal form, then $\mathrm{snf}(\alpha_i) = \alpha_i$ whenever $\varepsilon_i = -1$ or $\varepsilon_{i+1} = +1$. In other words, apart from horocyclic elements on "hilltops", Britton peak normal forms are already geodesics.

Since shortlex normal forms of horocyclic element can be computed in linear time, our real concern is to find Britton peak normal forms. We use $\|w\|$ as a measure of the length of an input word $w \in \{t, T\} * \mathbb{Z}$, since for horocyclic elements $\alpha \in \mathbb{Z}$, $|\mathrm{snf}(\alpha)| \in \Theta(\log|\alpha|)$. Hence $\|w\|$ is (asymptotically) the length of the word w with the integers written in binary.

2.6 Hills and Difficult Words

When computing geodesics for slopes, the idea was roughly to go from bottom to top and propagate upwards as many a's (or a^{-1}'s) as possible. Obviously,

things are not as easy when we have a word like $\alpha T \beta t \gamma$ with two possibilities of "going up" from the position of β. This motivates the following definition.

Definition 2.26. *A Britton-reduced word* $w = \alpha_0 t^{\varepsilon_1} \alpha_1 \dots \alpha_{n-1} t^{\varepsilon_n} \alpha_n$ *is called* difficult*, if* $\varepsilon_1 = -1$ *and* $\varepsilon_n = +1$.

For every Britton-reduced word w, there is a unique factorization

$$w = \alpha_k t \alpha_{k-1} \dots \alpha_1 t D T \beta_1 \dots \beta_{\ell-1} T \beta_\ell,$$

such that D is either difficult or horocyclic. In the latter case, w is a hill. An example is shown in Figure 2.10.

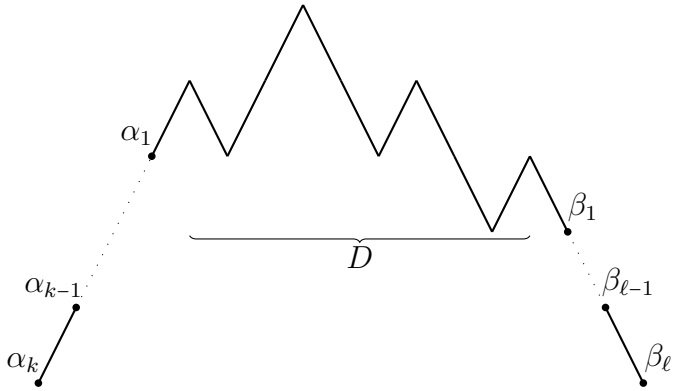


Figure 2.10: Factorization of a word

For the next proposition, let $DT(d)$ be an upper bound on the time needed for computing the Britton peak normal form of any difficult word D with $\|D\| \leq d$. Later in this chapter we will prove $DT(d) \in \mathcal{O}(d^3)$ for the case $p \mid q$.

Proposition 2.27. *([DL11], Thm. 9)*
Let $w = \alpha_k t \alpha_{k-1} \dots \alpha_1 t D T \beta_1 \dots \beta_{\ell-1} T \beta_\ell$ *be a Britton-reduced word, where* D *is either horocyclic or difficult.*

1.) *If* D *is horocyclic, then* $\mathrm{Bpnf}(w)$ *can be computed in* $\mathcal{O}(\|w\|)$ *time.*

2.) *If* D *is difficult, then* $\mathrm{Bpnf}(w)$ *can be computed in*

$$\mathcal{O}(DT(\|D\| + C \cdot (\max\{\|\alpha_i\|, \|\beta_j\|\} + 1)) + \|w\|^2)$$

time, for some constant C *depending only on* p *and* q.

If DT is polynomial, for example $DT(d) = d^m$ $(m \geq 2)$, the time bound from this proposition simplifies to $\mathcal{O}(\|w\|^m)$.

To a large degree, the proof resembles that of Proposition 2.16 (the analogous result for slopes). For this reason, we skip some of the details. Figure 2.11 shows the word w.

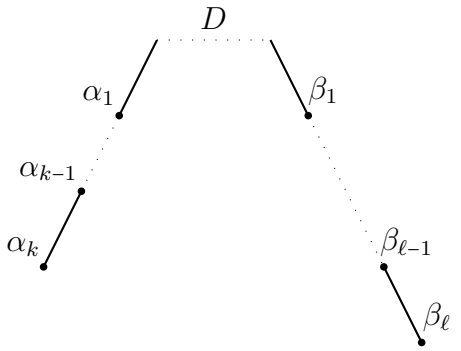


Figure 2.11: Input for the algorithm from Proposition 2.27

Proof. After some replacements of the form $\gamma t\delta \longrightarrow (\gamma \mp q)t(\delta \pm p)$ or $\gamma T\delta \longrightarrow (\gamma \pm p)T(\delta \mp q)$, we may assume that $0 \leq |\alpha_i|, |\beta_j| < |q|$ for $1 \leq i \leq k$, $1 \leq j \leq \ell$. This preprocessing takes $\mathcal{O}(\|w\|)$ time. Due to these replacements, $\|D\|$ might increase by up to $\mathcal{O}(\max\{\|\alpha_i\|, \|\beta_j\|\})$. In case 1.) we have a linear time algorithm for D, so this does not matter. For 2.), it suffices to prove the time bound $\mathcal{O}(DT(\|D\| + C) + \|w\|^2)$ for the new word.

As we did for slopes, we define subwords of w, but this time with four parameters:

$$w(i, j, \gamma, \delta) = \gamma t \alpha_{i-1} \dots \alpha_1 t D T \beta_1 \dots \beta_{j-1} T \delta$$

We prove by induction on $i + j$ that $\mathrm{pnf}(w(i, j, \gamma, \delta))$ can be computed for all $|\gamma|, |\delta| \leq r$ within in the time bound. As before, r is the smallest natural number such that $r \geq p \cdot \frac{r+|q|-1}{q} + |q| - 1$.

For $i + j = 0$, we need to compute the Britton peak normal form of $\gamma D \delta$. If D is horocyclic, nothing needs to be done. If D is difficult, we need $DT(\|\gamma D \delta\|) \leq DT(\|D\| + \|2r\|)$ time.

Let $i + j \geq 1$. Note that the peak is inside D. If $i \geq 1$ ($j \geq 1$ is similar), we compute $\mathrm{Bpnf}(w(i, j, \gamma, \delta))$ using the words $\mathrm{Bpnf}(w(i - 1, j, \tau, \delta))$ with $|\tau| \leq r$. Since $\mathrm{Bpnf}(w(i, j, \gamma, \delta))$ starts with ρt for some $\rho \equiv \gamma \bmod q$ with

$|\rho| < |q|$, we have

$$\mathrm{Bpnf}(w(i,j,\gamma,\delta)) = \min\{\rho t\,\mathrm{Bpnf}(w(i-1,j,\rho'+\alpha_{i-1},\delta)) \quad : \quad \rho \equiv \gamma \bmod q,\ |\rho| < |q|,\ \rho' = p\cdot\frac{\gamma-\rho}{q}\},$$

where min refers to the order $<_p$. The choice of r ensures $|\rho' + \alpha_{i-1}| \le r$. Each comparison takes $\mathcal{O}(\|w\|)$ time, resulting in the claimed time bound for the entire algorithm.

Let D be horocyclic. In order to eliminate the square from the term $\|w\|^2$, we optimize the algorithm using a similar idea as in the procedure for slopes. To make things easier, we assume that in the algorithm described above, we do all the steps on the left-hand side first (in other words, compute $\mathrm{Bpnf}(w(k,0,\alpha_k,\delta))$) and only after that we process the right-hand side.

The first part can be done independently for each δ. We store the differences of the norms of the words $\mathrm{Bpnf}(w(i,0,\gamma,\delta))$ (for $|\gamma| \le r$), their order with respect to $<_p$, and the differences of the lengths of their subwords left of the peak. Using only this constant amount of data, we can compare $\rho t\,\mathrm{Bpnf}(w(i,0,\gamma,\delta))$ and $\rho' t\,\mathrm{Bpnf}(w(i,0,\gamma',\delta))$ in constant time: first, we look at the difference between $\|\rho\| + \|\mathrm{Bpnf}(w(i,0,\gamma,\delta))\|$ and $\|\rho'\| + \|\mathrm{Bpnf}(w(i,0,\gamma',\delta))\|$, second, at the length differences of the subwords left of the peak, then we compare ρt and $\rho' t$ using $<_p$, and finally we look at the $<_p$ order of $\mathrm{Bpnf}(w(i,0,\gamma,\delta))$ and $\mathrm{Bpnf}(w(i,0,\gamma',\delta))$.

Having found $\mathrm{Bpnf}(w(i+1,0,\gamma,\delta))$ for all $|\gamma| \le r$, we update the information in constant time. For $i = 0$, the necessary data can be computed in $\mathcal{O}(\|w\|)$ time. A similar approach works for the right-hand side, where we compare words like $\mathrm{Bpnf}(w(k,j,\alpha_k,\delta))T\rho$ and $\mathrm{Bpnf}(w(k,j,\alpha_k,\delta'))T\rho'$. □

Remark 2.28. *For horocyclic elements we have shown that the linear time (and constant space) algorithm can be turned into a deterministic one-turn pushdown automaton, thus proving rationality of the horocyclic growth series. Something similar can be done for hills. However, since this is rather technical and we never use that statement in this thesis, we leave the details to the interested reader.*

So far, we have solved the problem of computing peak normal forms for hills and reduced the general task to the case of difficult words. This is as far as we get for arbitrary p and q. In the next section we restrict ourselves to the case where p divides q and solve the remaining problem under this assumption.

2.7 Geodesics in $\mathrm{BS}(p,q)$ with $p \mid q$

Throughout this section, we assume that p divides q. We deviate slightly from [DL11] in favor of a clearer exposition. We start with the case where the input word is a valley and then extend our solution to difficult words.

Definition 2.29. *In a Britton-reduced word $w = \alpha_0 t^{\varepsilon_1} \alpha_1 \dots \alpha_{n-1} t^{\varepsilon_n} \alpha_n$, a position $0 \le i \le n$ is called a* sink, *if $\varepsilon_i \neq +1$ and $\varepsilon_{i+1} \neq -1$. The number of sinks in w is denoted* $\mathrm{sk}(w)$.

The position $i = 0$ is a sink if $n = 0$ or $\varepsilon_1 = +1$. The number $\mathrm{sk}(w)$ is bounded by $\lfloor n/2 \rfloor + 1 \in \mathcal{O}(\|w\|)$.

Recall that a Britton-reduced valley is a word, in which every position has level ≤ 0 and the last position has level 0. Valleys will play a crucial role in the algorithms developed in this section. The next lemma is the main reason why the problem of finding geodesic normal forms becomes easier when $p \mid q$.

Lemma 2.30. *([DL11], Lem. 13) Let $p \mid q$ and v be a valley. Then $vp \sim pv$.*

Proof. A Britton-reduced valley is either horocyclic or a concatenation of two (non-empty) valleys or it has the form Tut for some valley u. If $v = \alpha \in \mathbb{Z}$, the claim is obvious. If $v = uw$ for two valleys u and w, then, by induction, $uwp \sim upw \sim puw$. Finally, if $v = Tut$, then

$$\begin{aligned} vp &= Tutp \\ &\sim Tuqt \\ &\sim Tqut && \text{(by induction, since } p \mid q) \\ &\sim pTut = pv. \end{aligned}$$

□

Valleys always have their peak at the last position. This makes the description of the order $<_p$ particularly simple: For two Britton-reduced valleys $v_1\alpha_1$ and $v_2\alpha_2$, where v_1 and v_2 end with the letter t (if they are non-empty), we have $v_1\alpha_1 <_p v_2\alpha_2$ if either $\|v_1\alpha_1\| < \|v_2\alpha_2\|$ or $\|v_1\alpha_1\| = \|v_2\alpha_2\|$ and $v_1 <_s v_2$ or $\|v_1\alpha_1\| = \|v_2\alpha_2\|$ and $v_1 = v_2$ and $\alpha_1 <_s \alpha_2$.

Definition 2.31. *A Britton-reduced valley $V = \alpha_0 t^{\varepsilon_1} \alpha_1 \dots \alpha_{n-1} t^{\varepsilon_n} \alpha_n$ is called a* standard valley *if $\alpha_n = 0$ and $|\alpha_i| < 2|q|$ for $0 \le i < n$.*

Proposition 2.32. *([DL11], Lem. 14) For every Britton-reduced valley v, there is a standard valley V and an integer $\gamma \in \mathbb{Z}$ with $v \sim V\gamma$ and $V\gamma \le_p v$. Given v, such V and γ can be found in $\mathcal{O}(\|v\|)$ time.*

Proof. We use induction on the structure of v to find V and γ:

1.) For $v = \alpha \in \mathbb{Z}$, take $V = 0$ and $\gamma = \alpha$.

2.) Let $v = uw$, where u and w are shorter valleys and u ends with t. By induction, we find a standard valley U and an integer β with $u \sim U\beta$ and $U\beta \leq_p u$. Applying induction a second time, we get a standard valley W and an integer γ such that $\beta w \sim W\gamma$ and $W\gamma \leq_p \beta w$. Defining $V := UW$, we get $v = uw \sim U\beta w \sim UW\gamma = V\gamma$. Since the partition of v into u and w does not cut through a horocyclic element, we have $v = uw \geq_p U\beta w \geq_p UW\gamma = V\gamma$.

3.) Finally, let $v = \alpha Tut$ for some valley u. If $|\alpha| \geq 2|q|$, then $\mathrm{snf}(\alpha)$ contains the letter t and has the form $\mathrm{snf}(\alpha) = t\,\mathrm{snf}(\zeta \cdot p)T\alpha'$ with $|\alpha'| < |q|$. We define $v' := \alpha' Tut(\zeta \cdot q)$. Since $p \mid \zeta \cdot q$, $v \sim v'$. Moreover, $\mathrm{snf}(\zeta \cdot q) = t\mathrm{snf}(\zeta \cdot p)T$ and therefore

$$\begin{aligned}\|v\| = \|\alpha Tut\| &= |\mathrm{snf}(\alpha)| + \|Tut\| = |t\,\mathrm{snf}(\zeta \cdot p)T\alpha'| + \|Tut\| \\ &= |\alpha'| + \|Tut\| + |t\,\mathrm{snf}(\zeta \cdot p)T| = \|v'\|\,.\end{aligned}$$

Together with $v' \leq_s v$, this implies $v' \leq_p v$. We replace v with v' and from now on assume that $v = \alpha Tut\beta$ with $|\alpha| < 2|q|$.

By induction, $u \sim U\delta$ and $u \geq_p U\delta$ for some standard valley U and $\delta \in \mathbb{Z}$. If $|\delta| < 2|q|$, $V := \alpha TU\delta t$ is a standard valley and we take $\gamma := \beta$. Otherwise, $\mathrm{snf}(\delta) = t\,\mathrm{snf}(\mu \cdot p)T\nu$ with $|\nu| < |q|$. We have $v = \alpha tuT\beta \sim \alpha tU\delta t\beta \sim \alpha tU\nu T(\mu \cdot p + \beta) =: v'$. Again, we get $v' \leq_p v$. We define V to be the standard valley $\alpha tU\nu T$ and $\gamma := \mu \cdot p + \beta$, which completes the proof.

□

Proposition 2.32 implies that Britton peak normal forms of valleys are standard valleys with some integer appended. The next lemma restricts the set of possible integers, thereby preparing the ground for a dynamic programming approach. We define a constant r similar to the one in Section 2.4. Let r be the smallest positive integer such that

$$r \geq p \cdot \frac{r + 4|q| - 2}{|q|} + 4|q| - 2.$$

Lemma 2.33. *(cf. [DL11], Lem. 14) Let V be a standard valley. We define a set of integers*

$$R(V) = p\mathbb{Z} \cap \{\rho \in \mathbb{Z} : |\rho| \leq r \cdot \mathrm{sk}(V)\}.$$

For any integer $\rho \in \mathbb{Z}$ and any standard valley V_ρ such that $V \sim V_\rho\,\rho$, we have $\rho \in R(V)$.

Proof. We prove the claim by structural induction on V.

1.) If $V = 0$, then necessarily $\rho = 0 \in R(V)$.

2.) Let $V = UW$ be the product of two shorter non-zero standard valleys. We claim that $\rho = \sigma + \tau$ for some $\sigma \in R(U)$ and $\tau \in R(W)$. Note that $\mathrm{sk}(V) = \mathrm{sk}(U) + \mathrm{sk}(W)$, so the bound holds.
We can split V_ρ into two standard valleys U', W' in such a way that $U \sim U'\zeta$ and $\zeta W \sim W'\rho$. By induction, this implies $\zeta \in R(U) \subseteq p\mathbb{Z}$, so W and ζ commute. Therefore, $W \sim W'(\rho - \zeta)$ and hence $\tau := \rho - \zeta \in R(W)$.

3.) Let $V = \alpha TU\beta t$ with $|\alpha|, |\beta| < 2|q|$. Write $V_\rho = \alpha' TU'\beta' t$ with $|\alpha'|, |\beta'| < 2|q|$ and U' a standard valley. Necessarily, there are $\mu, \sigma, \zeta \in \mathbb{Z}$ such that $\alpha = \mu \cdot p + \alpha'$, $(\mu \cdot q)U \sim U'\sigma$, $\sigma + \beta = \zeta \cdot q + \beta'$, and $\zeta \cdot p = \rho$. Since $\mu \cdot q$ and U commute, we have $U \sim U'(\sigma - \mu \cdot q)$ and, by induction, $\sigma - \mu \cdot q \in R(U)$. Since $\mathrm{sk}(U) = \mathrm{sk}(V)$, we obtain

$$\begin{aligned}
|\rho| &\le p \cdot |\zeta| \le p \cdot \frac{|\sigma| + |\beta - \beta'|}{|q|} \le p \cdot \frac{|\sigma - \mu \cdot q| + |\mu \cdot q| + |\beta - \beta'|}{|q|} \\
&\le p \cdot \frac{r \cdot \mathrm{sk}(U) + 4|q| - 2}{|q|} + |\alpha - \alpha'| \le p \cdot \frac{r \cdot \mathrm{sk}(U) + 4|q| - 2}{|q|} + 4|q| - 2 \\
&\le \mathrm{sk}(U) \cdot \left(p \cdot \frac{r + 4|q| - 2}{|q|} + 4|q| - 2 \right) \le r \cdot \mathrm{sk}(U) = r \cdot \mathrm{sk}(V).
\end{aligned}$$

□

For each $\rho \in R(V)$ there may be several standard valleys V_ρ such that $V \sim V_\rho\,\rho$. We have to find the one that is minimal with respect to $<_p$. With the help of Lemma 2.33, it is not particularly difficult to do this in polynomial time. Note that the time bound in [DL11] is overly optimistic. We prove a cubic bound here.

Proposition 2.34. *There is an $\mathcal{O}(\mathrm{sk}(V) \cdot \|V\|^2)$ time algorithm, which, given a standard valley V, computes for every $\rho \in R(V)$ the word*

$$V_\rho = \min\{U \,:\, U \text{ is a standard valley with } V \sim U\rho\}.$$

The minimum is taken with respect to $<_p$. For some $\rho \in R(V)$, we might have $V_\rho = \min \varnothing$, in which case V_ρ remains undefined.

Proof. We prove the time bound $C \cdot \mathrm{sk}(V) \cdot \|V\|^2$ (for some sufficiently large constant C) using the same kind of induction as we did in the proof of Lemma 2.33.

1.) If $V = 0$, then $V_0 = 0$. For $\rho \neq 0$, V_ρ does not exist.

2.) Let $V = UW$ for two shorter non-zero standard valleys U and W. By Lemma 2.33, we have

$$V_\rho = \min\{U_\sigma W_\tau \,:\, \sigma \in R(U), \tau \in R(W), \sigma + \tau = \rho\}.$$

There are $|R(V)| \in \mathcal{O}(\mathrm{sk}(V)) = \mathcal{O}(\mathrm{sk}(U) + \mathrm{sk}(W))$ many values for ρ that we have to consider. For each ρ, we have to find the minimum among $\min\{|R(U)|, |R(W)|\} \in \mathcal{O}(\min\{\mathrm{sk}(U), \mathrm{sk}(W)\})$ words, which takes the same number of comparisons. For each comparison, we need $\mathcal{O}(\|U\| + \|W\|)$ time. Recursively computing the U_σ and W_τ takes $C \cdot \mathrm{sk}(U) \cdot \|U\|^2 + C \cdot \mathrm{sk}(W) \cdot \|W\|^2$ time. This adds up to

$$\begin{aligned}
&\mathcal{O}\left((\mathrm{sk}(U) + \mathrm{sk}(W)) \cdot \min\{\mathrm{sk}(U), \mathrm{sk}(W)\} \cdot (\|U\| + \|W\|)\right) \\
&\quad + C \cdot \mathrm{sk}(U) \cdot \|U\|^2 + C \cdot \mathrm{sk}(W) \cdot \|W\|^2 \\
&\leq C \cdot \left(\mathrm{sk}(U) \cdot (\|U\| + \|W\|)^2 + \mathrm{sk}(W) \cdot (\|U\| + \|W\|)^2\right) \\
&= C \cdot (\mathrm{sk}(U) + \mathrm{sk}(W)) \cdot (\|U\| + \|W\|)^2 = C \cdot \mathrm{sk}(V) \cdot \|V\|^2.
\end{aligned}$$

3.) Finally, let $V = \alpha T U \beta t$. For each $\rho \in R(V)$ we have

$$V_\rho = \min\{\alpha' T U_\sigma \beta' t : \alpha = \mu \cdot p + \alpha',\ \sigma \in R(U),\ \sigma + \beta = \zeta \cdot q + \beta',\ (\zeta + \mu) \cdot p = \rho\}.$$

Since $|\alpha|, |\alpha'|, |\beta|, |\beta'| < 2|q|$, there is only a constant number of choices for μ and hence also for ζ and σ. For every $\rho \in R(V) = R(U)$, we have to compare a constant number of words of length $\mathcal{O}(\|U\|)$. Together with the recursion, we get the time bound

$$\begin{aligned}
\mathcal{O}(\mathrm{sk}(U) \cdot \|U\|) + C \cdot \mathrm{sk}(U) \cdot \|U\|^2 &\leq C \cdot \mathrm{sk}(U) \cdot (\|U\| + 2)^2 \\
&\leq C \cdot \mathrm{sk}(V) \cdot \|V\|^2.
\end{aligned}$$

□

Putting all things together, we obtain:

Corollary 2.35. *The (Britton) peak normal form of a Britton-reduced valley v can be computed in $\mathcal{O}(\mathrm{sk}(v) \cdot \|v\|^2)$ time.*

Proof. According to Proposition 2.32, we find a standard valley V and an integer γ with $v \sim V\gamma$ and $V\gamma \leq_p v$ in linear time. We have

$$\mathrm{Bpnf}(v) = \mathrm{Bpnf}(V\gamma) = \min\{V_\rho(\rho+\gamma) \,:\, \rho \in R(V)\},$$

where V_ρ is the standard valley from Proposition 2.34. Thus, finding the Britton peak normal form amounts to computing all the V_ρ ($\rho \in R(V)$) and carrying out a minimum search.

Due to the properties of valleys, the peak normal form of v is

$$\mathrm{pnf}(v) = V_\rho \,\mathrm{snf}(\rho+\gamma).$$

□

It remains to close the gap between valleys and difficult words. We start with an observation on Britton peak normal forms of difficult words.

Lemma 2.36. *Let w be a difficult word and i its peak. Then*

$$\mathrm{Bpnf}(T^{\mathrm{lev}(i)}w) = T^{\mathrm{lev}(i)}\mathrm{Bpnf}(w).$$

Proof. There is nothing to do if $i = 0$. For $i > 0$, let $\alpha_0 t^{\varepsilon_1}\alpha_1 \dots \alpha_{n-1}t^{\varepsilon_n}\alpha_n$ be the Britton peak normal form of $T^{\mathrm{lev}(i)}w$. By Proposition 2.6 and because $p \mid q$, the numbers $\alpha_0, \dots, \alpha_{\mathrm{lev}(i)-1}$ are all multiples of p. For each $0 \leq j < \mathrm{lev}(i)$, there is a position $j' \geq \mathrm{lev}(i)$ with the same level as j. If there is more than one such position j', we choose the leftmost one. By Lemma 2.30, each α_j can be shifted to the position j':

$$\alpha_j(T\alpha_{j+1}\dots\alpha_{j'-1}t)\alpha_{j'} \;\sim\; (T\alpha_{j+1}\dots\alpha_{j'-1}t)(\alpha_j+\alpha_{j'})$$

See Figure 2.12 for an example. This decreases the word with respect to $<_p$, contradicting the fact that w is a Britton peak normal form, unless $\alpha_0 = \ldots = \alpha_{\mathrm{lev}(i)-1} = 0$. □

Proposition 2.37. *The Britton peak normal form of a difficult word w can be found in $\mathcal{O}(\mathrm{sk}(w)\cdot\|w\|^2)$ time.*

Proof. Let $w = u\beta v$ with β being the horocyclic element on the peak. Let $\ell, m \geq 0$ be the smallest integers such that $T^\ell w t^m$ is a valley. Compute standard valleys U and V and integers γ, δ such that $T^\ell u \sim U\gamma$ and $T^m v^{-1} \sim V\delta$, according to Proposition 2.32. Using Proposition 2.34, compute the words U_σ ($\sigma \in R(U)$) and V_ρ ($\rho \in R(V)$). By Lemma 2.36, all U_σ start with T^ℓ and all V_ρ with T^m. Thus, we get:

$$\mathrm{Bpnf}(w) = \min\{U_\sigma(\sigma+\gamma+\beta-\delta-\rho)V_\rho^{-1} \,:\, \sigma \in R(U), \rho \in R(V)\}.$$

The minimum can be computed in $\mathcal{O}(\mathrm{sk}(W)\cdot\|W\|)$ time, which is absorbed by the time bound of Proposition 2.34. □

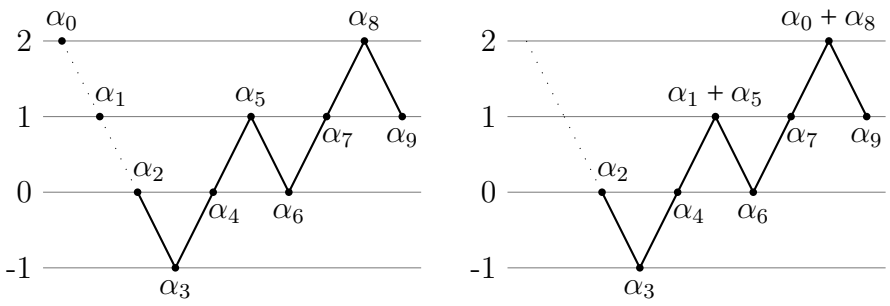


Figure 2.12: Shifting numbers towards the peak (The indicated levels refer to the original word w.)

We summarize the results of this Section in the following Theorem:

Theorem 2.38. *([DL11], Thm. 12) Given any word w in $\mathrm{BS}(p,q)$ where $p \mid q$, $\mathrm{Bpnf}(w)$ and $\mathrm{pnf}(w)$ can be computed in $\mathcal{O}(\mathrm{sk}(w) \cdot \|w\|^2) \subseteq \mathcal{O}(\|w\|^3)$ time.* □

2.8 Geodesics and Growth in $\mathrm{BS}(p, \pm p)$

In this final section, we deal with the formerly excluded case of $p = |q|$. The simplest example $|q| = 1$ has already been mentioned briefly in Section 2.2. Both for $\mathrm{BS}(1,1) \simeq \mathbb{Z}\times\mathbb{Z}$ and for $\mathrm{BS}(1,-1) = \mathbb{Z}\rtimes\mathbb{Z}$, $\{t^m a^n : m,n \in \mathbb{Z}\}$ is a set of geodesic (shortlex) normal forms. It is obviously regular, and so the growth series is rational (see Example 2.5). Given any word $w \in \{a, a^{-1}, t, t^{-1}\}^*$, a logspace-bounded Turing machine can compute the corresponding geodesic normal form in $\tilde{\mathcal{O}}(|w|)$ time.

The case $p = |q|$ differs in a fundamental way from $p < |q|$, since there is no heuristic of the kind "propagate as many a's as possible up to hilltops". The idea that locally changing a word has only limited effect on parts of that word further "up" (this is essentially what the constant r was for), loses validity. On the other hand, finding geodesics is facilitated by the fact that the only way to shorten a word is by free cancellation.

Proposition 2.39. *Every horocyclic element in $\mathrm{BS}(p, \pm p)$, has a^n (for some $n \in \mathbb{Z}$) as its unique geodesic.*

Proof. In $\mathrm{BS}(p, \pm p)$, every Britton reduction decreases the length exactly by 2. Thus, the Britton-reduced form, which for horocyclic elements is uniquely a^n, is the only geodesic. □

The normal forms presented in the next lemma and theorem were first introduced in Section 4 of [EJ92]. We alter them slightly so as to allow $q = -p$ in addition to $q = p$. We also show that our modification not only yields geodesic normal forms but even shortlex normal forms. Finally, we give a time bound for computing these normal forms.

Lemma 2.40. *Let $2 \leq p = |q|$. We call a word $w = \alpha_0 t^{\varepsilon_1} \alpha_1 \dots \alpha_{n-1} t^{\varepsilon_n} \alpha_n$ reduced, if*

(i) w is Britton-reduced,

(ii) $|\alpha_i| < p$ for $0 \leq i < n$,

(iii) $|\alpha_i| + |\alpha_j| \leq p$ for all $i \neq j$ with $\operatorname{sign}(\alpha_i) \neq \operatorname{sign}(q)^{j-i}\operatorname{sign}(\alpha_j)$, and

(iv) $|\alpha_i| < |\alpha_j|$ or $0 \leq \alpha_i = |\alpha_j|$ for all $i < j$ with $\operatorname{sign}(\alpha_i) \neq \operatorname{sign}(q)^{j-i}\operatorname{sign}(\alpha_j)$ and $|\alpha_i| + |\alpha_j| = p$.

If w_1 and w_2 are both reduced and $w_1 \sim w_2$, then $w_1 = w_2$.

We give a full proof of this lemma, since this is omitted in [EJ92], where the authors claim that the proof is rather long (which it isn't).

Proof. Since both w_1 and w_2 are Britton-reduced and describe the same group element, the number of t's and the sequence of their exponents are equal, thus

$$\begin{aligned} w_1 &= \alpha_0 t^{\varepsilon_1} \alpha_1 \dots \alpha_{n-1} t^{\varepsilon_n} \alpha_n \quad \text{and} \\ w_2 &= \beta_0 t^{\varepsilon_1} \beta_1 \dots \beta_{n-1} t^{\varepsilon_n} \beta_n. \end{aligned}$$

We use induction on n. For $n = 0$ there is nothing to prove, so let $n \geq 1$. If $\alpha_0 = \beta_0$, we can use induction on w_1 and w_2 with the common prefix $\alpha_0 t^{\varepsilon_1} = \beta_0 t^{\varepsilon_1}$ cut off. Thus, we assume $\alpha_0 \neq \beta_0$. From $w_1^{-1} w_2 \sim 1$ and the fact that both words are Britton-reduced, we deduce that $\alpha_0 - \beta_0 \equiv 0 \bmod p$ and due to (ii) even $|\alpha_0 - \beta_0| = p$. By symmetry, let $\beta_0 < 0 < \alpha_0$. In order to simplify the rest of the proof, we assume $q > 0$. Choose $i \geq 1$ such that $\alpha_i < 0$ and $|\alpha_i|$ is maximal. If several such i exist, take the smallest; if none exist, take $i = n$. Then

$$w_1 \sim \underbrace{(\alpha_0 - p)}_{=\beta_0} t^{\varepsilon_1} \underbrace{\alpha_1 \dots t^{\varepsilon_{i_1}} (\alpha_i + p) t^{\varepsilon_i} \dots \alpha_{n-1} t^{\varepsilon_n} \alpha_n}_{w_1'}.$$

In order to apply induction, it remains to show that w_1' is reduced. (i) is obvious and (ii) is a consequence of $\alpha_i < 0$ or $i = n$. For (iii) take any $j \neq i$

with $\alpha_j < 0$. Since $|\alpha_j| \leq |\alpha_i|$, we have $|\alpha_j| + |\alpha_i + p| = -\alpha_j + \alpha_i + p \leq p$. The latter becomes an equality only if $\alpha_j = \alpha_i$. Then $i < j$, by the choice of i, so (iv) holds as well.

For $q < 0$, all of this remains true if we replace α_i by $\alpha_i \cdot (-1)^i$ and α_j by $\alpha_j \cdot (-1)^j$ in our reasoning. □

Theorem 2.41. *Let $2 \leq p = |q|$. The set of reduced words coincides with the set of shortlex normal forms for $\mathrm{BS}(p, q)$. This set is regular and the growth series of $\mathrm{BS}(q, \pm q)$ is rational. Given any word, the corresponding shortlex normal form can be computed in quadratic time ($\tilde{\mathcal{O}}(n^2)$ time on a Turing machine).*

Proof. The properties of Lemma 2.40 translate directly into rules:

(i) Compute a Britton-reduction of w (see Theorem 2.3).

(ii) For every $i < n$ write $\alpha_i = \mu \cdot p + \nu$ with $|\nu| < p$. Replace α_i by ν and add $\mathrm{sign}(q)^{n-i} \cdot \mu \cdot p$ to α_n.

(iii) For every $i \neq j$ with $\mathrm{sign}(\alpha_i) \neq \mathrm{sign}(q)^{j-i}\mathrm{sign}(\alpha_j)$ and $|\alpha_i| + |\alpha_j| > p$, replace α_i by $\alpha_i - \mathrm{sign}(\alpha_i) \cdot p$ and α_j by $\alpha_j - \mathrm{sign}(\alpha_j) \cdot p$.

(iv) For every $i < j$ with $\mathrm{sign}(\alpha_i) \neq \mathrm{sign}(q)^{j-i}\mathrm{sign}(\alpha_j)$, $|\alpha_i| + |\alpha_j| = p$, and either $|\alpha_i| > |\alpha_j|$ or $-\alpha_j = \alpha_i < 0$, swap α_i and α_j.

Rules (i) and (iii) shorten the word while rule (iv) preserves the length but decreases the word lexicographically. Rule (ii) either shortens the word or lexicographically decreases it. Therefore, shortlex normal forms are reduced and the first claim of the theorem follows from Lemma 2.40.

The rules can be applied in their order, since none of them affects the applicability of the ones before: rules (ii)-(iv) leave Britton-reduced words Britton-reduced, rules (iii) and (iv) do not make the absolute value of any α_i larger than p, and (iv) just swaps α_i and α_j. Rules (i) and (ii) take linear time, whereas (iii) and (iv) take quadratic time. As all numbers are bounded by the input length, arithmetic operations can be done in polylogarithmic time on a Turing machine.

Since all four properties of Lemma 2.40 can be checked by a finite automaton, the set of shortlex normal forms is regular. This already implies the rationality of the growth series. Moreover, one can explicitly compute the growth series using these properties, see [EJ92]. □

Chapter 3

Power Circuits

In this chapter we will describe the data structure needed in Chapter 4 to solve the word problem in the Baumslag-Gersten groups and in Higman's groups. These so-called power circuits can store the huge integer values that appear as exponents in Britton normal forms in these groups.

Power circuits were introduced by Myasnikov, Ushakov, and Won and first published in [MUW12]. In a second paper [MUW11] they used power circuits to show that the word problem for the Baumslag-Gersten group is polynomial time decidable. While some ideas presented in this chapter are due to their work, there are important differences. First of all, we use a different (and hopefully more accessible) notation, following [DLU12] and [DLU13]. We also allow multiple markings in one circuit. A second modification, which distinguishes [Lau12] and this thesis from earlier papers, is the generalization from base 2 to arbitrary bases $q \geq 2$. The possibility of this generalization was claimed in [MUW12], but no details or proofs were given.

3.1 Power Circuits, Markings, and Evaluation

For the remainder of this thesis, we fix an integer $q \geq 2$ and the interval $D = \{-q+1, \ldots, q-1\} \subseteq \mathbb{Z}$. We start with a directed acyclic edge-labeled graph without multi-edges, given by $\Pi = (\Gamma, \delta)$. Here, Γ is a finite set which acts as the set of nodes (or vertices). The labeled edges (or arcs) are given by the mapping $\delta : \Gamma \times \Gamma \to D$ where $\delta(u, v) = 0$ means that there is no edge from u to v and $\delta(u, v) = e \neq 0$ implies an edge from u to v labeled with the number e. In other words, the edge set is $\operatorname{supp} \delta$, the support of the map δ. In addition, we require that the resulting graph $(\Gamma, \operatorname{supp} \delta)$ is acyclic. We shall make this assumption throughout this chapter without mentioning it

again. For any operation on graphs introduced in this chapter, it will be obvious that acyclicity is preserved.

A marking of $\Pi = (\Gamma, \delta)$ is a mapping $M : \Gamma \to D$. Often we regard M more intuitively as a labeled subset of the nodes of Π, where the subset is $\operatorname{supp} M$ and the labels are given by $M|_{\operatorname{supp} M} : \operatorname{supp} M \to D$. In this sense, $M = 0$ (the constant zero marking) and $M = \varnothing$ (the empty marking) are the same. Each node $u \in \Gamma$ induces a marking Λ_u, called the *successor marking* of u, defined by

$$\Lambda_u : \Gamma \to D;\ v \mapsto M(u, v).$$

From the more intuitive point of view, the successor marking of a node u consists of the target nodes of edges starting at u and their labels are given by those of the edges.

The evaluation function ε assigns a real number to each node and each marking of Π. As Π is acyclic, we can define ε inductively:

$$\begin{aligned} \varepsilon(M) &= \sum_{u \in \operatorname{supp} M} M(u) \cdot \varepsilon(u) && \text{for each marking } M \\ \varepsilon(u) &= q^{\varepsilon(\Lambda_u)} && \text{for each node } u \in \Gamma \end{aligned}$$

Note that this implies $\varepsilon(\varnothing) = 0$, so leaves in the graph (nodes with no outgoing edges) receive the value 1. For every node $u \in \Gamma$ we have

$$\varepsilon(\Lambda_u) = \log_q \varepsilon(u).$$

Example 3.1. *Figure 3.1 shows an example of such a graph for $q = 3$. The set of nodes is $\Gamma = \{u_1, u_2, u_3, u_4, u_5\}$ and δ is given by*

$$\begin{aligned} &\delta(u_2, u_1) = +1, \quad \delta(u_3, u_1) = +2, \quad \delta(u_4, u_1) = -1, \\ &\delta(u_4, u_2) = -2, \quad \delta(u_4, u_3) = +1, \quad \delta(u_5, u_2) = +2, \\ &\delta(u_5, u_3) = +1, \quad \delta(u_5, u_4) = -2. \end{aligned}$$

The nodes evaluate to

$$\varepsilon(u_1) = 1,\ \varepsilon(u_2) = 3,\ \varepsilon(u_3) = 9,\ \varepsilon(u_4) = 9,\ \varepsilon(u_5) = \frac{1}{27}.$$

Lemma 3.2. *Let $\Pi = (\Gamma, \delta)$ be as described above. The following statements are equivalent:*

(i) $\varepsilon(u) \in q^{\mathbb{N}_0} = \{q^n : n \in \mathbb{N}_0\}$ *for every node* $u \in \Gamma$

(ii) $\varepsilon(\Lambda_u) \geq 0$ *for every node* $u \in \Gamma$

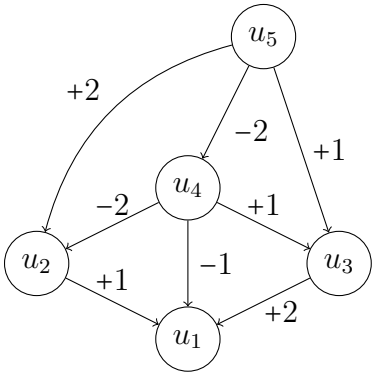


Figure 3.1: Example of an edge-labeled graph ($q = 3$)

(iii) $\varepsilon(M) \in \mathbb{Z}$ *for every possible marking* M *in* Π

Proof. This is easily seen by noetherian induction with respect to a topological order of Γ (i.e., some order compatible with the directed edges). □

Definition 3.3. *A* power circuit *(sometimes just "circuit") is a finite acyclic edge-labeled graph* $\Pi = (\Gamma, \delta)$ *without multiple edges that meets the equivalent conditions of Lemma 3.2.*

Example 3.4. *The graph in Figure 3.1 is not a power circuit since* $\varepsilon(u_5) \notin \mathbb{Z}$. *In contrast, Figure 3.2 depicts a power circuit for* $q = 2$. *The values of the nodes are given for illustrative purposes. In general, these number grow too fast to be written. The marking* M *evaluates to* $\varepsilon(M) = 29$.

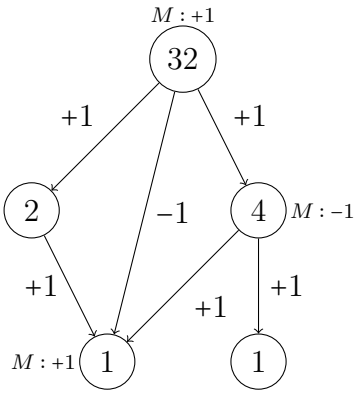


Figure 3.2: Example of a power circuit ($q = 2$)

In Corollary 3.16 we will show that it can be efficiently tested whether a given graph is a power circuit. In [MUW12], a power circuit does not have to satisfy the criteria of Lemma 3.2, but if it does, it is called proper. In this parlance, we only deal with proper power circuits.

The next proposition shows how to convert numbers from "ordinary" (e.g. binary) representation to markings in power circuits.

Lemma 3.5. *For every integer $n \in \mathbb{Z}$, there is a marking M in a power circuit with $\mathcal{O}(\log|n|)$ nodes and $\mathcal{O}(\log|n| \cdot \log\log|n|)$ edges such that $\varepsilon(M) = n$.*

Proof. Write n as a q-ary number, i.e., $n = (-1)^s \sum_{i=0}^{k} \alpha_i \cdot q^i$ with $s \in \{0,1\}$, $0 \le \alpha_i < q$ and $k = \lceil \log_q(|n|+1) \rceil - 1 \in \mathcal{O}(\log|n|)$. Create a power circuit with nodes $u_0, \ldots, u_k$ such that $\varepsilon(u_i) = q^i$. This can be done inductively: If $u_1, \ldots, u_{i-1}$ already exist, write i as a q-ary number $i = \sum_{\ell=0}^{i-1} \beta_\ell \cdot q^\ell$ (this number has in fact only $\mathcal{O}(\log i) \subseteq \mathcal{O}(\log\log|n|)$ digits) and take $\Lambda_{u_i}(u_\ell) = \beta_\ell$ for all $0 \le \ell < i$. Finally, define $M(u_i) = (-1)^s \alpha_i$. □

The time complexity of this algorithm is dominated by writing the number n in q-ary notation. This takes $\tilde{\mathcal{O}}(\log^2|n|)$ time. In group theoretic applications this is negligible, since n is usually given in unary (the input is a word over the group's generators, see Chapter 4), making $|n|$ the input size rather than $\log|n|$.

3.2 Arithmetic Operations

With regard to the group theoretic applications of power circuits in the next chapter, we need to implement two arithmetic operations: given two markings K and M, we want to compute markings which values $\varepsilon(K)+\varepsilon(M)$ and $\varepsilon(K) \cdot q^{\varepsilon(M)}$. For both, we need the notion of cloning.

Let $\Pi = (\Gamma, \delta)$ be a power circuit and $u \in \Gamma$ a node. The operation Clone with result $v = \textsc{Clone}(u)$ creates a new node v with the same successor marking as u, but no incoming arcs. We extend this operation to markings M, by cloning every single node in $\operatorname{supp} M$. The resulting marking $\textsc{Clone}(M)$ is defined as the marking consisting of all these clones, where the signs are copied from M:

$$\textsc{Clone}(M) : \Gamma \dot\cup \{\textsc{Clone}(u) \,:\, u \in \operatorname{supp} M\} \to D;\ \textsc{Clone}(u) \mapsto M(u),\ \Gamma \ni u \mapsto 0.$$

Example 3.6. *In Figure 3.3, the marking M, consisting of two nodes, is cloned.*

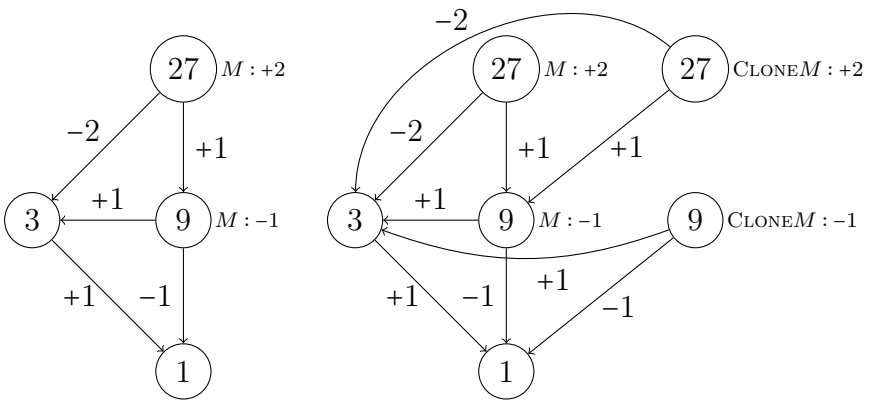


Figure 3.3: Cloning a marking ($q = 3$)

Now we can define arithmetic operations. Let $\Pi = (\Gamma, \delta)$ be a power circuit and let K and M be markings in Π. If the supports of K and M are disjoint, the mapping $K + M$ defined by $(K + M)(u) = K(u) + M(u)$ is a marking with $\varepsilon(K + M) = \varepsilon(K) + \varepsilon(M)$. In general, however, the supports of K and M will not be disjoint. In this case we have nodes $u \in \operatorname{supp} K \cap \operatorname{supp} M$ with $K(u) + M(u) \notin D$, hence $K + M$ is not a valid marking. We solve this problem by cloning: for every node u with $K(u) + M(u) \notin D$, we create a clone $u' = \textsc{Clone}(u)$ and modify $K + M$ by putting $(K + M)(u) := K(u)$ and $(K + M)(u') := M(u)$. We obtain a valid marking in the (now enlarged) circuit with value $\varepsilon(K) + \varepsilon(M)$.

Example 3.7. *In Figure 3.4, $\varepsilon(K) = 7$ and $\varepsilon(M) = 35$ are added. In the resulting marking, the node with value 1 cancels out, whereas both the original node with value 4 and its newly created clone are included.*

For the second operation, we observe that

$$\varepsilon(K) \cdot q^{\varepsilon(M)} = \sum_{u \in \operatorname{supp} K} q^{\varepsilon(\Lambda_u)} \cdot q^{\varepsilon(M)} = \sum_{u \in \operatorname{supp} K} q^{\varepsilon(\Lambda_u) + \varepsilon(M)},$$

so in principle we could just introduce new edges from each node in $\operatorname{supp} K$ to each node in $\operatorname{supp} M$, where the edge signs come from the respective target nodes. This works as long as

1. no cycles are introduced into the circuit,

2. no multi-edges between two nodes are introduced,

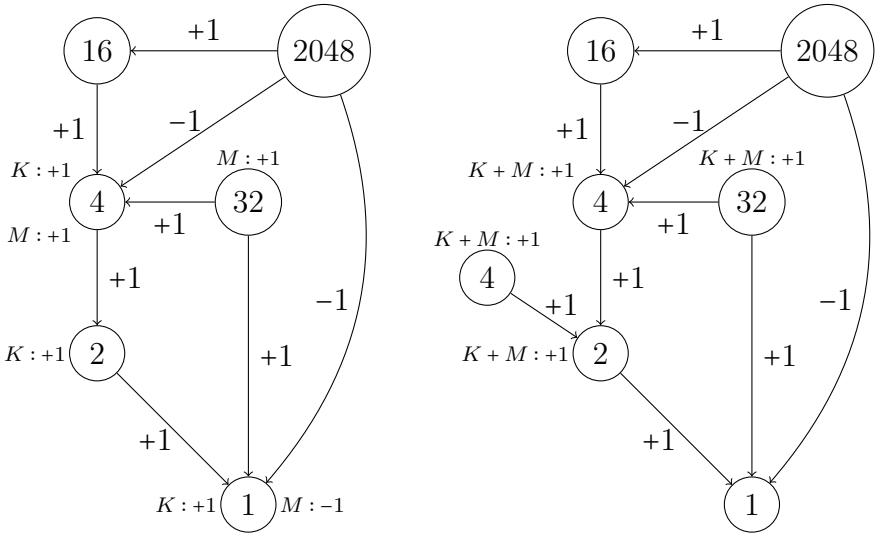


Figure 3.4: Addition of markings ($q = 2$)

3. there are no edges between nodes in $\operatorname{supp} K$ (as the value of the origin of such an edge would change unintentionally, when adding edges from the target node to $\operatorname{supp} M$), and

4. no other marking in the circuit is affected (note that the original value of K is lost in any case).

Again, cloning provides a solution. Let $K' := \textsc{Clone}(K)$ and $M' := \textsc{Clone}(M)$ and introduce new edges by putting $\delta(u, v) := M(v)$ for all $u \in \operatorname{supp} K'$, $v \in \operatorname{supp} M'$. Being clones, nodes in $\operatorname{supp} K'$ and $\operatorname{supp} M'$ have no incoming edges, which prevents cycles and multi-edges. Also, no other marking in the circuit depends on K' or M' directly (by containing these nodes) or indirectly (by containing nodes that are topologically above any node in K' or M'). An example (in which no further cloning is necessary) is shown in Figure 3.5.

Finally, the operation $M \mapsto -M$ which negates the value of M is easy to conduct without any complications or the need for cloning.

By nature, power circuits are particularly suited to compress integers that arise from the operations discussed so far. This includes the tower function tow_q (sometimes called "tetration") which is defined by $\operatorname{tow}_q(0) = 1$ and $\operatorname{tow}_q(n+1) = q^{\operatorname{tow}_q(n)}$. Figure 3.6 shows how to express the value $\operatorname{tow}_q(n)$ by a marking in a power circuit with $n + 1$ nodes.

Other operations such as multiplication of two markings are possible as well (see [MUW12], Sect. 7.3), but the size of the circuit grows by more than

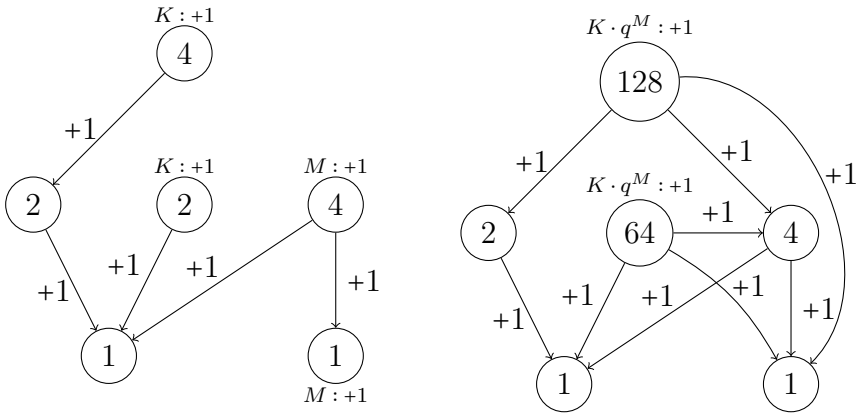


Figure 3.5: Multiplication of $\varepsilon(K)$ by $q^{\varepsilon(M)}$ ($q = 2$)

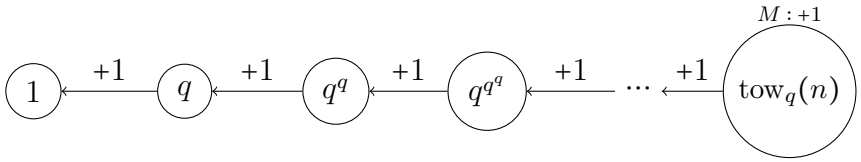


Figure 3.6: Marking with value $\mathrm{tow}_q(n)$

$\mathcal{O}(\operatorname{supp} K + \operatorname{supp} M)$. As we will not need multiplication (except by powers of q) in the following chapter, we do not expand on this.

3.3 Reduction

The operations $K + M$ and $K \cdot q^M$ introduced in the previous section are quite efficient. Assuming that the graph is stored using adjacency lists, the time they take depends only on the size of the markings M and K, not on the size of the circuit. The price for this efficiency is that the structure of a power circuit can quickly become rather intransparent. In particular, it is unclear how (in)equality of the values of two markings can be determined in an arbitrary circuit. Remember that evaluating the nodes or markings is not an option, due to the vast growth permitted by power circuits. For this reason, we restrict ourselves to a subclass of circuits and augment them with some additional data:

Definition 3.8. *A* reduced power circuit *is a power circuit* $\Pi = (\Gamma, \delta)$ *together with an ordered list of its nodes* $\Gamma = (u_1, \ldots, u_n)$ *and a bit vector*

$(b_1, \ldots, b_{n-1}) \in \mathbb{B}^{n-1}$ *such that*

(i) different nodes evaluate to different numbers, i.e., for all $u, v \in \Gamma$ *with* $u \neq v$, $\varepsilon(u) \neq \varepsilon(v)$,

(ii) the nodes are sorted by value, i.e., $\varepsilon(u_1) < \varepsilon(u_2) < \ldots < \varepsilon(u_n)$,

(iii) $b_i = 1$ *if and only if* $q \cdot \varepsilon(u_i) = \varepsilon(u_{i+1})$.

Proposition 3.9. *(cf. [MUW12], Sect. 2.1 and [DLU12], Prop. 5) Given a reduced circuit and two markings* K *and* M*, the values* $\varepsilon(K)$ *and* $\varepsilon(M)$ *can be compared (yielding "<", "=", or ">" as the result) in* $\mathcal{O}(|\Gamma|)$ *time. The algorithm can also determine whether* $|\varepsilon(K) - \varepsilon(M)| = 1$.

Proof. Assume that we want to determine whether for a sum $\varepsilon = \sum_{i=0}^{n} \delta_i \cdot q^i$ with $|\delta_i| \leq 2q - 2$ whether we have $\varepsilon \leq -2$, $\varepsilon = -1$, $\varepsilon = 0$, $\varepsilon = +1$ or $\varepsilon \geq +2$. We can do this inductively using the following procedure:

1.) If $\delta_n = 0$, use induction on $\varepsilon = \sum_{i=0}^{n-1} \delta_i \cdot q^i$.

2.) If $|\delta_n| \geq 2$, then $|\varepsilon| \geq 2 \cdot q^n - \sum_{i=0}^{n-1} (2q-2) \cdot q^i = 2$ and the sign of ε is the same as the sign of δ_n.

3.) If $|\delta_n| = 1$, look at δ_{n-1}. If $\delta_{n-1} = 0$ or if it has the same sign as δ_n, then $|\varepsilon| \geq q^n - \sum_{i=0}^{n-2} (2q-2) \cdot q^i = q^{n-1}(q-2) + 2 \geq 2$ since $q \geq 2$. Again, ε has the same sign as δ_n. If δ_{n-1} has the opposite sign of δ_n, use induction on $\varepsilon = \hat{\delta}_{n-1} \cdot q^{n-1} + \sum_{i=0}^{n-2} \delta_i \cdot q^i$, where $\hat{\delta}_{n-1} = \delta_n \cdot q + \delta_{n-1} \in \{-2q+2, \ldots, 2q-2\}$.

The answer to the original problem can be found by applying this algorithm to the mapping $M - K : \Gamma \to \{-2q+2, \ldots, 2q-2\}$ given by $(M-K)(u) = M(u) - K(u)$. Note that the absolute indices i of the δ_i are not actually needed. Instead one can use the information provided by the reduced circuit. □

Corollary 3.10. *For two markings* K *and* M *in a reduced circuit* $\Pi = (\Gamma, \delta)$*, it can be tested in* $\mathcal{O}(|\Gamma|)$ *time whether* $q^{\varepsilon(K)}$ *divides* $\varepsilon(M)$.

Proof. Let u be the node of minimal value in $\operatorname{supp} M$. As node values are unique in the circuit, $q^{\varepsilon(K)} \mid \varepsilon(M)$ if and only if $q^{\varepsilon(K)} \mid \varepsilon(u)$. Using Proposition 3.9, we can check the equivalent condition $\varepsilon(K) \leq \varepsilon(\Lambda_u)$. □

Power circuits arising from a sequence of arithmetic operations are usually far from being reduced. Every cloning creates a pair of nodes with the same value. Therefore we need an algorithm that takes an arbitrary circuit and produces an equivalent reduced circuit. In this context, equivalence means

that for each node and each marking in the old circuit, there is one with the same value in the reduced circuit. Before specifying the algorithm, we need some preparations.

Definition 3.11. *A list $u_1 \dots, u_k$ of nodes in a power circuit is called a* chain *(starting at u_1), if $q \cdot \varepsilon(u_i) = \varepsilon(u_{i+1})$ for all $1 \le i < n$. It is called a maximal chain, if it is not part of a longer chain.*

Note that chains have nothing to do with paths in the graph. For example, the graph in Figure 3.6 is not a chain, unless n is very small. In arbitrary power circuits, chains are difficult to spot. However, in a reduced power circuit, they can easily be identified using the bit vector.

In a reduced circuit, the maximal chain starting at the unique node with value 1 is of particular interest. It is called the *base chain* of the power circuit. For later use, we define in Algorithm 1 a procedure ProlongBaseChain which appends a new node at the top of this chain without destroying the reducedness property of a circuit.

Algorithm 1: Procedure ProlongBaseChain

input : a reduced power circuit $\Pi = (\Gamma, \delta)$
output: a reduced power circuit $\Pi' = (\Gamma \dot\cup \{u\}, \delta')$ which is Π with an additional node u prolonging the base chain of Π

1 Let $\Gamma = (v_0, \dots, v_n)$ be the ordered list of the nodes of the reduced circuit Π. Using this list and the bit vector, find the smallest $i \ge 0$ such that $\varepsilon(v_i) > q^i$.
2 As in Lemma 3.5, write i as a q-ary number $i = \sum_{\ell=0}^{i-1} \alpha_\ell \cdot q^\ell$ and use this to define the marking $M(v_\ell) = \alpha_\ell$ with value $\varepsilon(M) = i$. Insert a new node u with $\Lambda_u = M$ into the circuit.
3 Place u in the ordered list of nodes between v_{i-1} and v_i.
4 Check whether $q \cdot \varepsilon(u) = \varepsilon(v_i)$ by applying Proposition 3.9 to Λ_u and Λ_{v_i} (both are contained in the reduced circuit Π). Set the bit vector for u accordingly.

The procedure ProlongBaseChain takes $\mathcal{O}(|\Gamma|)$ time on a RAM.

Now we can state an algorithm that reduces power circuits. Reduction is done node by node. This means that at any point during the reduction procedure, the circuit consists of a reduced part and a part that is not yet reduced. The nodes in the non-reduced part are processed in topological order. Therefore, the procedure only has to work for nodes which have all their successors in the already reduced part.

This approach allows us to be more economical in the reduction procedure. Instead of always reducing the complete circuit, we can take into account that parts of it might already be reduced. This will turn out to be useful in applications. The procedure ExtendReduction described in Algorithm 2 takes as its input not only the power circuit but also a list $\mathcal{M}$ of markings that need to be adjusted during reduction in order to preserve their value.

Proposition 3.12. *([Lau12], Prop. 2.11) The procedure* ExtendReduction *is correct and takes* $\tilde{\mathcal{O}}\left((|\Gamma|+|U|)\cdot(|U|+m)\right)$ *time. The circuit growth* $|\Gamma' \setminus \Gamma|$ *is bounded by* $2|U|$.

Proof. At first, a topological order of U is computed. The time for this is bounded by the size of the subgraph U (nodes and edges) which is $\mathcal{O}(|U|^2)$. In the main loop starting at step 2, the nodes are eliminated from U one by one. Let $n = |\Gamma| + |U|$ be the number of nodes in the input graph. Since Γ grows during the procedure (although we keep calling it Γ for convenience), $\mathcal{O}(n)$ is the correct bound for the size of Γ.

For each node $u_i \in U$, its position in the ordered list of Γ has to be found in step 5. Since u_j is chosen to be topologically minimal, the successor marking Λ_u is contained in the reduced circuit Γ, so u can be compared to any node $v \in \Gamma$ in $\mathcal{O}(n)$ time (see Proposition 3.9). Using binary search, $\mathcal{O}(\log n)$ comparisons suffice, taking $\tilde{\mathcal{O}}(n \cdot |U|)$ time in total.

For the insertion of u_i in Γ, we distinguish two cases. In the first one (step 7), there is no node in Γ with the same value as u_i. In this case, u_i is moved from U to Γ without any modification. Markings containing u_i (including successor markings, i.e., edges with target u_i) are not affected either.

The second case (step 11), where there is a node v_j with the same value as u_i, is more complicated. Figure 3.7 shows an example. The idea is to delete u_i and replace it in all markings M (both markings from M and successor markings of nodes in U) by v_j. This may cause v_j to by "overmarked" by M, i.e., $M(v_j) \notin D$. For example, in the simplest case $q = 2$, if $M(u_i) = M(v_j) = 1$, then $M(v_j) = 2$ after the replacement. The solution is inspired by the idea of carry digits that are used when adding two q-ary numbers: if $|M(v_j)| \geq q$, subtract the appropriate number $\alpha \cdot q$ and add α to the value that M assigns to the next node v_{j+1} in the chain, which has q times the value of v_j. The carry might propagate to the end of the chain, which is why we preventively prolonged it with v.

Note that the time bound for one execution of step 15 is not $\tilde{\mathcal{O}}(|\Gamma|)$, but rather $\mathcal{O}(|\Gamma| \cdot \#\text{of markings})$. Since this estimate is not sufficient to prove the claimed bound, we count the total amount of time spent in step 15 during the whole procedure instead. The key observation is that for every carry that has

Algorithm 2: Procedure ExtendReduction

input : a graph $\Pi = (\Gamma \dot\cup U, \delta)$ with $\delta|_{\Gamma\times U} = 0$ and $(\Gamma, \delta|_{\Gamma\times\Gamma})$ a reduced power circuit, a list $\mathcal{M} = (M_1, \dots M_m)$ of markings in Π

output: a reduced power circuit $\Pi' = (\Gamma', \delta')$ with $\Gamma \subseteq \Gamma'$ and $\delta'|_{\Gamma\times\Gamma} = \delta|_{\Gamma\times\Gamma}$ and $\delta'|_{\Gamma\times(\Gamma'\setminus\Gamma)} = 0$, a list $\mathcal{M}' = (M'_1, \dots, M'_m)$ of markings in Π' such that $\varepsilon(M_i) = \varepsilon(M'_i)$

Compute a topological order of U, i.e., $U = (u_1, \dots, u_k)$ such that $\delta(u_i, u_j) \neq 0$ implies $i > j$.

for $i = 1, \dots, k$ **do**

- $U := U \setminus \{u_i\}$
- If $\Gamma = \varnothing$, set $\Gamma := \{u_1\}$ (a circuit with just one node is obviously reduced) and continue with the iteration $i = 2$.
- Let $\Gamma = (v_1, v_2, \dots)$ be the ordered list of the nodes of the reduced circuit Γ. Using binary search, find the minimal j such that $\varepsilon(u_i) \leq \varepsilon(v_j)$. Comparing u_i to some $v \in \Gamma$ is done by comparing Λ_{u_i} to Λ_v.
- **if** $\varepsilon(\Lambda_{u_i}) < 0 = \varepsilon(\Lambda_{v_1})$ **then** the graph Π is not a power circuit; abort the algorithm.
- **if** $\varepsilon(u_i) < \varepsilon(v_j)$ *(or no such v_j exists)* **then**
  - $\Gamma := \Gamma \cup \{u_i\}$
  - Insert u_i into Γ's sorted list of nodes between v_{j-1} and v_j.
  - Check whether $\varepsilon(\Lambda_{u_i}) + 1 = \varepsilon(\Lambda_{v_j})$ and set the bit vector for u_i accordingly
- **else** $\varepsilon(u_i) = \varepsilon(v_j)$
  - Find the last node v_k of the maximal chain starting at v_j and create $v := \textsc{Clone}(v_k)$.
  - Multiply the value of v by q by adding 1 to Λ_v: Let v_ℓ be the first node in the base chain with $\Lambda_v(v_\ell) < q - 1$. If such v_ℓ does not exist, call ProlongBaseChain to create it. Set $\Lambda_v(v_1) = \dots = \Lambda_v(v_{\ell-1}) = 0$ and increment $M(v_\ell)$ by one.
  - Insert v in the ordered list after v_k and set the bit vector for v by comparing Λ_v to $\Lambda_{v_{k+1}}$.
  - **foreach** $M \in \{\Lambda_u : u \in U\} \cup \mathcal{M}$ *with* $u_i \in \operatorname{supp} M$ **do**
    - Replace u_i in M by v_j, i.e., set $M(v_j) := M(v_j) + M(u_i)$ and $M(u_i) := 0$. If now $M(v_j) = \alpha \notin D$, write $\alpha = \beta \cdot q + \gamma$ with $\gamma \in D$, set $M(v_j) := \gamma$ and add β to $M(v_{j+1})$. If again $M(v_{j+1}) \notin D$, repeat. This terminates at the latest at the newly created node v which is not marked by M.

to be moved to the next node in the chain, the number $C := \sum_M \sum_{v \in \Gamma \cup U} |M(v)|$ decreases. Initially, $C \leq (q-1) \cdot (n \cdot (|U| + m))$, so the total time complexity of the loop starting at step 15 is $\mathcal{O}(n \cdot (|U| + m))$. □

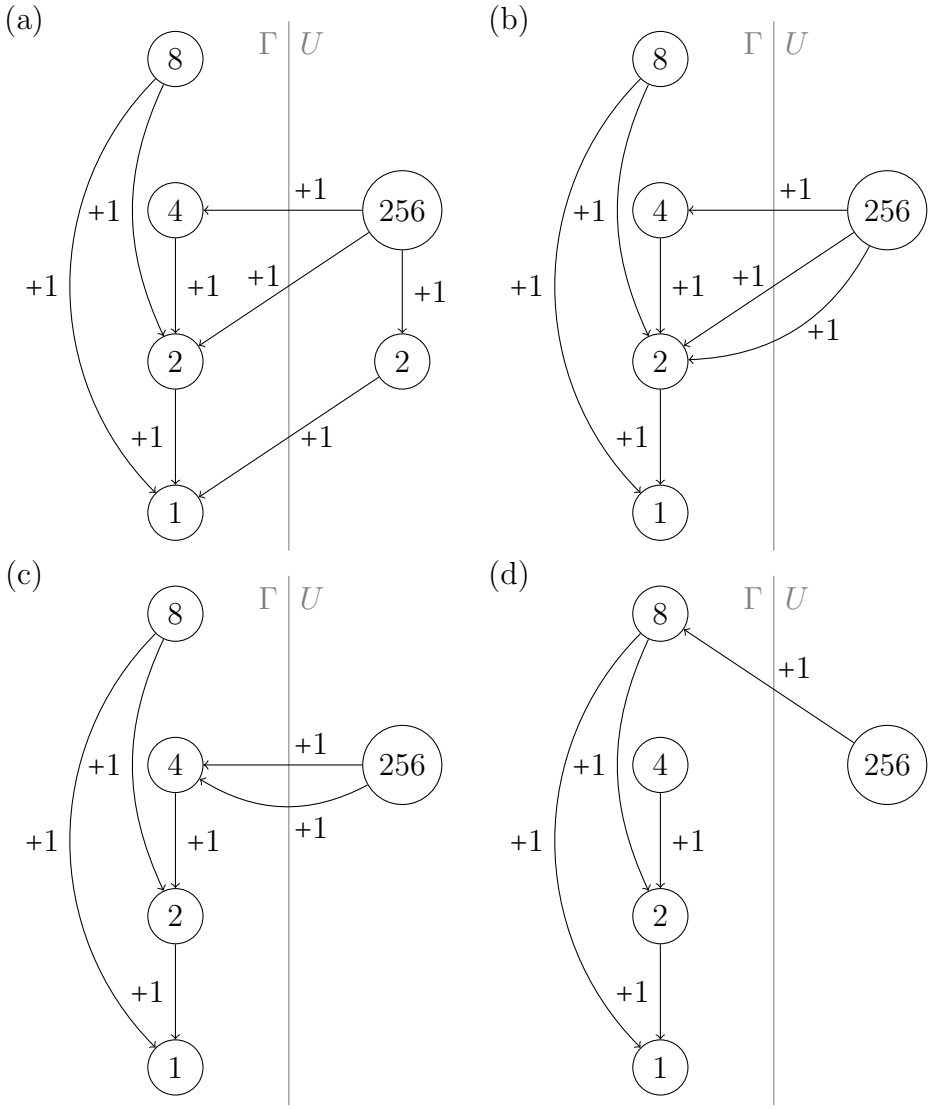


Figure 3.7: Reduction step ($q = 2$)

Remark 3.13. *Not all markings need to be included in* $\mathcal{M}$*. Since* Π *remains a subcircuit of* Π' *and the values of nodes in* Γ *do not change, all markings whose support is completely contained in* Γ *are automatically preserved. Only*

markings using nodes in U have to be put into $\mathcal{M}$. In most applications, $\mathcal{M}$ consists only of a constant number of markings.

Remark 3.14. *The bound on the circuit growth given in Proposition 3.12 is not tight. A more detailed analysis shows that a call to* ProlongBaseChain *is only necessary once every time $|\Gamma|$ grows by a factor of q. If one does some "cleaning up" in the circuit (for instance delete unmarked nodes with no incoming edges), [MUW12] shows that the growth during reduction is even bounded by 1. However, this bound is of no importance in our applications since during arithmetic operations cloning increases the size by $\mathcal{O}(|U|)$ anyway.*

In practice, reduction rarely ever increases the circuit size. Usually, the circuit even shrinks.

Theorem 3.15. *([Lau12], Thm. 2.14) There is a procedure* Reduce *which, given a power circuit $\Pi = (\Gamma, \delta)$ and a list $\mathcal{M} = (M_1, \ldots, M_m)$ of markings in Π, returns a reduced circuit $\Pi' = (\Gamma', \delta')$ and a list $\mathcal{M}' = (M'_1, \ldots, M'_m)$ of markings in Π' such that $\varepsilon(M_i) = \varepsilon(M'_i)$ $(1 \leq i \leq m)$.* Reduce *takes $\tilde{\mathcal{O}}(|\Gamma|^2 + |\Gamma| \cdot m)$ time and the size of Γ' is bounded by $2|\Gamma|$.*

Proof. Invoke ExtendReduction with $\varnothing$ as the reduced part and the whole circuit as U. □

Step 6 in ExtendReduction tests whether $\varepsilon(\Lambda_u) \geq 0$. This is one of the equivalent conditions specified in Lemma 3.2 for a graph to be a power circuit. Therefore, reduction is at the same time a test whether a graph is a power circuit:

Corollary 3.16. *([Lau12], Cor. 2.15) Given a dag $\Pi = (\Gamma, \delta)$ it can be determined in $\tilde{\mathcal{O}}(|\Gamma|^2)$ time whether Π is a power circuit.* □

3.4 Compactness

In this section, we will show that by using a richer data structure for reduced circuits, the time complexity of ExtendReduction can be reduced to $\mathcal{O}((|\Gamma| + |U|) \cdot (|U| + m))$. This eliminates the logarithmic factors both for Reduce and for the test from Corollary 3.16. We start by taking a closer look at the kind of sums that occur when we evaluate markings.

3.4.1 Compact Power Sums

A power sum is a formal sum $S = \sum_{i \geq 0} \alpha_i \cdot q^i$ whose coefficients α_i are in D and only finitely many are non-zero. The value $\varepsilon(S) \in \mathbb{Z}$ is the integer we

get when evaluating the formal sum S like a usual term. On the set of all power sums, we define a rewriting system $\mathcal{P}$ consisting of the rules

$$\begin{aligned}
&(1) \quad \alpha \cdot q^i + \beta \cdot q^{i+1} \longrightarrow (\alpha - q) \cdot q^i + (\beta + 1) \cdot q^{i+1} && \text{for } \alpha > 0,\ \beta < 0,\\
&(2) \quad \alpha \cdot q^i + \beta \cdot q^{i+1} \longrightarrow (\alpha + q) \cdot q^i + (\beta - 1) \cdot q^{i+1} && \text{for } \alpha < 0,\ \beta > 0,
\end{aligned}$$

and for $i < j$

$$\begin{aligned}
&(3) \quad \alpha \cdot q^i + (q-1) \cdot (q^{i+1} + \ldots + q^j) + \beta \cdot q^{j+1}\\
&\qquad \longrightarrow (\alpha - q) \cdot q^i + (\beta + 1) \cdot q^{j+1} && \text{for } \alpha > 0,\ \beta < q - 1,\\
&(4) \quad \alpha \cdot q^i + (-q+1) \cdot (q^{i+1} + \ldots + q^j) + \beta \cdot q^{j+1}\\
&\qquad \longrightarrow (\alpha + q) \cdot q^i + (\beta - 1) \cdot q^{j+1} && \text{for } \alpha < 0,\ \beta > -q + 1.
\end{aligned}$$

Although we use shorthands like a for $a \cdot q^0$ or $a \cdot (1+q+q^2)$ for $a \cdot q^0 + a \cdot q^1 + a \cdot q^2$, power sums should be regarded as strings and the rules as rewriting rules. For example, we are not allowed to make an algebraic rearrangement before applying a rule like in

$$1 \cdot q + 1 \cdot q^2 = 1 \cdot q - 1 \cdot q^2 + 2 \cdot q^2 \underset{(1)}{\Longrightarrow} -2 \cdot q + 0 \cdot q^2 + 2 \cdot q^2 \qquad (q = 3),$$

since rule (1) could not have been applied to the original power sum.

None of the rules change the value of the power sum. We omit the proof of the next lemma, since it consists of a long but simple enumeration of cases.

Lemma 3.17. *The rewriting system $\mathcal{P}$ is locally confluent.* □

Lemma 3.18. *The rewriting system $\mathcal{P}$ is terminating and therefore confluent ([Jan88], Prop. 1.1.19).*

Proof. Let $S_1 \overset{*}{\underset{\mathcal{P}}{\Longrightarrow}} S_2 \overset{*}{\underset{\mathcal{P}}{\Longrightarrow}} \ldots$ be a sequence of rewritings. Since none of the rules of $\mathcal{P}$ increase the number of non-zero coefficients, this number must eventually reach a minimum. Thus, ignoring a finite number of terms, we can assume that the number of non-zeros is constant within the sequence. No rule in $\mathcal{P}$ moves a non-zero coefficient to the left (in the direction of smaller exponents). As the value of the S_i is fixed, non-zeros cannot be moved indefinitely to the right either. Again, by disregarding a finite prefix of the sequence, we assume that the positions of the non-zero coefficients are fixed. At this point, no application of (3) or (4) is possible. Finally, rules of type (1) and (2) move pairs of consecutive coefficients with opposite signs to the left (or remove them), which can also occur only finitely often. Thus, the sequence $S_1, S_2, \ldots$ is eventually constant. □

We call power sums that are irreducible with respect to $\mathcal{P}$ *compact*. If $S = \sum_{i\geq 0} \alpha_i q^i$ is compact, then so is $-S = \sum_{i\geq 0}(-\alpha_i)q^i$.

Proposition 3.19. *([Lau12], Prop. 2.18)*

(i) For each power sum there is a unique compact power sum with the same value.

(ii) Compact power sums have the minimal number of non-zero coefficients among all power sums of the same value.

(iii) If S and T are compact power sums, then $\varepsilon(T) = \varepsilon(S) + 1$ if and only if

$$S = \sum_{j=0}^{i-1} \alpha_j \cdot q^j + \alpha_i \cdot q^i + U \cdot q^{i+1} \quad \text{and}$$
$$T = \sum_{j=0}^{i-1} \beta_j \cdot q^j + \beta_i \cdot q^i + U \cdot q^{i+1}$$

for some power sum U, $\beta_i = \alpha_i + 1$, and for each $0 \leq j < i$ either $\alpha_j = q-1$ and $\beta_j = 0$ or $\alpha_j = 0$ and $\beta_j = -q+1$.

(iv) The usual order on D gives rise to a lexicographical order on power sums (where coefficients of higher powers of q are compared before those of lower powers). Restricted to compact power sums, this lexicographical order coincides with the order by value.

Proof. For (i) it suffices to show that for two power sums S and T with the same value, we have $S \overset{*}{\underset{\mathcal{P}}{\Longleftrightarrow}} T$. This is true, since applying the rules of $\mathcal{P}$ forward or backward, one can turn any power sum into an ordinary q-ary number with coefficients from $\{0, \ldots, q-1\}$. (For instance, for positive values of S, use rules of type (1) backward and rules of type (2) forward to push negative coefficients to the right until they eventually disappear.)

Claim (ii) follows from the fact that no rule increases the number of non-zero coefficients.

For the "if" part of (iii), we observe that component-wise subtraction yields

$$T - S = -(q-1)\sum_{j=0}^{i-1} q^j + q^i,$$

which evaluates to 1. For the "only if" part, let $S = \sum_{\ell\geq 0} \alpha_\ell \cdot q^\ell$ be compact. Consider $S' = (\alpha_0 + 1) + \sum_{\ell\geq 1} \alpha_\ell \cdot q^\ell$. If S' is a valid power sum (i.e., $\alpha_0 + 1 \in D$) and S' is compact, it already has the desired form (for $i = 0$). Otherwise we have one of the following cases:

1.) S' is not a valid power sum, since $\alpha_0 = q-1$. Let $k \geq 0$ be the maximum number such that $\alpha_0 = \ldots = \alpha_k = q-1$. We transform S' into

$$(\alpha_{k+1}+1)\cdot q^{k+1} + \sum_{\ell > k+1} \alpha_\ell \cdot q^\ell$$

and use induction on $(\alpha_{k+1}+1) + \sum_{\ell>k+1} \alpha_\ell \cdot q^{\ell-k-1}$.

2.) A rule of type (1) can be applied. We have $\alpha_0 = 0$ and $\alpha_1 < 0$. Applying the rule gives

$$S' \underset{(1)}{\Longrightarrow} (-q+1) + (\alpha_1+1)\cdot q + \sum_{\ell \geq 2} \alpha_\ell \cdot q^\ell.$$

Use induction on $(\alpha_1+1) + \sum_{\ell\geq 2} \alpha_\ell \cdot q^{\ell-1}$.

3.) A rule of type (3) can be applied. We have $\alpha_0 = 0$ and $\alpha_1 = \ldots = \alpha_k = q-1$ and $\alpha_{k+1} < q-1$ for some $k \geq 1$. This yields

$$S' \underset{(3)}{\Longrightarrow} (-q+1) + (\alpha_{k+1}+1)\cdot q^{k+1} + \sum_{\ell > k+1} \alpha_\ell \cdot q^\ell$$

and induction applies to $(\alpha_{k+1}+1) + \sum_{\ell>k} \alpha_\ell \cdot q^{\ell-k-1}$.

Finally, (iv) is a consequence of (iii). □

The notion of compactness was introduced in [MUW12] for $q = 2$ and subsequently used in [DLU13]. Our definition originates from [Lau12] and it is a generalization that inherits most of the original characteristics as we have seen in Proposition 3.19.

There is, however, one important difference: it is much less obvious for $q > 2$ how to make a power sum compact in linear time. In the case $q = 2$ it suffices to apply the rules of $\mathcal{P}$ from left to right. Yet, if for instance $q = 3$, the application of rule (1) to

$$1 + q + q^2 + q^3 - 2q^4 \underset{(1)}{\Longrightarrow} 1 + q + q^2 - 2q^3 - q^4$$

turns the previously compact prefix $1 + q + q^2 + q^3$ into the $\mathcal{P}$-reducible sum $1 + q + q^2 - 2q^3$.

Proposition 3.20. *([Lau12], Prop. 2.19) Any power sum $S = \sum_{i=0}^{n} \alpha_i \cdot q^i$ can be transformed into a compact power sum with the same value in $\mathcal{O}(n)$ time.*

Proof. Any two power sums S and T with $\varepsilon(S) = \varepsilon(T)$ can be transformed into each other using only replacements of the form

$$\alpha \cdot q^{i-1} + \beta \cdot q^i \longrightarrow (\alpha \mp q) \cdot q^{i-1} + (\beta \pm 1) \cdot q^i. \qquad (\star_i^{\pm})$$

Moreover, at most one application of $(\star_i^+)$ or $(\star_i^-)$ is needed for each i. In fact, whether $(\star_i^+)$ or $(\star_i^-)$ or neither is needed, depends only on S and T. This can be seen using induction on i.

Let $T = \sum_{i=0}^{n+1} \beta_i \cdot q^i$ be the compact power sum with $\varepsilon(S) = \varepsilon(T)$. Define $J_i \in \{+1, -1, 0\}$ $(1 \le i \le n)$ depending on whether the replacement $(\star_i^+)$ or $(\star_i^-)$ or neither of them occurs in the sequence $S \overset{*}{\underset{(\star_i^{\pm})}{\Longrightarrow}} T$. For convenience, we define $J_0 = J_{n+2} = 0$ and $\alpha_{n+1} = 0$. Then we have $\beta_i = \alpha_i + J_i - q \cdot J_{i+1}$ for $0 \le i \le n+1$.

It remains to compute the values J_i $(1 \le i \le n+1)$. These are the unique solution of the following system of conditions $(1 \le i \le n+1)$:

$$\begin{aligned}(\diamond_i)\quad & \alpha_{i-1} + J_{i-1} - q \cdot J_i \in D \text{ and } \alpha_i + J_i - q \cdot J_{i+1} \in D \text{ and} \\ & \operatorname{sign}(\alpha_{i-1} + J_{i-1} - q \cdot J_i) \cdot \operatorname{sign}(\alpha_i + J_i - q \cdot J_{i+1}) \neq -1 \text{ and} \\ & \text{if } \operatorname{sign}(\alpha_{i-1} + J_{i-1} - q \cdot J_i) = \operatorname{sign}(\alpha_i + J_i - q \cdot J_{i+1}) = \pm 1, \\ & \qquad \text{then } \alpha_i + J_i - q \cdot J_{i+1} \neq \pm(q-1).\end{aligned}$$

Note that each condition $(\diamond_i)$ can be checked locally without the knowledge of any other value than J_{i-1}, J_i, J_{i+1} (and α_{i-1}, α_i). For all $2 \le i \le n+2$ we define $\mathcal{J}_i$ to be the set of possible values for J_i, depending on J_{i-1}:

$$\begin{aligned}\mathcal{J}_i := \{(J_{i-1}, J_i) \in \{-1, 0, +1\}^2 \,:\, & \text{there are } J_1, \ldots, J_{i-2} \text{ such that} \\ & J_0 = 0, J_1, \ldots, J_{i-2}, J_{i-1}, J_i \\ & \text{satisfy } (\diamond_1), \ldots, (\diamond_{i-1})\}\end{aligned}$$

The set $\mathcal{J}_2$ can be computed directly in constant time. For $i \ge 3$ we have

$$\mathcal{J}_i = \{(J_{i-1}, J_i) \,:\, \exists J_{i-2} : (J_{i-2}, J_{i-1}) \in \mathcal{J}_{i-1} \wedge J_{i-2}, J_{i-1}, J_i \text{ satisfy } (\diamond_{i-1})\}.$$

Hence, knowing $\mathcal{J}_{i-1}$, the set $\mathcal{J}_i$ can be computed in constant time. Since we already know that there is exactly one solution, $\mathcal{J}_{n+2}$ must contain a unique pair $(J_{n+1}, 0)$. For this value of J_{n+1} we find exactly one pair $(J_n, J_{n+1}) \in \mathcal{J}_{n+1}$ and so on. □

Example 3.21. *Let $q = 3$ and suppose we want to make $S = 1 + q - 2q^2 - 2q^3 +$*

$q^4 - q^5$ *compact. We get:*

$$\begin{aligned}
\mathcal{J}_2 &= \{(0,0),(1,1)\}\\
\mathcal{J}_3 &= \{(0,-1),(1,0)\}\\
\mathcal{J}_4 &= \{(-1,-1)\}\\
\mathcal{J}_5 &= \{(-1,0)\}\\
\mathcal{J}_6 &= \{(0,0),(0,-1)\}\\
\mathcal{J}_7 &= \{(0,0)\}
\end{aligned}$$

From $(J_6, 0) \in \mathcal{J}_7$ *we get* $J_6 = 0$. *Looking at* $\mathcal{J}_6$, *we find* $J_5 = 0$, *and so on. We end up with* $J_0 = J_1 = J_2 = 0$, $J_3 = J_4 = -1$, $J_5 = J_6 = J_7 = 0$ *which results in* $T = 1 + q + q^2 - q^5$.

3.4.2 Inherent Limitations of Power Circuits

Property (ii) of Proposition 3.19 can be used to prove lower bounds for power circuits. The following two propositions are straightforward generalizations of the results in Section 9 of [MUW12].

Proposition 3.22. *(cf. [MUW12], Prop. 9.1) Dividing a number, given as a marking in a power circuit, by* q^2-1 *(and representing the result as a marking in the circuit) can cause non-elementary growth of the circuit.*

Proof. Let $m = \mathrm{tow}_q(n)$ and $A_m = \sum_{i=1}^{m} q^{2i}$ and $B_m = q^{2m+2}$. We have $A_m = \frac{B_m}{q^2-1}$. B_m can be realized as a marking in a power circuit with $n+3$ nodes, see Figure 3.8. Yet, the power sum used to define A_m is compact (no two consecutive coefficients are non-zero), so by Proposition 3.19 (ii), no marking using fewer than m nodes can evaluate to A_m. □

Proposition 3.23. *(cf. [MUW12], Sect. 9.2) Multiplication of linearly many numbers given as markings in a power circuit can cause exponential growth of the circuit.*

Proof. Let k be a constant which we will choose later. There is a power circuit with $n+1$ nodes in which we can choose markings $M_k, \ldots, M_n$ with values $\varepsilon(M_i) = \mathrm{tow}_q(i) + 1$ (cf. Figure 3.6). Multiplying these numbers and expanding, we get

$$\begin{aligned}
\prod_{i=k}^{n} \varepsilon(M_i) = \prod_{i=k}^{n} (\mathrm{tow}_q(i) + 1) &= \sum_{\sigma_k,\ldots,\sigma_n \in \{0,1\}} \prod_{\substack{k \le i \le n,\\ \sigma_i = 1}} \mathrm{tow}_q(i)\\
&= \sum_{\sigma_k,\ldots,\sigma_n \in \{0,1\}} \prod_{\substack{k \le i \le n,\\ \sigma_i = 1}} q^{\mathrm{tow}_q(i-1)} = \sum_{\sigma_k,\ldots,\sigma_n \in \{0,1\}} q^{s(\sigma_k,\ldots,\sigma_n)},
\end{aligned}$$

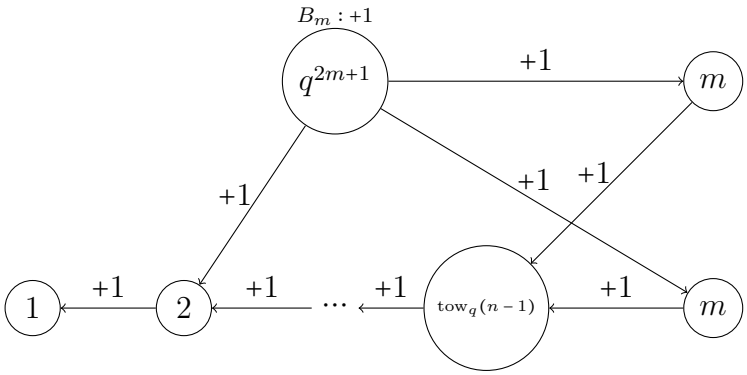


Figure 3.8: Power circuit for B_m

where $s(\sigma_k, \ldots, \sigma_n) = \sum_{i=k}^{n} \sigma_i \cdot \mathrm{tow}_q(i-1)$. For sufficiently large k, the sum $\sum q^{s(\sigma_k,\ldots,\sigma_n)}$ is compact. Hence, any power containing a marking with value $\prod_{i=k}^{n} \varepsilon(M_i)$ must have at least $2^{n-k+1} \in \Theta(2^n)$ nodes. □

3.4.3 Power Circuits and Trees

Property (iv) of Proposition 3.19 is the main motivation for the following definition. The idea of using a tree to store markings originates from [DLU13].

Definition 3.24. *A power circuit $\Pi = (\Gamma, \delta)$ is called* treed *(a shorthand for "reduced and equipped with additional data in the form of a tree") if*

(i) *Π is reduced (i.e., different nodes evaluate to different values and it is equipped with an ordered list $\Gamma = (u_1, \ldots, u_n)$ of the nodes and a bit vector $(b_1, \ldots, b_{n-1}) \in \mathbb{B}^{n-1}$, see Definition 3.8)*

(ii) *Π is equipped with a directed tree T in which each node has up to $|D|$ outgoing edges, labeled with pairwise distinct values from D (and ordered by increasing values). For each leaf in T, the unique path from the root to this leaf must consist of exactly $n = |\Gamma|$ edges. The sequence of labels $\alpha_1, \ldots, \alpha_n$ of such a path (read from the leaf to the root) corresponds to a marking $M : u_i \mapsto \alpha_i$. For each leaf, the power sum $\sum_{i=1}^{n} \alpha_i \cdot \varepsilon(u_i)$ (which evaluates to $\varepsilon(M)$) must be compact. Any marking represented as a leaf in this way is called* compact*.*

(iii) *All successor markings Λ_v ($v \in \Gamma$) are compact.*

The exact data structure used for storing the tree is not important, as long as the usual tree operations (such as enumerating all children of a node,

adding a child, getting the parent of a node) can be performed in constant time. In addition, we need some means of enumerating all the nodes in each level of the tree (a linked list, for example).

Example 3.25. *Figure 3.9 shows a treed circuit (with $q = 2$) alongside the tree containing its successor markings and an additional compact marking M. The order of the nodes is implicitly given by the lowest level of the tree.*

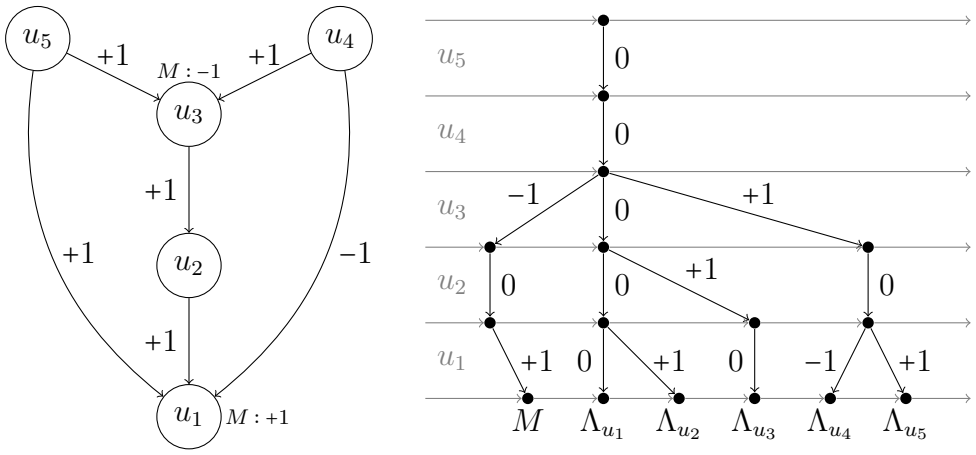


Figure 3.9: Treed power circuit ($q = 2$) with compact marking M

The most time consuming step in ExtendReduction was to find the position of a new node in the sorted list of Γ. Using binary search, this took $\mathcal{O}(n \log n)$ time. If Γ is treed and Λ_u is compact, we can improve this to $\mathcal{O}(n)$: the position of the leaf corresponding to Λ_u already determines the position of u in Γ. In order to adjust the bit vector, we have to read the paths from the root of the tree to the respective leaves and check the condition given by Proposition 3.19 (iii). On the other hand, making a circuit treed is more complicated than just reducing it.

For the time analysis of the new reduction procedure, we will use amortization with respect to a potential function pot, mapping power circuits to numbers, see Section 17.3 in [CLRS09]. An algorithm on power circuits is said to run in amortized time t, if the real running time is bounded by $t + \mathrm{pot}(\Pi) - \mathrm{pot}(\Pi')$, where Π is the input and Π' is the resulting circuit. Therefore, an algorithm may take longer than its indicated amortized time, as long as it decreases the potential by the same amount. The potential can be thought of as an asset which is accumulated whenever the algorithm finishes early and reduced if the algorithm does not terminate in time.

The number of leaves in a treed power circuit depends not only on the circuit itself (i.e. Γ and δ) but also on the number of compact markings that are not successor markings. Furthermore, a power circuit usually grows in the course of the execution of an algorithm. In order to keep time analysis as simple as possible, we take n to be an upper bound on the number of nodes and leaves.

Definition 3.26. *For a power circuit $\Pi = (\Gamma, \delta)$, the number of maximal chains (see Def. 3.11) is denoted by* $\operatorname{ch}(\Pi)$. *The* potential *of Π is* $\operatorname{pot}(\Pi) = \operatorname{ch}(\Pi) \cdot n$.

In a situation where $\Pi = (\Gamma \dot\cup U, \delta)$ is a graph with a power circuit $\Pi' = (\Gamma, \delta|_{\Gamma\times\Gamma})$ as a subgraph, we define $\operatorname{ch}(\Pi) := \operatorname{ch}(\Pi')$ *and* $\operatorname{pot}(\Pi) := \operatorname{pot}(\Pi')$.

Lemma 3.27. *([Lau12], Lem. 2.24) There is a procedure* InsertNode, *which takes a treed power circuit $\Pi = (\Gamma, \delta)$ and a compact marking M in Π and which inserts a new node u with $\Lambda_u = M$ into Π, leaving the circuit treed.* InsertNode *runs in amortized time $\mathcal{O}(n)$.*

Proof. Since M is compact, the position of v in the ordered list of nodes is determined by the position of the leaf corresponding to M in the tree. The bit vector can be adjusted by comparing M to the successor markings of the nodes immediately before and after v. The tree has to be "stretched" by inserting a new level corresponding to v. All edges on this level are labeled 0, since no marking uses the new node v. This takes $\mathcal{O}(n)$ time. The insertion of v may increase $\operatorname{ch}(\Pi)$ by one. Thus, the potential grows by up to n. □

Incrementing a compact marking by 1 is easier than doing the same with an arbitrary marking. Let M be a compact marking in a treed circuit and let u_1, u_2 be the unique nodes with values 1 and q, respectively. If $M(u_1) < q-1$, $M(u_1)$ can be incremented directly. If $M(u_1) = q-1$, then $M(u_2) < q-1$ due to compactness. In this case, we set $M(u_1) := 0$ and increment $M(u_2)$. However, the resulting marking is not always compact.

Lemma 3.28. *([Lau12], Lem. 2.25) There is a procedure* CompactifyMarking *which, given an arbitrary marking M in a treed circuit $\Pi = (\Gamma, \delta)$, computes in $\mathcal{O}(n)$ time a compact marking M' in Π with $\varepsilon(M') = \varepsilon(M)$, given that the following condition holds:*

> *For each maximal chain $C \subseteq \Gamma$, either the power sum $\varepsilon(M|_C)$ is compact or* $\operatorname{supp} M$ *does not contain the top node of C.* (3.1)

Proof. If $S = \sum_{i=k}^{\ell} \alpha_i \cdot q^i$ is a power sum, only the coefficients of $q^k, \dots, q^{\ell+1}$ can be non-zero in the corresponding compact power sum. For each maximal

chain $C \subseteq \Gamma$, we can apply the algorithm from Proposition 3.20 to the power sum $\varepsilon(M|_C)$ individually. Condition (3.1) ensures that a node for $q^{\ell+1}$ exists in Π, if needed. After that, we create a path for the marking in the tree. □

In the case of ProlongBaseChain, Condition (3.1) is satisfied. The successor marking of the new top node of the base chain only uses nodes further down in that chain which, by definition, already exist. This shows:

Corollary 3.29. *([Lau12], Cor. 2.26) The procedure* ProlongBaseChain *given in Algorithm 1 can be adapted for treed power circuits. The amortized time complexity is* $\mathcal{O}(n)$. □

Corollary 3.30. *([Lau12], Cor. 2.27) There is a procedure* IncrementMarking *which, given a compact marking* M *in a treed power circuit* $\Pi = (\Gamma, \delta)$*, computes a compact marking* M' *in a (possibly enlarged) treed circuit* Π' *with* $\varepsilon(M') = \varepsilon(M)+1$. IncrementMarking *takes* $\mathcal{O}(n)$ *amortized time and increases the circuit size by* $1 + \mathrm{ch}(\Pi) - \mathrm{ch}(\Pi')$.

Proof. We invoke ProlongBaseChain which creates a new node u. If u is not the new top node of the base chain, the insertion of u has linked two maximal chains, decreasing $\mathrm{ch}(\Pi)$ by one. We use the released potential to pay for the $\mathcal{O}(n)$ time used so far and repeat until we have created a new top node of the base chain which is not used by any marking.

As discussed above, incrementing M by 1 only affects the two nodes with values 1 and q. As a consequence, the compactification process of the incremented marking is limited to the base chain of Π and ends at the newly created node. □

The procedure ExtendTree given in Algorithm 3 replaces ExtendReduction for treed circuits.

Proposition 3.31. *([Lau12], Prop. 2.28) The procedure* ExtendTree *is correct and runs in amortized time* $\mathcal{O}(n\cdot(|U|+m))$. *The circuit growth* $|\Gamma' \setminus \Gamma|$ *is bounded by* $4 \cdot |U| + \mathrm{ch}(\Pi) - \mathrm{ch}(\Pi')$.

Proof. The basic structure of ExtendTree is the same as that of ExtendReduction, so we focus on the differences. Between cycles of the main loop, we keep up the following invariants for all markings $M \in \mathcal{M} \cup \{\Lambda_u : u \in U\}$:

(a) If the support of M is completely contained in Γ, M is compact.

(b) For each maximal chain $C \subseteq \Gamma$, either the power sum $\varepsilon(M|_C)$ is compact or the top node of C is not contained in $\operatorname{supp} M$.

The second invariant is true by assumption, the first one is established in step 1.

The time for finding v_j in step 7 is reduced to $\mathcal{O}(n)$ due to the representation of markings as leaves.

If Γ contains no node with the same value as u_i, we can insert it as we did in ExtendReduction. Remember that Λ_{u_i} is compact due to (a). The case $\varepsilon(u_i) = \varepsilon(v_j)$ also resembles ExtendReduction and has the same (although now amortized) time bound.

Let $C \subseteq \Gamma$ be the maximal chain starting at v_j. If a marking $M \in \mathcal{M} \cup \{\Lambda_u : u \in U\}$ had u_i in its support, the power sum $\varepsilon(M|_C)$ is probably not compact anymore after replacing u_i in M by v_j. In order to restore (b), we create a new top node for C in step 15. As we have seen in Corollary 3.30, this takes amortized time $\mathcal{O}(n)$ and increases the circuit size by $1 + \mathrm{ch}(\Pi) - \mathrm{ch}(\Pi')$.

All markings M whose support is completely contained in Γ after the processing of u_i must be made compact in order to maintain (a). This is done in step 18. Invariant (b) ensures that Condition (3.1) of Lemma 3.28 holds. □

With regard to invariant (b), we can weaken the requirement of ExtendTree that the Γ-parts of markings must correspond to compact power sums and alternatively demand that the top nodes of the respective chains are not marked by M. However, in our applications, the stronger precondition always holds. If we drop the requirement entirely, we lose the time bound. Up to $\mathcal{O}(n)$ new nodes are required to make an arbitrary marking compact. It takes $\mathcal{O}(n^2)$ time to create these nodes.

The next theorem is the analogon of Theorem 3.15 for treed circuits.

Theorem 3.32. *([Lau12], Thm. 2.29) There is a procedure* MakeTree *which, given a power circuit $\Pi = (\Gamma, \delta)$ and a list $\mathcal{M} = (M_1, \ldots, M_m)$ of markings in Π, returns a treed circuit $\Pi' = (\Gamma', \delta')$ and a list $\mathcal{M}' = (M'_1, \ldots, M'_m)$ of compact markings in Π' such that $\varepsilon(M_i) = \varepsilon(M'_i)$ $(1 \le i \le m)$.* MakeTree *takes $\mathcal{O}(n^2 + n \cdot m)$ time and the size of Γ' is bounded by $4|\Gamma|$.* □

Remark 3.33. *Seen from the outside,* ExtendReduction *and* ExtendTree *as well as* Reduce *and* MakeTree *behave very much alike. In applications, all four procedures are used as "black boxes" and of the resulting circuits only the weaker property of reducedness is used. Therefore, in order to simplify nomenclature, we will speak of "reduction" and "reduced circuits", subsuming both concepts. The reader may then choose whether to use the simpler reduction concept at the cost of logarithmic factors or go for the more complicated procedures for treed circuits with better asymptotic time complexity.*

Algorithm 3: Procedure ExtendTree

input : a graph $\Pi = (\Gamma \dot\cup U, \delta)$ with $\delta|_{\Gamma\times U} = 0$ and $(\Gamma, \delta|_{\Gamma\times\Gamma})$ a treed power circuit, a list $\mathcal{M} = (M_1, \dots M_m)$ of markings in Π. For all $M \in \mathcal{M} \cup \{\Lambda_u : u \in U\}$, the power sum $\varepsilon(M|_\Gamma)$ must be compact.

output: a treed power circuit $\Pi' = (\Gamma', \delta')$ with $\Gamma \subseteq \Gamma'$ and $\delta'|_{\Gamma\times\Gamma'} = \delta|_{\Gamma\times\Gamma}$, a list $\mathcal{M}' = (M'_1, \dots, M'_m)$ of compact markings in Π' such that $\varepsilon(M_i) = \varepsilon(M'_i)$

foreach *marking* $M \in \mathcal{M} \cup \{\Lambda_u : u \in U\}$ *with* $\operatorname{supp} M \subseteq \Gamma$ **do**
    CompactifyMarking(M)
Compute a topological order $U = (u_1, \dots, u_k)$.
for $i = 1, \dots, k$ **do**
    $U := U \setminus \{u_i\}$
    if $\Gamma = \varnothing$ **then** set $\Gamma := \{u_1\}$, create a tree for u_1 and insert all markings with support $\{u_1\}$. Continue with $i = 2$.
    Traversing the lowest level of the tree, find the first node v_j in the ordered list $\Gamma = (v_1, v_2, \dots)$ such that $\varepsilon(u_i) \le \varepsilon(v_j)$.
    if $\varepsilon(\Lambda_{u_i}) < 0$ **then** abort the algorithm.
    if $\varepsilon(u_i) < e(v_j)$ *(or no such* v_j *exists)* **then**
        Insert u_i into Γ using InsertNode.
    else $\varepsilon(u_i) = \varepsilon(v_j)$
        Prolong the maximal chain $\dots, v_j, v_{j+1}, \dots, v_k$ by a new node by creating a copy of Λ_{v_k} and applying IncrementMarking and InsertNode.
        foreach *marking* M *with* $u_i \in \operatorname{supp} M$ **do**
            Replace u_i in M with v_j by adding $M(u_i)$ to $M(v_j)$ and setting $M(u_i) := 0$. If, after this, $M(v_j) \notin D$, add $\pm q$ to $M(v_j)$ and ± 1 to $M(v_{j+1})$. Repeat if $M(v_{j+1}) \notin D$ and so on.
    repeat
        Prolong the maximal chain starting at v_j using IncrementMarking and InsertNode
    until *the newly created node is the top of the chain*
    foreach *marking* $M \in \mathcal{M} \cup \{\Lambda_u : u \in U\}$ *of which* u_i *was the last node not in* Γ **do**
        CompactifyMarking(M)

Chapter 4

The Word Problem in Extensions of $\mathrm{BS}(1, q)$

In this chapter, we apply the data structures and algorithms developed in Chapter 3 to algorithmic group theory. We show that the word problem for two classes of groups – generalized forms of the Baumslag-Gersten groups and Higman's groups – is decidable in polynomial time.

4.1 $\mathrm{BS}(1, q)$ and Swapping

All groups discussed in this chapter are extensions of the solvable Baumslag-Solitar group $\mathrm{BS}(1, q) = \langle a, t \mid tat^{-1} = a^q \rangle$. This group is isomorphic to the semidirect product $\mathbb{Z}[1/q] \rtimes \mathbb{Z}$, where $\mathbb{Z}[1/q]$ is the additive group of quotients with powers of q as denominators and $\mathbb{Z}$ acts on this group via multiplication by q. The elements of $\mathbb{Z}[1/q] \rtimes \mathbb{Z}$ can be written in a unique way as pairs $(u, k) \in \mathbb{Z}[1/q] \times \mathbb{Z}$ with the following formulae for multiplication and inversion:

$$\begin{aligned} (u, k)(v, \ell) &= (u + v \cdot q^k, k + \ell) \\ (u, k)^{-1} &= (-u \cdot q^{-k}, -k) \end{aligned} \tag{4.1}$$

The isomorphism $\mathrm{BS}(1, q) \to \mathbb{Z}[1/q] \rtimes \mathbb{Z}$ is given by $a \mapsto (1, 0)$, $t \mapsto (0, 1)$. The inverse of this map is $(u, k) \mapsto t^x a^v t^{k-x}$ where $u = q^x \cdot v$, $v \in \mathbb{Z}$, and $x \leq 0$ is maximal. Note that we avoid non-integer exponents like in $a^{1/q}$ by writing $t^{-1}at$ instead. We formalize this technique in the following definition:

Definition 4.1. *We define* $[u, x, k] := (u \cdot q^x, x + k) \in \mathbb{Z}[1/q] \rtimes \mathbb{Z} = \mathrm{BS}(1, q)$ *for* $u, x, k \in \mathbb{Z}$, $x \leq 0 \leq k$ *and call* $[u, x, k]$ *a* triple.

If U, X, *and* K *are markings in a power circuit with base* q *and* $\varepsilon(X) \leq 0$, $\varepsilon(K) \geq 0$, *we call* $T = [U, X, K]$ *a* triple marking *and define its value* $\varepsilon(T) := [\varepsilon(U), \varepsilon(X), \varepsilon(K)] \in \mathrm{BS}(1, q)$.

Each element of $\mathrm{BS}(1,q)$ can be written as a triple, but not in a unique way. For instance, $[2,0,0] = (2,0) = [4,-1,1]$ in $\mathrm{BS}(1,2)$.

The group operations (4.1) translate to formulae for multiplication and inversion of triples:

$$\begin{aligned}[u,x,k]\cdot[v,y,\ell] &= [u\cdot q^{-y}+v\cdot q^{k}, x+y, k+\ell]\\ [u,x,k]^{-1} &= [-u,-k,-x]\end{aligned} \tag{4.2}$$

Furthermore, $[u,x,k] \in \langle a\rangle \leq \mathrm{BS}(1,q)$ if and only if $x = -k$ and $u\cdot q^x \in \mathbb{Z}$, in which case $[u,x,k] = [u\cdot q^x, 0, 0]$. Similarly, $[u,x,k] \in \langle t\rangle \leq \mathrm{BS}(1,q)$ if and only if $u = 0$, and finally $[u,x,k]$ is the identity of the group if and only if $u = 0$ and $x = -k$.

For the Baumslag-Solitar group $\mathrm{BS}(1,q)$ alone, the use of triple markings in power circuits is not necessary. As we have seen in Chapter 2, binary representation of integers offers sufficient compression. This changes when we introduce another operation beside those given by the group.

Definition 4.2. *In* $\mathrm{BS}(1,q)$, swap *is the partially defined operation given by*

$$\mathrm{swap}(u,k) \coloneqq (k,u) \qquad \text{if } u\in\mathbb{Z}$$

or equivalently for triples

$$\mathrm{swap}[u,x,k] \coloneqq \begin{cases}[(x+k)\cdot q^{-u\cdot q^x}, u\cdot q^x, 0] & \text{if } 0\geq u\cdot q^x\in\mathbb{Z}\\ [x+k, 0, u\cdot q^x] & \text{if } 0\leq u\cdot q^x\in\mathbb{Z}.\end{cases} \tag{4.3}$$

Example 4.3. *Using* swap*, we can define*

$$\begin{aligned} w_0 &= (1,0) \quad \text{and}\\ w_{n+1} &= \mathrm{swap}(w_n)\cdot(1,0)\cdot\mathrm{swap}(w_n)^{-1} \quad \text{for } n\geq 0.\end{aligned}$$

The length of the terms w_n (as a sequence of tuples and operators) is exponential in n. Yet, $w_n \sim (\mathrm{tow}_q(n),0)$ in $\mathrm{BS}(1,q)$. Hence, if we want to compute normal forms of expressions containing swap*, we need higher compression for integers.*

The next result shows that power circuits allow us to solve the word problem in $\mathrm{BS}(1,q)$ despite the complexity added by the swap operation. At the same time, this is a first demonstration of how to use power circuits in specific applications. The techniques exhibited here will be reused in the next section.

Algorithm 4: Algorithm for the word problem in BS$(1,q)$ with swap

input : a term w consisting of letters a, t, operators $\cdot, ^{-1}$, swap, and brackets

output: "True" if $w \sim 1$ in BS$(1,q)$, otherwise "False"

1 Create a base q power circuit Π with one node v (which has $\varepsilon(v) = 1$).
2 For each occurrence of a letter a or t in w, create a new marking M with $\operatorname{supp} M = \{v\}$ and $M(v) = +1$. Replace the letter by $[M, 0, 0]$ (if it was an a) or by $[0, 0, M]$ (if it was a t).
3 REDUCE$(\Pi, \mathcal{M})$ (where $\mathcal{M}$ contains all the markings created so far)
4 **while** $|w| > 0$ **do**
5 Observing brackets and operator precedence, find in w an occurrence of $v \in \{[U,X,K]\cdot[V,Y,L], [U,X,K]^{-1}, \mathrm{swap}([U,X,K])\}$.
6 **if** *none was found* **then return** *False*
7 **if** $v = [U,X,K]\cdot[V,Y,L]$ **then**
8 Create clones U', X', K', V', Y', L' of the respective markings. Compute markings $W = U' \cdot q^{-Y'} + V' \cdot q^{K'}$, $Z = X' + Y'$, and $M = K' + L'$ using the algorithms from Section 3.2. Note that no further cloning is needed. Replace v in w by $[W, Z, M]$.
9 **if** $v = [U,X,K]^{-1}$ **then**
10 Create clones U', X', K' and markings $W = -U', Z = -K', M = -X'$ and replace v by $[W, Z, M]$.
11 **if** $v = \mathrm{swap}([U,X,K])$ **then**
12 Test whether $q^{-\varepsilon(X)}$ divides $\varepsilon(U)$ (see Corollary 3.10) and if not, **return** *False*. Otherwise, depending on the sign of $\varepsilon(U)$, create clones U', X', X'', K' and markings $W = X' + K'$ and $Z = U' \cdot q^{X'}$ or, in case $\varepsilon(U) < 0$, $W = (K' + X') \cdot q^{Z}$. Replace v by $[W, 0, Z]$ or $[W, Z, 0]$, accordingly.
13 Call EXTENDREDUCTION$(\Gamma, D, \mathcal{M})$, where D is the set containing all clones created during this iteration and $\mathcal{M}$ contains the newly created markings W, Z, M. Check if the newly inserted triple evaluates to the identity in BS$(1,q)$ and if so, remove it from w.
14 **return** *True*

Theorem 4.4. *(cf. [DLU13], Thm. 15) Algorithm 4 decides the word problem for the algebraic structure* BS$(1,q)$ *with multiplication, inversion and swapping in* $\mathcal{O}(|w|^4)$ *time.*

Proof. We work on a sequence w of triple markings in a power circuit and

operators (and possibly brackets). Between cycles of the main loop (starting at step 4), we keep up the following invariants: the circuit Π is reduced, no triple marking in w equals the identity in $\mathrm{BS}(1,q)$, and the group element represented by w does not change.

In order to establish these invariants, the letters of the input are rewritten as triple markings. This creates a circuit with one node and at most $n = |w|$ markings. Reducing this circuit takes $\mathcal{O}(n^2)$ time.

In the main loop, w is shortened as much as possible. The replacements reflect the rules for multiplication, inversion, and swapping in $\mathrm{BS}(1,q)$ which we introduced earlier in this chapter. There are at most n cycles of the main loop, since each cycle decreases the length of w. On the power circuit level, we need a constant number of basic operations (addition, multiplication by a power of q, comparison) in each cycle of the main loop. Note that we use clones for all arithmetic operations and call ExtendReduction immediately afterwards, hence Γ stays reduced all the time. In order to get an upper bound for the time complexity, we first need to estimate the growth of Π.

We call the sum $\omega = \sum |\mathrm{supp}\, M|$, taken over all markings M in w, the *weight* of Π. During the replacements in the main loop, ω never increases. For example, when replacing $[U,X,K]\cdot[V,Y,L]$ by $[W,Z,M]$, we have $|\mathrm{supp}\, W| \le |\mathrm{supp}\, U| + |\mathrm{supp}\, V|$, $|\mathrm{supp}\, Z| \le |\mathrm{supp}\, X| + |\mathrm{supp}\, Y|$, and $|\mathrm{supp}\, M| \le |\mathrm{supp}\, K| + |\mathrm{supp}\, L|$. Similar inequalities hold for the other two cases. Hence, the number of non-reduced nodes $|D|$ for each call of ExtendReduction is in $\mathcal{O}(\omega) = \mathcal{O}(n)$. Therefore, the circuit size remains bounded by $\mathcal{O}(n^2)$ and ExtendReduction needs $\tilde{\mathcal{O}}(n^2 \cdot n)$ time. All the other operations in the main loop are linear time, so we get an overall bound of $\tilde{\mathcal{O}}(n^4)$.

The stronger precondition of ExtendTree is also satisfied: the markings W, Z, M are completely outside Γ and the successor markings of their nodes are either exact copies of (compact) markings inside Γ (e.g. in step 8 for $m \in \mathrm{supp}\, M$, $\Lambda_m = \Lambda_v$ with $v \in \mathrm{supp}\, K$ or $v \in \mathrm{supp}\, L$) or the union of such markings and nodes outside Γ (e.g. in step 12 for $z \in \mathrm{supp}\, Z$, $\Lambda_z = \Lambda_u + X'$ with $u \in \mathrm{supp}\, U$). Hence we get the time bound $\mathcal{O}(n^4)$. □

4.2 Higman's Groups $H_f(1,q)$

The group H_4 was introduced by Higman in 1951 and served to provide the first known example of a finitely generated infinite simple group [Hig51]. Such a group can be found by taking a minimal non-trivial quotient of H_4, since H_4 is infinite and has no non-trivial normal subgroups of finite index.

The group H_4 belongs to a family of groups H_f ($f \ge 1$) with f generators

and f relators:

$$H_f = \langle a_1, \ldots, a_f \mid a_{i+1} a_i a_{i+1}^{-1} = a_i^2 \ (i \in \mathbb{Z}/f\mathbb{Z}) \rangle$$

For $f < 4$ these groups are trivial, which is easy to see for $f \in \{1, 2\}$, but surprisingly hard for $f = 3$. The latter case was proven by Hirsch [Hig51], see also §23 in [Neu54] and [All12]. If $f \geq 4$, then H_f is infinite ([Ser02], Sect. 1.4).

Until recently, Higman's Group H_4 was a candidate for a "natural" group with non-elementary word problem. This was suggested by the huge compression that this group allows. In fact, the words $w(i, j)$ defined by

$$\begin{aligned} w(i, 0) &:= a_i \quad \text{and} \\ w(i, j+1) &:= w(i+1, j)\, a_i\, w(i+1, j)^{-1} \end{aligned}$$

have exponential length in j, yet $w(i, j) \sim a_i^{\mathrm{tow}_2(j)}$. Thus any algorithm solving the word problem by computing normal forms like a_i^n, needs $\mathrm{tow}_2(\Theta(n))$ space, even when using binary notation. Despite this, it was shown in [DLU12] that the word problem for H_4 is decidable in $\mathcal{O}(n^6 \cdot \log n)$ time and [DLU13] improved this bound to $\mathcal{O}(n^6)$. Both results rely on power circuits.

Following [Lau12], we generalize H_f even further by replacing the underlying Baumslag-Solitar group $\mathrm{BS}(1, 2)$ with $\mathrm{BS}(1, q)$. As we have seen in Section 4.1, power circuits with base q are naturally suited for computations in this group.

Definition 4.5. *The* generalized Higman group $\mathrm{H}_f(1, q)$ *is defined as*

$$\mathrm{H}_f(1, q) = \langle a_1, \ldots, a_f \mid a_{i+1} a_i a_{i+1}^{-1} = a_i^q \ (i \in \mathbb{Z}/f\mathbb{Z}) \rangle. \tag{4.4}$$

While for $f > 4$, $H_f = \mathrm{H}_f(1, 2)$ retains the essential properties of H_4, this is not true for $\mathrm{H}_f(1, q)$ in general. For example, for all $f \geq 1$, the map

$$\begin{aligned} \mathrm{H}_f(1, 3) &\twoheadrightarrow \mathbb{Z}/2\mathbb{Z}; \\ a_1 &\mapsto 1, \\ a_2, \ldots, a_f &\mapsto 0 \end{aligned}$$

sends $\mathrm{H}_f(1, 3)$ onto a finite non-trivial group.

In this section we will prove:

Theorem 4.6. *([Lau12], Thm. 4.2) Let $q \geq 2, f \geq 4$. The word problem for the generalized Higman group $\mathrm{H}_f(1, q)$ can be solved in $\mathcal{O}(n^6)$ time.*

The key observation for the solution of the word problem is the decomposition of $\mathrm{H}_f(1,q)$ into a series of amalgamations of f copies of the Baumslag-Solitar group $\mathrm{BS}(1,q) = \langle a,t \mid tat^{-1} = t^q \rangle$, see [Ser02]:

$$\mathrm{H}_f(1,q) = G_{1,\dots,f-1} *_{F_{1,f-1}} G_{f-1,f,1},$$

where

$$G_{1,\dots,f-1} = \langle a_1,\dots,a_{f-1} \mid a_{i+1}a_ia_{i+1}^{-1} = a_i^q \ (1 \le i < f-1)\rangle \text{ and}$$
$$G_{f-1,f,1} = \langle a_{f-1}, a_f, a_1 \mid a_f a_{f-1} a_f^{-1} = a_{f-1}^q,\ a_1 a_f a_1^{-1} = a_f^q \rangle.$$

In both cases $F_{1,f-1}$ is the subgroup generated by a_1 and a_{f-1} which in fact freely generate $F_{1,f-1}$ (since $f \ge 4$). Furthermore, we can break $G_{1,\dots,f-1}$ and $G_{f-1,f,1}$ down to

$$G_{1,\dots,f-1} = G_{1,2} *_{F_2} G_{2,3} *_{F_3} \dots *_{F_{f-2}} G_{f-2,f-1} \text{ and}$$
$$G_{f-1,f,1} = G_{f-1,f} *_{F_f} G_{f,1},$$

where $G_{i,i+1} = \langle a_i, a_{i+1} \mid a_{i+1}a_ia_{i+1}^{-1} = a_i^q \rangle$, $F_i = \langle a_i \rangle$ and the indices are read in $\mathbb{Z}/f\mathbb{Z}$.

Each group $G_{i,i+1}$ is a copy of the Baumslag-Solitar group $\mathrm{BS}(1,q)$ and thus isomorphic to the semidirect product $\mathbb{Z}[1/q] \rtimes \mathbb{Z}$. Since there are several copies of $\mathrm{BS}(1,q)$ involved, we endow the pairs and triples from Section 4.1 with a subscript:

$$[u,x,k]_i = (u \cdot q^x, x+k)_i = a_{i+1}^x a_i^u a_{i+1}^k \in G_{i,i+1}$$

In order to solve the word problem for $\mathrm{H}_f(1,q)$, we first need a solution for the subgroup membership problem of $F_{1,e}$ in $G_{1,\dots,e}$ (with $e \ge 3$; this covers both $G_{1,\dots,f-1}$ and $G_{f-1,f,1}$). Furthermore, we have to do this in an effective way, i.e., given a sequence of pairs $(u,k)_i$ representing an element of $F_{1,e}$, we have to find a corresponding sequence of pairs of the form $(u,0)_1$ and $(0,\ell)_{e-1}$.

We start by giving a reduction system $\mathcal{L}$ for $G_{1,\dots,e}$:

(1)	$(u,k)_i(v,\ell)_i \longrightarrow (u + v\cdot q^k, k+\ell)_i$	for $1 \le i \le e-1$
(2)	$(u,k)_i(v,0)_{i+1} \longrightarrow (u, k+v)_i$	for $1 \le i < e-1$ and $v \in \mathbb{Z}$
(3)	$(u,0)_{i+1}(v,\ell)_i \longrightarrow (v\cdot q^u, \ell+u)_i$	for $1 \le i < e-1$ and $u \in \mathbb{Z}$
(4)	$(u,k)_{i+1}(0,\ell)_i \longrightarrow (u+\ell\cdot q^k, k)_{i+1}$	for $1 \le i < e-1$
(5)	$(0,k)_i(v,\ell)_{i+1} \longrightarrow (k+v, \ell)_{i+1}$	for $1 \le i < e-1$

On the right hand sides of these rules, we identify the pair $(0,0)_i$ with the empty sequence 1, so for example $(0,1)_i(-1,0)_{i+1} \longrightarrow 1$ is an instance of rule (2).

The system $\mathcal{L}$ is not confluent, but it has the Britton property of being confluent on strings that equal the group identity. Note that in general, Britton reductions do not work for iterated amalgamated products like $G_{1,\dots,e}$. However, in this case the amalgamated subgroups are disjoint due to the structure of the Baumslag-Solitar group, and we get:

Proposition 4.7. *If w is an $\mathcal{L}$-reduced word that equals 1 in $G_{1,\dots,e}$, then w is the empty word.* □

Let $\mathcal{L}'$ be the system $\mathcal{L}$ extended by the rules

$$(6)\quad (x_1,-x_2)_1(x_2,-x_3)_2\dots(x_{e-2},-x_{e-1})_{e-2}(\widetilde{x}_{e-1},x_e)_{e-1} \longrightarrow (x_1,0)_1(\widetilde{x}_{e-1}-x_{e-1},x_e)_{e-1}$$

$$(7)\quad (-x_{e-1}\cdot q^{x_e},x_e)_{e-1}(-x_{e-2}\cdot q^{x_{e-1}},x_{e-1})_{e-2}\dots(-x_2\cdot q^{x_3},x_3)_2(x_1\cdot q^{x_2},\widetilde{x}_2)_1 \longrightarrow (0,x_e)_{e-1}(x_1,\widetilde{x}_2-x_2)_1,$$

where all $x_i \neq 0$.

The new rules respect the group structure and hence Proposition 4.7 holds for $\mathcal{L}'$ as well. They are not length-increasing, since $e \geq 3$.

Starting with an arbitrary sequence w of pairs $(u,k)_i$ representing an element in $G_{1,\dots,e}$, one can compute an equivalent $\mathcal{L}'$-reduced word $\hat{w}$ with linearly many operations: First, compute an $\mathcal{L}$-reduced word $\tilde{w}$, then apply rules (6) and (7). Note that the latter leave $\tilde{w}$ $\mathcal{L}$-reduced. The rules (6) and (7) can be applied in one pass from left to right, since the left-most application of any of the two prolongs the $\mathcal{L}'$-reduced prefix of $\tilde{w}$.

Proposition 4.8. *([Lau12], Prop. 4.4) Let w be a sequence of pairs $(u,k)_i$ representing an element of the subgroup $F_{1,e} \leq G_{1,\dots,e}$. If w is $\mathcal{L}'$-reduced, then w is an alternating sequence of pairs of types $(u,0)_1$ and $(0,\ell)_{e-1}$.*

Proof. Let $w = (u_1,k_1)_{i_1}(u_2,k_2)_{i_2}\dots(u_n,k_n)_{i_n}$. We assume that

$$w \sim \tilde{w} = (v_1,0)_1(0,\ell_1)_{e-1}(v_2,0)_1(0,\ell_2)_{e-1}\dots \in F_{1,e}$$

and that $\tilde{w}$ contains no trivial pairs $(0,0)_i$ which makes $\tilde{w}$ $\mathcal{L}$-reduced. The case where $\tilde{w}$ starts with $(0,\ell_1)_{e-1}(v_1,0)_1\dots$ is similar. The sequence

$$\tilde{w}^{-1}w = \dots(-v_2,0)_1(0,-\ell_1)_{e-1}(-v_1,0)_1(u_1,k_1)_{i_1}(u_2,k_2)_{i_2}(u_3,k_3)_{i_3}\dots$$

equals 1 in $G_{1,\dots,e}$ and must therefore $\mathcal{L}$-reduce to the empty sequence. Note that both $\tilde{w}^{-1}$ and w are $\mathcal{L}$-reduced, so any $\mathcal{L}$-reduction can only occur at the border between the two words.

Clearly, we cannot have $i_1 \geq 3$ or else $\tilde{w}^{-1}w$ would be $\mathcal{L}$-reduced. If $i_1 = 2$, then a reduction of type (2) is possible if $k_1 = 0$ and $u_1 \in \mathbb{Z}$, in which case we get $(-v_1, 0)_1(u_1, 0)_2 \xrightarrow{(2)} (-v_1, u_1)_1$. But after that, the sequence is $\mathcal{L}$-reduced since $k_1 = 0$ and $u_1 \in \mathbb{Z}$ imply $i_2 \neq 1$.

Hence, we are left with $i_1 = 1$. In that case, we get $(-v_1, 0)_1(u_1, k_1)_1 \xrightarrow{(1)} (-v_1 + u_1, k_1)_1$. If this is $(0, 0)_1$, we have $(u_1, k_1)_{i_1} = (v_1, 0)_1$ and we proceed inductively with the remaining sequence. Otherwise, we must have $u_1 = v_1$ in order to continue applying rules. If $e \geq 4$, the next rule can only apply to $(0, k_1)_1(u_2, k_2)_{i_2}$, so $i_2 = 2$ and we get $(0, k_1)_1(u_2, k_2)_2 \xrightarrow{(5)} (k_1 + u_2, k_2)_2$. Again, the sequence is $\mathcal{L}$-reduced unless $u_2 = -k_1$. We iterate this argument until we arrive at

$$\tilde{w}^{-1}w \overset{*}{\underset{\mathcal{L}}{\Longrightarrow}} \dots (-v_2, 0)_1(0, -\ell_1)_{e-1}(0, k_{e-2})_{e-2}(u_{e-1}, k_{e-1})_{i_{e-1}} \dots .$$

On the way, we have found $u_1 = v_1, u_2 = -k_1, u_3 = -k_2, \dots, u_{e-2} = -k_{e-3}$, and $i_j = j$ for $1 \leq j \leq e-2$. One further reduction of type (4) brings us to

$$\tilde{w}^{-1}w \overset{*}{\underset{\mathcal{L}}{\Longrightarrow}} \dots (-v_2, 0)_1(k_{e-2} \cdot q^{-\ell_1}, -\ell_1)_{e-1}(u_{e-1}, k_{e-1})_{i_{e-1}} \dots .$$

Since $\ell_1 \neq 0$, the next reduction requires $i_{e-1} = e-1$. Thus, rule (6) of $\mathcal{L}'$ can be applied to the prefix $(u_1, k_1)_1 \dots (u_{e-1}, k_{e-1})_{e-1}$ of the original word w. □

Proposition 4.8 shows that the system $\mathcal{L}'$ can solve the subgroup membership problem with respect to $F_{1,e} \leq G_{1,\dots,e}$.

For the amalgamated product $\mathrm{H}_f(1,q) = G_{1,\dots,f-1} *_{F_{1,f-1}} G_{f-1,f,1}$, Theorem 1.4 can be stated as follows:

Proposition 4.9. *Let $w = w_1 w_2 \dots w_s$ be a non-empty sequence with $w_i \in \{(u,k)_i : 1 \leq i < f-1\}^*$ or $w_i \in \{(u,k)_i : f-1 \leq i \leq f\}^*$, alternatingly. If w equals 1 in $\mathrm{H}_f(1,q)$, then $w_i \in F_{1,f-1}$ for some index i.* □

From this proposition, we can derive Algorithm 5 which solves the word problem in $\mathrm{H}_f(1,q)$. In this algorithm, the word $w = w_1 w_2 \dots w_s$ is split into $w_1 \dots w_t$ and $w_{t+1} \dots w_s$. The first part is an alternating sequence of group elements from $G_{1,\dots,f-1}$ and from $G_{f-1,f,1}$, and no w_i with $i \geq t$ is in the subgroup $F_{1,f-1}$. In each loop cycle either t increases, or t decreases by one and s decreases. Thus, the loop is executed only linearly often.

In order to get a time bound for Algorithm 5, it remains to show how the tests and arithmetic operations on the pairs $(u,k)_i$ are to be performed in an efficient way. This is similar to Algorithm 4.

At the beginning of Algorithm 5, we create a power circuit with base q consisting of a single node v with $\varepsilon(v) = 1$. We represent each pair $(u,k)_i$ by

Algorithm 5: Procedure for solving the word problem in $\mathrm{H}_f(1,q)$

input : a word w over a_i, a_i^{-1} $(1 \le i \le f)$
output: the answer to $w \stackrel{?}{\sim} 1$ in $\mathrm{H}_f(1,q)$

Rewrite the input w by replacing each $a_i^{\pm 1}$ with $(\pm 1, 0)_i$.
Break w into subsequences $w = w_1 w_2 \dots w_s$ such that in each w_j the subscripts of all pairs are either in $\{1, 2, \dots, f-2\}$ or in $\{f-1, f\}$.
Let $t := 0$.
while $(t = 0 \wedge s > 1) \vee (0 < t < s)$ **do**
    if $t = 0 \wedge s > 1$ **then**
        $\mathcal{L}'$-reduce w_1. If w_1 becomes empty, remove it (thereby decreasing s) and continue with the next iteration.
        if $w_1 \in F_{1,f-1}$ **then**
            Merge w_1 and w_2. Before doing so, if w_1 and w_2 are from different groups (one from $G_{1,\dots,f-1}$ and the other one from $G_{f-1,f,1}$), swap all the pairs in w_1 using the following rules: $(x,0)_1 \leftrightarrow (0,x)_f$ and $(0,x)_{f-2} \leftrightarrow (x,0)_{f-1}$
        else
            Increment t by one.
    else $(0 < t < s)$
        if *w_t and w_{t+1} are both from $G_{1,\dots,f-1}$ or both from $G_{f-1,f,1}$* **then**
            Merge w_t and w_{t+1}.
            Decrement t by one.
        else
            $\mathcal{L}'$-reduce w_{t+1}. If w_{t+1} becomes empty, remove it and continue with the next iteration.
            if $w_{t+1} \in F_{1,f-1}$ **then**
                Perform the replacements $(x,0)_1 \leftrightarrow (0,x)_f$ and $(0,x)_{f-2} \leftrightarrow (x,0)_{f-1}$ in w_{t+1}, then merge it with w_t.
                Decrement t by one.
            else
                Increment t by one.
return *whether* $s = 0$

a triple marking. Of the three markings in each initial triple, two are empty and the third has value +1 or −1 and can be created using v. Let ω be the weight, i.e., the sum of the sizes of (the supports of) all the markings in w. The initial value of ω is exactly $n = |w|$ and it never increases during the

algorithm.

After step 1, we reduce the circuit, which takes $\mathcal{O}(n^2)$ time. From now on, we keep Π reduced so that tests like "$\varepsilon(M) \leq \varepsilon(K)$?" or "$q^{\varepsilon(M)} \mid \varepsilon(K)$?" are possible at any time.

The last thing we have to discuss is $\mathcal{L}'$-reduction of words. For each operation (addition or multiplication by a power of q), the involved markings are cloned, the operation is performed on the clones and finally, the circuit is reduced using ExtendReduction. The set U for this call contains at most $\mathcal{O}(\omega)$ nodes and the list $\mathcal{M}$ contains only a constant number of markings (the results of the operation). The circuit size $|\Gamma|$ grows by $\mathcal{O}(\omega)$. The whole algorithm performs $\mathcal{O}(n)$ $\mathcal{L}'$-reductions, each of which applies $\mathcal{O}(n)$ rules from $\mathcal{L}'$. Each such rule in turn necessitates a constant number of calls of ExtendReduction (these dominate the time complexity). The circuit size remains bounded by $\mathcal{O}(n^2 \cdot \omega) \subseteq \mathcal{O}(n^3)$. Thus, one call of ExtendReduction takes $\mathcal{O}(n^3 \cdot \omega) \subseteq \mathcal{O}(n^4)$ time. We get a total time bound of $\mathcal{O}(n^6)$.

This concludes the proof of Theorem 4.6.

Remark 4.10. *The constant hidden in the $\mathcal{O}$ notation in Theorem 4.6 can be made independent of f. If f is larger than the size of the input string w, some letter (a_f, say) does not occur in w. In this case, we can solve the word problem in the group $G_{1,\dots,f-1}$ instead of $\mathrm{H}_f(1,q)$. If even more letters are missing, we can replace $G_{1,\dots,f-1}$ by some free product $*_{i=1}^{n} G_{1,\dots,e_i}$ with $\sum e_i \in \mathcal{O}(|w|)$.*

4.3 Baumslag-Gersten Groups

Definition 4.11. *Let $q \geq 2$. The Baumslag-Gersten group $\mathrm{BG}(1,q)$ is the one-relator group defined by*

$$\mathrm{BG}(1,q) = \langle a, b \mid a^{a^b} = a^q \rangle = \langle a, b \mid (bab^{-1})a(bab^{-1})^{-1} = a^q \rangle \\ \simeq \langle a, b, t \mid bab^{-1} = t, tat^{-1} = a^q \rangle.$$

This is a generalization of the notion usually found in the literature, which focuses on $\mathrm{BG}(1,2)$. Sometimes this group is just called the "Baumslag group", for example in [MUW11]. The group was introduced by Baumslag in 1969 to serve as an example of a non-cyclic group all of whose finite quotients are cyclic.

Like $\mathrm{H}_f(1,q)$, the group $\mathrm{BG}(1,q)$ allows compression of tower function magnitude: the words T_n, defined by $T_0 = t$ and $T_{n+1} = bT_naT_n^{-1}b^{-1}$ have exponential length in n, but $T(n) \sim t^{\mathrm{tow}_q(n)}$. It can be shown that the Dehn function of $\mathrm{BG}(1,2)$ is non-elementary ([Ger91], §6). More precisely, it grows

like $\mathrm{tow}_2(\log n)$ [Pla04]. This implies that the group is neither automatic nor hyperbolic.

The word problem for $\mathrm{BG}(1,2)$ was first proved to be polynomial time solvable in [MUW11]. This was surprising, given the huge growth of the Dehn function. The time bound shown in [MUW11] is $\mathcal{O}(n^7)$. In [DLU12] this was reduced to $\tilde{\mathcal{O}}(n^3)$, and [DLU13] finally brought the complexity down to $\mathcal{O}(n^3)$. In this Section we will prove:

Theorem 4.12. *(cf. [MUW11], Thm. 5.5 and [DLU12], Thm. 9) Let $q \geq 2$. The word problem for the Baumslag-Gersten group $\mathrm{BG}(1,q)$ is solvable in $\mathcal{O}(n^3)$ time.*

An interesting thing about the algorithm for the word problem of $\mathrm{BG}(1,q)$ is that we deviate substantially from the basic idea of the last two sections. Instead of keeping the power circuit reduced and thus having to clone a lot, we try to keep track of the structure of the graph. This allows us to avoid cloning altogether.

Proof. The group $\mathrm{BG}(1,q)$ can be viewed as an HNN extension of $\mathrm{BS}(1,q)$ with stable letter b:

$$\begin{aligned}\mathrm{BG}(1,q) &\simeq \langle a,t,b \mid tat^{-1} = a^q, bab^{-1} = t\rangle \\ &\simeq \mathrm{HNN}(\langle a,t \mid tat^{-1} = a^q\rangle, b, \langle a\rangle \tilde{\to} \langle t\rangle)\end{aligned}$$

Thus the Britton reductions $ba^nb^{-q} \to t^n$ and $b^{-1}t^nb \to a^n$ and Lemma 1.1 provide a standard way of solving the word problem. We transform this into a rewriting system using the triple notation for $\mathrm{BS}(1,q) = \langle a,t \mid tat^{-1} = a^q\rangle$:

$$\begin{aligned}
[u,x,k]\cdot[v,y,\ell] &\longrightarrow [u\cdot q^{-y} + v\cdot q^k, x+y, k+\ell], \\
b[u,x,k]b^{-1} &\longrightarrow [0,0,u\cdot q^x] && \text{if } x = -k,\ q^{-x} \mid u,\ u \geq 0 \\
b[u,x,k]b^{-1} &\longrightarrow [0,u\cdot q^x,0] && \text{if } x = -k,\ q^{-x} \mid u,\ u < 0 \\
b^{-1}[u,x,k]b &\longrightarrow [x+k,0,0] && \text{if } u = 0
\end{aligned}$$

In fact, this already shows that the word problem is solvable in $\mathcal{O}(n^4)$ time. All we have to do is interpret conjugation by b as $\mathrm{swap}(\cdot)$ and apply Algorithm 4. However, we aim for a cubic time bound.

To this end, we perform rewritings on a sequence w consisting of letters b, b^{-1} and triples $[U,X,K]$, while maintaining the following invariants:

(a) The supports of all markings that appear in w are pairwise disjoint.

(b) For all triples $[U,X,K]$ in w, the nodes in $\mathrm{supp}\,U$ have no incoming edges.

(c) For all triples $[U, X, K]$ in w, all the incoming edges of nodes in $\operatorname{supp} X$ and $\operatorname{supp} K$ originate in $\operatorname{supp} U$. For edges ending in $\operatorname{supp} X$, their label is the negative of the value that X assigns to the target node, i.e., $\delta(u, x) = -X(x)$ for $x \in \operatorname{supp} X$, $u \in \operatorname{supp} U$.

We create a new node v and a marking $M : v \mapsto 1$ for every occurrence of a, a^{-1}, t, or t^{-1} in the input string w and replace it by the triple $[M, 0, 0]$, $[-M, 0, 0]$, $[0, 0, M]$, or $[0, -M, 0]$, respectively. Thus, the initial circuit has no edges and each node is marked by exactly one marking, so the invariants hold. The size is $|\Gamma| = n = |w|$. Our aim is to prove that due to the invariants no cloning is necessary and hence the circuit size remains bounded by n.

For multiplication of two triples $[U, X, K]$ and $[V, Y, L]$, the invariants remain valid: we insert new edges from U to Y (with the correct label) and from V to K. After that, we take the component-wise (disjoint) union of both triples. Figure 4.1 illustrates this operation.

For $b[U, X, K]b^{-1}$ we test whether $\varepsilon(X) = -\varepsilon(K)$ and $q^{-\varepsilon(X)} \mid \varepsilon(U)$ and we detect the sign of $\varepsilon(U)$. This is done by creating a copy of the whole circuit and reducing the copy. After the tests we trash the copy. This might seem wasteful, but it ensures that we keep the invariants which would surely be destroyed during reduction. If both tests are positive, we introduce new edges from U to X. Since existing edges have the negative values of the target nodes, they cancel out. Hence, no cloning is needed. We replace $b[U, X, K]b^{-1}$ by $[0, 0, U]$ or $[0, U, 0]$, depending on the sign of U. The operation is illustrated in Figure 4.2.

Finally, for $b^{-1}[U, X, K]b$, we check whether $\varepsilon(U) = 0$ by reducing a copy of the circuit as before. In the positive case, we can remove all nodes in $\operatorname{supp} U$ from the circuit, since they have no incoming edges and no other marking uses them. This removes all incoming edges of X and K, so the union $X + K$ can be placed in the first component of the new triple $[X + K, 0, 0]$. This operation is shown in Figure 4.3.

Note that the invariants wouldn't hold if we tried to swap a pair $(u, k) \in \mathbb{Z} \times \mathbb{Z}$ with both u and k non-zero. This is why we didn't get the $\mathcal{O}(n^3)$ time bound in Theorem 4.4.

Using the strategy from Lemma 1.1, we need $\mathcal{O}(n)$ arithmetic operations and, more importantly, a linear number of reductions. The latter take $\mathcal{O}(n^2)$ time each, since the circuit size remains bounded by n. This gives an overall time bound of $\mathcal{O}(n^3)$. □

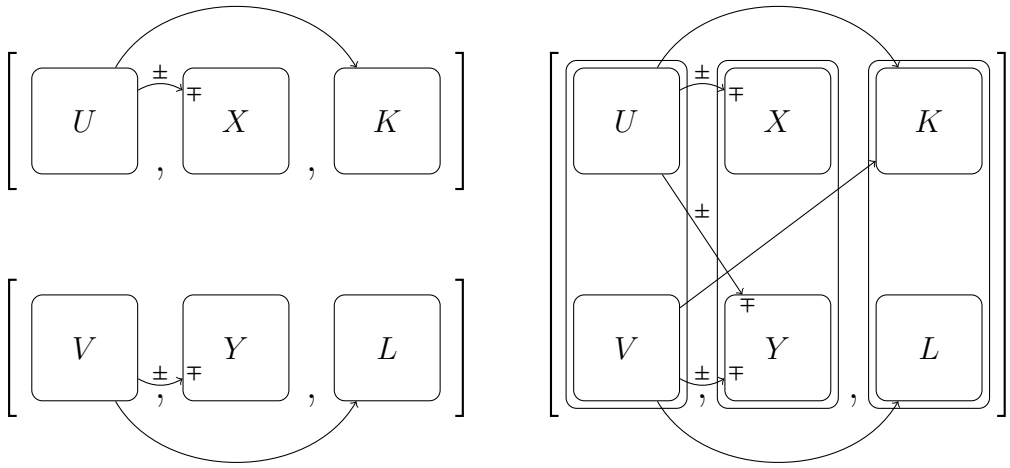


Figure 4.1: Multiplication of the triples $[U, X, K]$ and $[V, Y, L]$

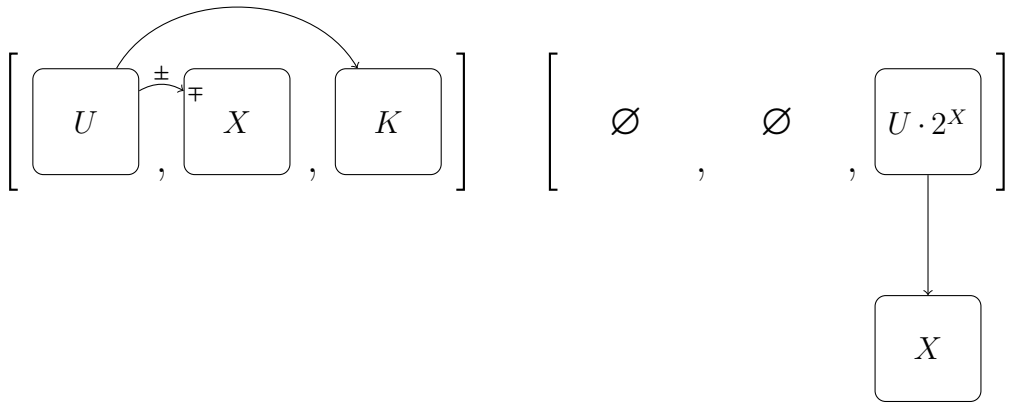


Figure 4.2: Swapping a triple with $\varepsilon(X) + \varepsilon(K) = 0$

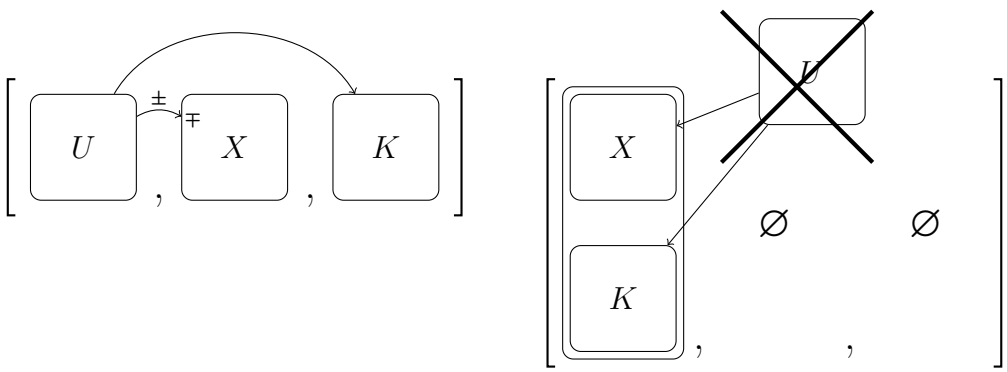


Figure 4.3: Swapping a triple with $\varepsilon(U) = 0$

Chapter 5

Conclusion and Open Questions

We hope to have emphasized the importance of data compression techniques in algorithmic group theory. In this sense, this thesis might be seen as part of a larger effort of applying such methods in group theory, cf. [Loh04, Sch08]. Starting with the exponential compression offered by binary representation, we have proved that the non-trivial problem of finding geodesics in a substantial class of Baumslag-Solitar groups is solvable in polynomial time. This is a large extension of the previously known results which dealt mainly with the solvable case $\mathrm{BS}(1, q)$. The problem remains open for the groups $\mathrm{BS}(p, q)$ with $p \nmid q$. Here we have achieved partial results and identified the difficult cases. However, it seems that entirely new techniques are required for a complete solution of these cases. It is also possible that there is no solution or that the problem is co-NP complete.

In the second part of this thesis, we have introduced power circuits, a data structure that is able to store integers with the magnitude of the tower function. We have implemented arithmetic operations such as addition and multiplication by a power of q and, most importantly, we have given new reduction procedures that turn power circuits into a form that allows efficient comparison of markings. This is both a technical improvement over the original form of power circuits given in [MUW12] as well a conceptual one, since only the putting together of all markings in a single power circuit allows for our improvements in time complexity. Furthermore, we have introduced power circuits with arbitrary base $q \geq 2$ and given counterparts of concepts such as compactness.

As a consequence, we were able to find efficient algorithms for the word problem in groups that are based on $\mathrm{BS}(1, q)$ and have some means of swapping their generators. This leads to huge compression, which was until recently considered a serious obstacle to solving the word problem efficiently. We have given polynomial-time algorithms for two classes of such groups –

the Baumslag-Gersten groups $\mathrm{BG}(1,q)$ and Higman's groups $\mathrm{H}_f(1,q)$. In the latter case, we were also able to generalize the algorithm to arbitrary values $f \geq 4$. Maybe surprisingly, the value of f does not affect the asymptotical complexity of the word problem for $\mathrm{H}_f(1,q)$.

Both the Baumslag-Gersten groups $\mathrm{BG}(1,q)$ and the Higman groups $\mathrm{H}_f(1,q)$ can be generalized even further to $\mathrm{BG}(p,q)$ and $\mathrm{H}_f(p,q)$. Unfortunately, our results do not translate to $p > 1$. On a technical level, this is because the underlying Baumslag-Solitar group $\mathrm{BS}(p,q)$ is not a semi-direct product when $p > 1$. The deeper reason is that power circuits are unable to perform division by integers that are not powers of q (such as p, which would be needed for Britton reductions like $ta^{n \cdot p}t^{-1} \longrightarrow a^{n \cdot q}$). We have seen in Chapter 2 that allowing $p > 1$ can make algorithmic problems in $\mathrm{BS}(p,q)$ much harder.

Although power circuits have so far only been applied to group theory, they might prove useful in other areas as well. The greatest obstacle is probably the small set of arithmetic operations that can be conducted without heavy growth of the circuit. Although operations other than addition and multiplication by a power of q may be possible, there are some serious limitations of power circuits when it comes to dividing numbers, see Section 3.4.2. Another shortcoming of power circuits seems to be the close tie between a circuit and its base q. For instance, we have no efficient way of converting a marking in a base q power circuit to one with the same value in another circuit with a different base q'.

Although a real-world implementation of power circuits requires a substantial amount of overhead, they can and have been got up and running. In particular, the algorithm for the word problem for $\mathrm{BG}(1,2)$ with the quite moderate $\mathcal{O}(n^3)$ time complexity has been implemented. Details can be found in the appendix.

Appendix

This chapter collects experimental work and results.

Horocyclic Growth in $\mathrm{BS}(2,3)$

Algorithm 5.1 is an implementation of the procedure from Section 2.4. It computes the first coefficients of the growth series of the horocyclic subgroup of $\mathrm{BS}(2,3)$. Details can be found in Section 2.4.1. The results are shown in Figure 5.1.

Algorithm 5.1: horo23.cpp

```
1  #include<iostream>
2  #include<string>
3  #include<cstring>
4  #include<cmath>
5  #include<cassert>
6  #include<pthread.h>
7
8  /* maximum length of words */
9  const unsigned int MAX_WORD_LENGTH = 139;
10
11 /* number of threads */
12 const unsigned int NUM_THREADS = 24;
13
14 /* size of chunks that a thread is given at a time */
15 const unsigned long CHUNK_SIZE = 1<<25;
16
17 /* maximum value of horocyclics */
18 const long MAX_HORO_VAL = (long)std::ceil(8*std::pow(3.0/2.0,
       MAX_WORD_LENGTH/2-1));
19
20 /* maximum height of slopes */
21 const unsigned int MAX_SLOPE_HEIGHT = MAX_WORD_LENGTH / 2 +
      5;
22
23 /* geodesic lengths of horocyclics elements 0,...,13 */
```

```
24 int small[14] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 8, 9, 10, 10,
       11};
25
26 /* Function for computing geodesic lengths of slopes that
       occur in shortlex normal forms of horocyclic elements.
       Assumes that the slope has the form "a_0 T a_1 T ... T a_h
       " with |a_0|<6 and |a_i|<3 for i>0 and |a_0|>=3 (>=4, in
       fact, otherwise it would not be horocyclic) if the height
       is non-zero. */
27 int compute_slope_geod_length(unsigned int height, const int*
       slope)
28 {
29     if(height == 0)
30         return small[slope[0]];
31
32     /* tables for geodesic lengths of u(i,g) with i>0, 0=<g
           =<10 */
33     int t1[14], t2[14];
34     int* table1 = t1;
35     int* table2 = t2;
36     memcpy(table1, small, sizeof(small));
37
38     for(int i = height - 1; i >= 0; i--)
39     {
40         for(int g = 0; g <= 10; g++)
41         {
42             if(g % 3 == 0)
43                 table2[g] = table1[slope[i + 1] + g / 3 * 2]
                       + 2;
44             else
45             {
46                 int div = g / 3, rem = g % 3;
47                 int snf1 = table1[slope[i + 1] + div * 2] + 2
                        + rem;
48                 int snf2 = table1[slope[i + 1] + (div + 1) *
                       2] + 2 + 3 - rem;
49                 table2[g] = (snf1 <= snf2 ? snf1 : snf2);
50             }
51         }
52         /* swap tables */
53         std::swap(table1, table2);
54     }
55     return table1[slope[0]];
56 }
57
58 /* compute geodesic length of a horocyclic element */
59 int compute_horo_geod_length(long horoVal)
60 {
```

```
61      /* prepare a slope suitable for compute_slope_geod_length
            */
62      int slope[MAX_SLOPE_HEIGHT+1];
63      unsigned int height = 0;
64      while(horoVal >= 6)
65      {
66          slope[height++] = (int)(horoVal % 3);
67          horoVal = horoVal / 3 * 2;
68      }
69      slope[height] = (int)horoVal;
70      return compute_slope_geod_length(height, slope);
71  }
72
73  /* helper function assigning tasks to threads */
74  pthread_mutex_t getNextTaskMutex;
75  long            horoCurrent = 1;
76
77  bool getNextTask(long* pHoroFirst, long* pHoroLast)
78  {
79      pthread_mutex_lock(&getNextTaskMutex);
80      if(horoCurrent >= MAX_HORO_VAL)
81      {
82          pthread_mutex_unlock(&getNextTaskMutex);
83          return false;
84      }
85      *pHoroFirst = horoCurrent;
86      horoCurrent = (horoCurrent + CHUNK_SIZE < MAX_HORO_VAL +
            1 ? horoCurrent + CHUNK_SIZE : MAX_HORO_VAL + 1);
87      *pHoroLast = horoCurrent - 1;
88      pthread_mutex_unlock(&getNextTaskMutex);
89      return true;
90  }
91
92  /* thread computing horocyclic geodesics in a certain range
        */
93  struct ThreadData {unsigned int numGeods[MAX_WORD_LENGTH +
        1];};
94  void* thread(void* p)
95  {
96      ThreadData* td = (ThreadData*)p;
97      memset(td->numGeods, 0, (MAX_WORD_LENGTH + 1) * sizeof(
            unsigned int));
98
99      long horoFirst, horoLast;
100     while(getNextTask(&horoFirst, &horoLast))
101     {
102         for(long n = horoFirst; n <= horoLast; n++)
103         {
```

```
104             unsigned int geodLength =
                    compute_horo_geod_length(n);
105             if(geodLength <= MAX_WORD_LENGTH)
106                 td->numGeods[geodLength]++;
107             else
108                 /* Increasing n by 1 can shorten the geodesic
                        length by at most 1, so if it was too
                       long this time, we may skip some values of
                        n. */
109                 n += geodLength - MAX_WORD_LENGTH - 1;
110         }
111     }
112     return 0;
113 }
114
115 int main()
116 {
117     pthread_mutex_init(&getNextTaskMutex, NULL);
118     pthread_t   threadids[NUM_THREADS];
119     ThreadData  td[NUM_THREADS];
120     for(unsigned int t = 0; t < NUM_THREADS; t++)
121     {
122         int rc = pthread_create(&threadids[t], NULL, &thread,
                (void*)&td[t]);
123         assert(rc >= 0);
124     }
125     /* wait for termination of all threads */
126     for(unsigned int t = 0; t < NUM_THREADS; t++)
127         pthread_join(threadids[t], NULL);
128     pthread_mutex_destroy(&getNextTaskMutex);
129
130     /* collect results */
131     unsigned int numGeods[MAX_WORD_LENGTH + 1];
132     memset(numGeods, 0, (MAX_WORD_LENGTH + 1) * sizeof(
           unsigned int));
133     numGeods[0] = 1; /* there is one geodesic of length 0 */
134     for(unsigned int t = 0; t < NUM_THREADS; t++)
135         for(unsigned int i = 0; i <= MAX_WORD_LENGTH; i++)
136             /* count both the positive and negative elements
                */
137             numGeods[i] += 2 * td[t].numGeods[i];
138
139     /* output */
140     for(unsigned int g = 0; g < MAX_WORD_LENGTH; g++)
141         std::cout << "B[" << g << "] = " << numGeods[g] <<
               std::endl;
142     return 0;
143 }
```

$B_0 = 1$	$B_{35} = 186$	$B_{70} = 44096$	$B_{105} = 10624906$
$B_1 = 2$	$B_{36} = 208$	$B_{71} = 51614$	$B_{106} = 12436968$
$B_2 = 2$	$B_{37} = 254$	$B_{72} = 60562$	$B_{107} = 14540864$
$B_3 = 2$	$B_{38} = 294$	$B_{73} = 70326$	$B_{108} = 17006524$
$B_4 = 2$	$B_{39} = 338$	$B_{74} = 82812$	$B_{109} = 19895006$
$B_5 = 2$	$B_{40} = 410$	$B_{75} = 96708$	$B_{110} = 23267412$
$B_6 = 2$	$B_{41} = 452$	$B_{76} = 112590$	$B_{111} = 27216446$
$B_7 = 2$	$B_{42} = 564$	$B_{77} = 132660$	$B_{112} = 31831420$
$B_8 = 4$	$B_{43} = 642$	$B_{78} = 154366$	$B_{113} = 37227130$
$B_9 = 2$	$B_{44} = 730$	$B_{79} = 180660$	$B_{114} = 43550214$
$B_{10} = 4$	$B_{45} = 912$	$B_{80} = 211876$	$B_{115} = 50933310$
$B_{11} = 4$	$B_{46} = 1016$	$B_{81} = 247002$	$B_{116} = 59574606$
$B_{12} = 6$	$B_{47} = 1174$	$B_{82} = 289592$	$B_{117} = 69671956$
$B_{13} = 4$	$B_{48} = 1448$	$B_{83} = 338376$	$B_{118} = 81504184$
$B_{14} = 8$	$B_{49} = 1618$	$B_{84} = 395482$	$B_{119} = 95325246$
$B_{15} = 10$	$B_{50} = 1926$	$B_{85} = 463626$	$B_{120} = 111481052$
$B_{16} = 6$	$B_{51} = 2266$	$B_{86} = 540940$	$B_{121} = 130415594$
$B_{17} = 12$	$B_{52} = 2594$	$B_{87} = 633632$	$B_{122} = 152527176$
$B_{18} = 16$	$B_{53} = 3102$	$B_{88} = 740944$	$B_{123} = 178383116$
$B_{19} = 10$	$B_{54} = 3588$	$B_{89} = 865508$	$B_{124} = 208677414$
$B_{20} = 20$	$B_{55} = 4222$	$B_{90} = 1014572$	$B_{125} = 244045370$
$B_{21} = 20$	$B_{56} = 4908$	$B_{91} = 1185106$	$B_{126} = 285451874$
$B_{22} = 22$	$B_{57} = 5756$	$B_{92} = 1385484$	$B_{127} = 333884804$
$B_{23} = 30$	$B_{58} = 6756$	$B_{93} = 1623088$	$B_{128} = 390492208$
$B_{24} = 34$	$B_{59} = 7842$	$B_{94} = 1895228$	$B_{129} = 456783838$
$B_{25} = 34$	$B_{60} = 9236$	$B_{95} = 2218648$	$B_{130} = 534205772$
$B_{26} = 46$	$B_{61} = 10822$	$B_{96} = 2596182$	$B_{131} = 624856804$
$B_{27} = 56$	$B_{62} = 12560$	$B_{97} = 3032416$	$B_{132} = 730862612$
$B_{28} = 60$	$B_{63} = 14640$	$B_{98} = 3551262$	$B_{133} = 854811826$
$B_{29} = 68$	$B_{64} = 17420$	$B_{99} = 4151324$	$B_{134} = 999825280$
$B_{30} = 88$	$B_{65} = 20066$	$B_{100} = 4854608$	$B_{135} = 1169416424$
$B_{31} = 94$	$B_{66} = 23570$	$B_{101} = 5682136$	$B_{136} = 1367811466$
$B_{32} = 124$	$B_{67} = 27654$	$B_{102} = 6641712$	$B_{137} = 1599816932$
$B_{33} = 128$	$B_{68} = 32166$	$B_{103} = 7769100$	$B_{138} = 1871178314$
$B_{34} = 156$	$B_{69} = 37816$	$B_{104} = 9091012$	$B_{139} = 2188581536$

Figure 5.1: The first 140 coefficients of the horocyclic growth series of $\mathrm{BS}(2,3)$

Implementations of Power Circuits

Under the author's guidance, Armin Weiß has implemented power circuits in C++ for $q = 2$. The code is available in the CRAG library [MU]. There are two implementations. The first one uses the simpler reduction method from Section 3.3 and stores graphs with linked incidence lists. The second one implements the more complicated tree version with compact markings (see Section 3.4.3). The paths of the trees are stored in the columns of matrices. Both implementations have the same interface and can be used interchangeably. Algorithm 5.2 demonstrates the usage of the library and Figure 5.2 shows the output, visualized with the open source software Graphviz. The letter R indicates that a node belongs to the reduced part of the circuit.

Algorithm 5.2: Demonstration of the power circuit implementation

```
1  PowerCircuitGraph   pc;
2  /* create (automatically) a marking m1 with value 9 */
3  Marking m1 = pc.createMarking(9);
4  /* create new nodes n1 and n2, both with m1 as successor
       marking */
5  Node n1 = pc.createNode(m1);
6  Node n2 = pc.createNode(m1);
7  /* create a marking m2 containing n1 and n2 */
8  vector<Node> nodelist;
9  nodelist.push_back(n1);
10 nodelist.push_back(n2);
11 Marking m2 = pc.createMarking(nodelist);
12 /* plot the circuit with m1 and m2 highlighted */
13 pc.draw("graph1", m1, m2);
14 /* reduce and plot again */
15 pc.reduce();
16 pc.draw("graph2", m1, m2);
```

The library also contains code for solving the word problem in the Baumslag-Gersten group. On current desktop computers, instances like $[T_{20}, t] \stackrel{?}{=} 1$ are solved in reasonable time (see Section 4.3 for the definition of T_n). The algorithm solving the word problem in Higman's group has not been implemented, since the $\mathcal{O}(n^6)$ time bound is beyond the capabilities of modern computers for practically all non-trivial instances.

It turns out that in practice the overhead needed for treed power circuits outweighs the benefits of the asymptotically more efficient reduction procedure. This is not surprising, since the logarithmic factor in the time complexity of the simpler version is bounded by a very small constant for all feasible instances. In contrast, creating and maintaining a tree structure multiplies the running time by a much larger factor.

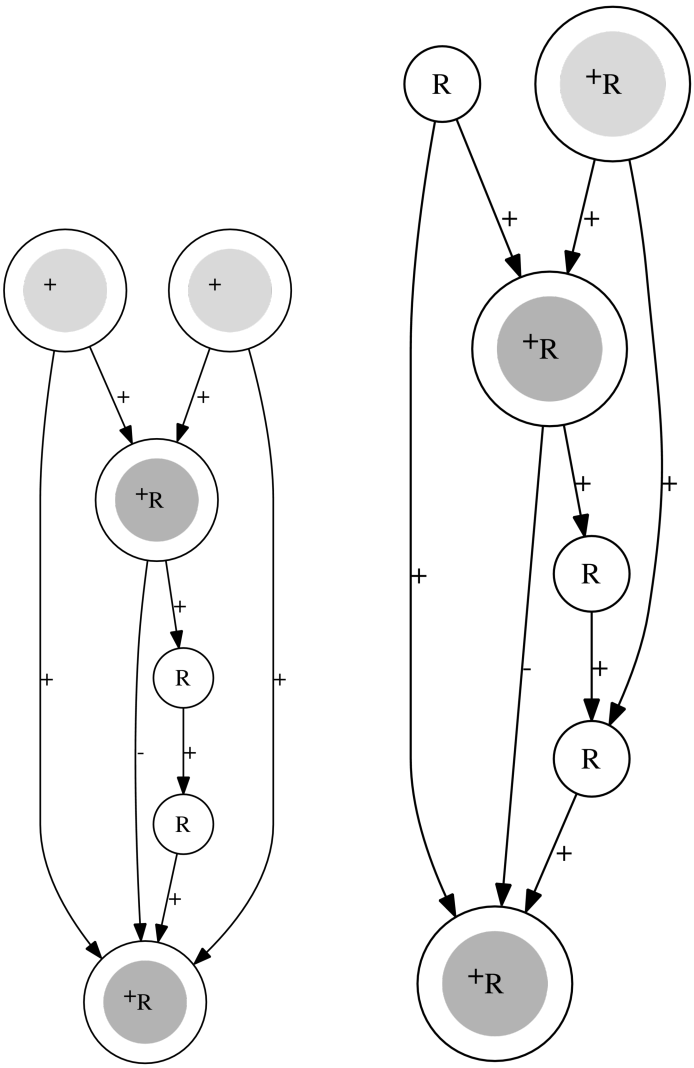


Figure 5.2: Demonstration of the power circuit implementation

Acknowledgements

First and foremost, I would like to thank my advisor Volker Diekert for his constant encouragement and support all along the research for and production of this thesis. I also thank the current and former members of my group at the *Institut für Formale Methoden der Informatik* in Stuttgart for all the helpful discussions and distractions during my work there. In particular, I would like to mention Markus Lohrey and Benjamin Hoffmann whose valuable comments and suggestions greatly helped me improve this thesis. Special thanks go to Armin Weiß who, as an undergraduate, implemented a great deal of the power circuit algorithms in the Appendix.

I am indebted to Stefan Funke and Jochen Eisner for providing the machinery on which the experimental results of section 2.4.1 were computed.

Parts of this work originate from visits to the *Southern Utah University* in Cedar City, UT and the *Stevens Institute of Technology* in Hoboken, NJ. I am much obliged to Eric Freden of SUU as well as to Sasha Ushakov at Stevens for their hospitality and support.

Finally, my thanks go to Peter Krauter who piqued my interest in mathematics in the first place (and who most likely prevented me from becoming a computer programmer instead of a computer scientist) and to Jörg Brüdern from whom I have learned a lot, not only about mathematics itself but also the art of teaching it.

Bibliography

[All12] Daniel Allcock. Triangles of Baumslag-Solitar Groups. *Canadian Journal of Mathematics*, 64(2):241–253, 2012.

[BO93] Ronald V. Book and Friedrich Otto. *String-rewriting systems*. Springer, 1993.

[Boo58] William W. Boone. The word problem. *Proceedings of the National Academy of Sciences*, 44(10):1061–1065, 1958.

[Bra74] Marcus Brazil. Growth Functions for Some Nonautomatic Baumslag-Solitar Groups. *Transactions of the American Mathematical Society*, 342:137–154, 1974.

[BS62] Gilbert Baumslag and Donald Solitar. Some two-generator one-relator non-Hopfian groups. *Bulletin of the American Mathematical Society*, 68(3):199–201, 1962.

[CEG94] D. J. Collins, M. Edjvet, and C. P. Gill. Growth series for the group $\langle x, y | x^{-1}yx = y^l \rangle$. *Archiv der Mathematik*, 62(1):1–11, 1994.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[CS63] Noam Chomsky and Marcel-Paul Schützenberger. The Algebraic Theory of Context-Free Languages. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, pages 118–161. North Holland, 1963.

[DDM10] Volker Diekert, Andrew J. Duncan, and Alexei G. Miasnikov. Geodesic rewriting systems and pregroups. In Oleg Bogopolski, Inna Bumagin, Olga Kharlampovich, and Enric Ventura, editors, *Combinatorial and Geometric Group Theory*, Trends in Mathematics, pages 55–91. Birkhäuser, 2010.

[Deh12] Max Dehn. Über unendliche diskontinuierliche Gruppen. *Mathematische Annalen*, 71:116–144, 1912.

[DL11] Volker Diekert and Jürn Laun. On Computing Geodesics in Baumslag-Solitar Groups. *International Journal of Algebra and Computation*, 21(1-2):119–145, 2011.

[DLU12] Volker Diekert, Jürn Laun, and Alexander Ushakov. Efficient algorithms for highly compressed data: The Word Problem in Higman's group is in P. In *29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)*, volume 14 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 218–229, 2012.

[DLU13] Volker Diekert, Jürn Laun, and Alexander Ushakov. Efficient algorithms for highly compressed data: The word problem in Higman's group is in P. *International Journal of Algebra and Computation*, 2013. To appear.

[EEO12] Murray Elder, Gillian Elston, and Gretchen Ostheimer. On groups that have normal forms computable in logspace. *Preprint*, 2012. `http://arxiv.org/abs/1201.4363`.

[EJ92] M. Edjvet and D. L. Johnson. The growth of certain amalgamated free products and HNN-extensions. *Journal of the Australian Mathematical Society*, 52:285–298, 1992.

[Eld10] Murray Elder. A linear time algorithm to compute geodesics in solvable Baumslag-Solitar groups. *Illinois Journal of Mathematics*, 54(1):109–128, 2010.

[FKS11] Eric M. Freden, Teresa Knudson, and Jennifer Schofield. Growth in Baumslag-Solitar groups I: subgroups and rationality. *London Mathematical Society Journal of Computation and Mathematics*, 14:34–71, 2011.

[Ger91] Stephen M. Gersten. Dehn functions and ℓ_1-norms of finite presentations. In Gilbert Baumslag and Charles F. Miller III, editors, *Algorithms and Classification in Combinatorial Group Theory*, pages 195–220. Springer, 1991.

[Ger93] Stephen M. Gersten. Isoperimetric and Isodiametric Functions of Finite Presentations. In *Geometric group theory*, volume 1 of

London Mathematical Society Lecture Note Series 181, pages 79–96. Cambridge University Press, 1993.

[Gri83] Rostislav I. Grigorchuk. On Milnor's problem of group growth. *Doklady Akademii Nauk SSSR*, pages 30–33, 1983.

[Har78] Michael A. Harrison. *Introduction to Formal Language Theory*. Addison Wesley, 1978.

[Hig51] Graham Higman. A finitely generated infinite simple group. *Journal of the London Mathematical Society*, 26:61–64, 1951.

[HNN49] Graham Higman, Bernhard H. Neumann, and Hanna Neumann. Embedding Theorems for Groups. *Journal of the London Mathematical Society*, 24:247–254, 1949.

[Jan88] Matthias Jantzen. *Confluent String Rewriting*, volume 14 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1988.

[Kui70] Werner Kuich. On the entropy of Context-Free Languages. *Information and Control*, 16:173–200, 1970.

[Lau12] Jürn Laun. Efficient algorithms for highly compressed data: The Word Problem in Generalized Higman Groups is in P. *Preprint*, 2012.
`http://arxiv.org/abs/1006.2570`.

[Loh04] Markus Lohrey. Word problems on compressed words. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, pages 906–918. Springer, 2004.

[LS01] Roger Lyndon and Paul Schupp. *Combinatorial Group Theory*. Classics in Mathematics. Springer, 2001.

[Mes72] Stephen Meskin. Nonresidually finite one-relator groups. *Transactions of the American Mathematical Society*, 164:105–114, 1972.

[Mil68] John W. Milnor. A note on curvature and fundamental group. *Journal of Differential Geometry*, 2(1):1–7, 1968.

[MU] Alexei G. Myasnikov and Alexander Ushakov. Cryptography And Groups (CRAG). Software Library.
`http://www.stevens.edu/algebraic/downloads.php`.

[MUW11] Alexei G. Myasnikov, Alexander Ushakov, and Dong Wook Won. The Word Problem in the Baumslag group with a non-elementary Dehn function is polynomial time decidable. *Journal of Algebra*, 345(1):324–342, 2011.

[MUW12] Alexei G. Myasnikov, Alexander Ushakov, and Dong Wook Won. Power circuits, exponential algebra, and time complexity. *International Journal of Algebra and Computation*, 22(6), 2012.

[Neu54] Bernhard H. Neumann. An Essay on Free Products of Groups with Amalgamations. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 246(919):503–554, 1954.

[Nov55] Pyotr S. Novikov. On the algorithmic unsolvability of the word problem in group theory. *Proceedings of the Steklov Institute of Mathematics*, 44:1–142, 1955.

[Pla04] A. N. Platonov. An isoperimetric function of the Baumslag-Gersten group. *Vestnik Moskovskogo Universiteta. Seriya I. Matematika, Mekhanika*, 3:12–17, 2004. English translation in Moscow University Mathematics Bulletin 59, 2004.

[Sch08] Saul Schleimer. Polynomial-time word problems. *Commentarii Mathematici Helvetici*, 83:741–765, 2008.

[Ser02] Jean-Pierre Serre. *Trees*. Springer, 2nd edition, 2002.

Index