THE DEHN FUNCTION, WORD PROBLEM, AND BOUNDED WORD PROBLEM FOR FINITELY GENERATED DECIDABLE GROUP PRESENTATIONS

BY

DESMOND FARRELL CUMMINS

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mathematics
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Professor Ilya Kapovich, Chair
Professor Sergei Ivanov, Director of Research
Professor Christopher Leininger
Professor Paul Schupp

Abstract

For finite group presentations, the word problem is solvable if and only if the Dehn function is computable. Additionally, the bounded word problem is always solvable. For finitely generated decidable group presentations, this is not always the case. The main result of the present work is to determine whether there exist examples of finitely generated decidable group presentations for each combination of solvability/unsolvability of the word problem, solvability/unsolvability of the bounded word problem, and computability/uncomputability of the Dehn function.

To My Parents, John and Frances Cummins, for their many years of loving support.

Acknowledgments

I would like to first thank my advisor, Professor Sergei Ivanov, for the past several years of excellent instruction and encouragement. I am grateful to have had the chance to work with such an accomplished mathematician. I would also like to thank Professor Paul Schupp for introducing me to group theory and for our many helpful conversations. Additionally, I owe a debt of gratitude to Professor Ilya Kapovich for his advice and support, and to Professor Christopher Leininger for both his friendship and his capacity as a committee member. I would especially like to thank Professors Julia Knight, Francis Connolly, and Peter Cholak for encouraging my interest in mathematics as an undergraduate. I also thank all of the faculty, staff, and graduate students in the Department of Mathematics at the University of Illinois for creating a stimulating and cultivating environment in which to learn.

Finally, I would like to thank my family, in particular, my parents, John and Frances, for their constant love and support.

The author acknowledges support of the National Science Foundation through a fellowship funded by the grant "EMSW21-MCTP: Research Experience for Graduate Students" (NSF DMS 08-38434).

Table of Contents

Chapter 1

Introduction

If A is a finite set of symbols, we use A^* to denote the set of finite sequences of symbols of A (including the empty sequence, denoted ε). If $L \subseteq A^*$, then we call L a *language* over A. For $L \subseteq A$, we say that L is *decidable* if there is a Turing machine (defined formally in section 2) that accepts an input $\mathbf{u} \in A^*$ if $\mathbf{u} \in L$ and halts in a non-accept state $\mathbf{u}$ if $\mathbf{u} \notin L$. We say that L is *enumerable* if there is a Turing machine that accepts an input $\mathbf{u} \in A$ if and only if $\mathbf{u} \in L$. If a process or a construction can be performed algorithmically we say that the process or construction is *effective*.

Let a countably generated group G be defined by a presentation in terms of generators and defining relators

$$P = \langle\ X \parallel R\ \rangle\ , \tag{1.0.1}$$

where $X = \{a_1, a_2, \dots\}$ is a countable alphabet and R is a set of defining relators which are nonempty cyclically reduced words over the alphabet $X^{\pm 1} = X \cup X^{-1}$ (we assume that R is closed under taking inverses and cyclic conjugates). Let $F(X)$ denote the free group over X, $|W|$ denote the length of a word $W \in F(X)$ over the alphabet $X^{\pm 1}$, and $\langle\langle R \rangle\rangle$ denote the normal closure of R in $F(X)$. Then the notation (1.0.1) means that G is the quotient group $F(X)/\langle\langle R \rangle\rangle$. Recall that a presentation (1.0.1) is called *finite* if X and R are finite.

For a group presentation $P = \langle X \| R \rangle$, if the generating set X is finite then R is a language over the finite alphabet $X^{\pm 1}$. If R is either decidable or enumerable then we say that P is decidable or enumerable, respectively.

We will also work with presentations P for which the set X is countably infinite. If R is a language over a countably infinite alphabet, then the definitions of decidable and enumerable do not technically apply to

R. We avoid this difficulty by requiring that the elements of X to be of the form $g_{\mathbf{y}}$, where the letter g is an element of a finite alphabet and the index $\mathbf{y}$ is a word over another finite alphabet. From this perspective, both X and R can be regarded as a languages over a finite alphabet (since R is a set of words over $X^{\pm 1}$). Thus, the definitions of decidable and enumerable can be applied to X and R. We say that P is *decidable* if X and R are decidable. Also, P is *enumerable* if X and R are enumerable.

A function $f : \mathbf{N} \to \mathbf{N}$ is called an *isoperimetric function* of a presentation $P = \langle X \| R \rangle$ of a group G if for every number n and every word w trivial in P with $|w| < n$, there exists a van Kampen diagram over P with boundary label w and area $< f(n)$. In other words, w is a product of at most $f(n)$ conjugates of the relators from R (see [8], [6], [3], [1]). The smallest isoperimetric function of a decidable presentation P is called the *Dehn function* of P. We will commonly denote Dehn functions by f. The concept of a Dehn function was introduced by Gromov in [5].

A presentation $P = \langle X \| R \rangle$ is *minimal* if if fulfils the following condition. For a relator set R' over X with $R' \subseteq R$, if R and R' have the same normal closure, then $R = R'$.

Let $f, g : \mathbf{N} \to \mathbf{N}$ be two functions. We write $f \preceq g$ if there exists a nonnegative constant d such that $f(n) < dg(dn) + dn$. All functions $g(n)$ considered in this paper grow at least as fast as n. For our purposes then, $f(n) \preceq g(n)$ if and only if $f(n) < dg(dn)$ for some positive constant d. Two functions f, g are called equivalent, denoted $f \approx g$, if $f \preceq g$ and $g \preceq f$. For a set of pairs of functions $\{(f_i, g_i) | i \in \mathbf{N}\}$, if $f_i \approx g_i$ then we say that these equivalences are *uniform* if there is a single constant d such that $f_i(n) < dg_i(dn)$ and $f_i(n) > dg_i(dn)$ for all $i \in \mathbf{N}$.

A function $f : \mathbf{N} \to \mathbf{N}$ is *superadditive* if for all natural numbers m, n we have $f(m + n) \geq f(m) + f(n)$.

For a group presentation $P = \langle X \| R \rangle$ where X is finite or countably infinite, we say that the word problem is solvable for P if the language of words over alphabet $X^{\pm 1}$ that are trivial in P is decidable. Consider the language of pairs (w, n) where w is a word over X, n is a positive integer written in unary, and there exists a P diagram with boundary label w and area no greater than n. If this language is decidable then we say that the *bounded word problem* for P is solvable. We say that the Dehn function f of P is computable if the language of pairs $(n, f(n))$, where n and $f(n)$ are written in binary, is decidable.

The bounded word problem is solvable for all finite presentations. Additionally, it is well known that for a finite presentation, the solvability of the word problem and the computability of the Dehn function are equivalent conditions. It is not clear whether any such equivalence holds for decidable presentations. In [4], Grigorchuk and Ivanov pose the following problem:

Problem 1. Let the relator set R of a finitely generated presentation (1.0.1) be decidable. Prove or disprove that

(a) If the word problem for (1.0.1) is solvable, then the Dehn j-function $f(x)$ of (1.0.1) is computable.

(b) If the Dehn function $f(x)$ of (1.0.1) is computable, then the word problem for (1.0.1) is solvable.

A counterexample to Problem 1(b) is given in [4, Example 2.4]. It is also pointed out in [4] that it would be of interest to consider a stronger and presumably more difficult version of Question 1 in which the relator set R is assumed to be minimal.

This question inspires the following observation. For a finitely generated decidable minimal presentation, there are eight possible cases regarding the solvability of the word problem, the solvability of the bounded word problem, and the computability of the Dehn function. These cases are displayed in the below table, in which "y" and "n" stand for yes and no. For example, case **2** refers to decidable minimal presentations that have solvable word problem and bounded word problem, but uncomputable Dehn function.

Case Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Solvable Word Problem	y	y	y	y	n	n	n	n
Solvable Bounded Word Problem	y	y	n	n	y	y	n	n
Computable Dehn Function	y	n	y	n	y	n	y	n

It is obvious that the presentation $\langle a \| a \rangle$ is an example of case **1**. In the following lemma we prove that examples of cases **2** and **5** do not exist.

Lemma 1.0.1. *There are no finitely generated decidable presentations that satisfy case **2** or case **5**.*

Proof. For case **2**, we observe that if a finitely generated decidable presentation P has solvable word problem and solvable bounded word problem, then we can compute the Dehn function as follows. For $n > 0$, use the solvability of the word problem to effectively find all of the finitely many trivial words in P of length $\leq n$.

Then use the solvability of the bounded word problem to find the area of the smallest P diagram for each of these trivial words. The area of the largest such diagram will be the value of the Dehn function of P on input n. Therefore no finitely generated decidable presentation satisfies case **2**.

For case **5** we observe that if a finitely generated decidable presentation P has solvable bounded word problem and computable Dehn function f, then the word problem for P can be solved as follows. To determine whether a word w is trivial in P, first compute $f(|w|)$. Then run the bounded word problem algorithm on the input $(w, f(|w|))$. If this input is accepted, then w is trivial in P. Otherwise, by the definition of the Dehn function, w is not trivial in P.

□

The purpose of this paper is to prove the following theorem, which provides an answer to the stronger "minimal" version of Question 1.

Theorem 1.0.2. *There exist examples of finitely generated decidable minimal presentations that satisfy cases **3**, **4**, **6**,**7**, and **8**.*

To provide these examples we will rely on previous results that involve constructing group presentations that simulate Turing machines. Several results on this topic exist, such as the classical Novikov-Boone-Higman-Aanderaa embedding of a finitely generated group G into a finitely presented group H [9]. In [8], Madlener and Otto explored the idea of constructing such an embedding so that the Dehn function of the presentation for H was not "much bigger" than the time complexity of the Turing machine it simulates. In [10] and [2], Birget, Ol'shanski, Rips, and Sapir construct such an embedding in which the Dehn function of the presentation H is equivalent to the fourth power of the time function of the Turing machine (provided the fourth power of time function is superadditive). Our primary tool will be a remarkable result proven in [10] by Sapir, Birget, and Rips:

Theorem 1.0.3. *Let $L \subseteq A^*$ be a language accepted by a k-tape Turing machine M with time function $T(n)$ for which $T(n)^4$ is superadditive. Then there exists a finite group presentation $P(M)$ with generating set X and with Dehn function equivalent to $T(n)^4$. Also, there exists an injective map $\mathcal{K} : A^* \to (X \cup X^{-1})^*$ such that*

1. *$\boldsymbol{u} \in L$ if and only if $\mathcal{K}(\boldsymbol{u}) = 1$ in $P(M)$;*
2. *$\mathcal{K}(\boldsymbol{u})$ has length $O(|\boldsymbol{u}|)$. There is a linear-time algorithm that takes as input a word $\boldsymbol{u}$ in A^* and outputs $\mathcal{K}(\boldsymbol{u})$.*

We will generalize this result by constructing group presentations to simulate computing devices that are more powerful than Turing machines. In section 2, we define a partial ordering $\sqsubseteq$ for Turing machines. If M_1 and M_2 are Turing machines and $M_1 \sqsubseteq M_2$ then we say M_1 is a submachine of M_2. For an increasing chain $M_1 \sqsubseteq M_2 \sqsubseteq \dots$ of Turing machines we will define a machine M_∞ which can be thought of as the limit of the chain $M_1 \sqsubseteq M_2 \sqsubseteq \dots$, i.e., the minimal object for which $M_i \sqsubseteq M_\infty$ for all $i \in \mathbf{N}$. We will call M_∞ a *union machine*. A union machine is not necessarily a Turing machine because it may violate the finiteness conditions that Turing machines must satisfy. However M_∞ does behave like a Turing machine; in fact, it is possible to use techniques similar to those of [10] to construct a group presentation $P(M_\infty)$ from M_∞. We use the fact that the group presentation $P(M_i)$ simulates the Turing machine M_i to prove that an analogous result holds in the limiting case, i.e., that $P(M_\infty)$ simulates M_∞ in much the same way that $P(M_i)$ simulates M_i. Specifically, we will prove the following theorem.

Theorem 1.0.4. *Let $L \subset A^*$ be a language accepted by an enumerable k-tape union machine M_∞. Suppose $T(n)$ is the time function of M_∞, where $T(n)^4$ is superadditive. There exists a finitely generated decidable minimal presentation $P'(M_\infty)$ and an injective map $h \circ \mathcal{K}$ from A^* to the generating set of $P'(M_\infty)$ such that:*

1. *For an input word $\boldsymbol{u}$ of M_∞, the word $h(\mathcal{K}(\boldsymbol{u}))$ is trivial in $P'(M_\infty)$ if and only if $\boldsymbol{u}$ is an acceptable input of M_∞.*
2. *L is decidable if and only if the word problem for $P'(M_\infty)$ is solvable.*
3. *$P'(M_\infty)$ has Denh function equivalent to $T(n)^4$.*

In the proof of Theorem 1.0.3, the authors begin with an arbitrary Turing machine M. From M they construct a symmetrized Turing machine M', from which they construct an S-machine $S(M')$, from which they construct a finite group presentation $P(M)$ (the formal definitions of symmetrized Turing machines and S-machines will be given later in the paper). The authors show that each of these constructions inherit certain properties from the original Turing machine M.

For an arbitrary union machine M_∞, we will follow a similar strategy. We will construct a symmetrized union machine M'_∞, a "union S-machine" $S(M'_\infty)$ and a countably generated group presentation $P(M_\infty)$. As in [10], we will then prove that each of these constructions inherits certain properties from M_∞. We will

complete the proof of Theorem 1.0.4 by using $P(M_\infty)$ to construct the finitely generated group presentation $P'(M_\infty)$ described in Theorem 1.0.4.

We will then use Theorem 1.0.4 to construct finitely generated decidable minimal group presentations that satisfy cases **3**,**4**,**7**, and **8**. As for case **6**, we will show in the final section that Theorem 1.0.3 from [10] can be used to construct an example for this case.

The proofs of this paper will heavily reference the proofs in [10]. We will adhere as closely as possible to the terms and notation used in [10]. Any significant deviation from the terms and notation of [10] will be explicitly pointed out. Also, when citing [10], we will provide page numbers for the reader's convenience.

Chapter 2

Turing machines and Union Machines

We will begin with formal definitions of Turing machines and union machines. In this paper all Turing machines are assumed to be nondeterministic. A k-tape Turing machine has k tapes and k heads. One can view it as a six-tuple

$$M = \langle A, \Gamma, Q, \Theta, \vec{s}, \vec{h} \rangle,$$

where A is the input alphabet and Γ is the tape alphabet ($A \subseteq \Gamma$). Each head has its own finite set of (disjoint) states, Q_i. The set of states of the machine M is $Q = Q_1 \times ... \times Q_k$. An element of Q is denoted $\vec{q} = (q_1, ..., q_k)$ where $q_i \in Q_i$. There is a state $\vec{s} \in Q$ called the start state, and a state $\vec{h} \in Q$ is called the accept state.

A *configuration* $\mathbf{c}_i$ of the ith tape of M is a word $\mathbf{c}_i = \alpha_i \mathbf{u} q_i \mathbf{v} \omega_i$ where $q_i \in Q_i$ is the current state of the ith head, $\mathbf{u} \in \Gamma^*$ is the word to the left of the head, and $\mathbf{v} \in \Gamma^*$ is the word to the right of the head. The letters α_i and ω_i are special letters of Γ called the ith left and right end marker, respectively. These letters may only appear at the left and right end of the ith tape; never anywhere else. If the ith tape of a Turing machine is in configuration $\mathbf{c}_i$, we say that $\mathbf{uv}$ is the word written in the ith tape. If $\mathbf{u}, \mathbf{v} = \varepsilon$ in $\mathbf{c}_i$, we say that the ith tape is empty.

A *configuration* $\mathbf{c}$ of M is a k-tuple

$$\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_k),$$

where $\mathbf{c}_i$ is a configuration of the ith tape. The length $|\mathbf{c}|$ of the configuration is the sum of the lengths of the words $\mathbf{c}_i$. The state of the configuration $\mathbf{c}$ is the tuple $\vec{q} = (q_1, ..., q_k)$, where q_i is the Q_i letter appearing in $\mathbf{c}_i$.

If a tape letter x is adjacent to a head letter q_i in a configuration $\mathbf{c}$, then we say that the head letter q_i

observes x in $\mathbf{c}$.

An *input configuration* is a configuration in which the word written in the first tape is in A^*, all other tapes are empty, the head of each tape observes its right marker ω_i, and the state of the configuration is $\vec{s}$.

The first tape of a Turing machine is called the *input tape*. The input tape is a read only tape; that is, the word written on the input tape does not change during the course of a computation. An *accept configuration* is any configuration for which the state is $\vec{h}$, the non-input tapes are empty, and the head of the first tape observes the right end letter ω_1.

The commands of M provide a way to pass from one configuration to another. A command $\tau \in \Theta$ may be applied to some configurations $\mathbf{c}$ of M, depending on the state of $\mathbf{c}$ and the letters observed by the heads in $\mathbf{c}$. In a one-tape machine every command τ is of the following form:

$$uqv \rightarrow u'q'v',$$

where $u, v, u'v'$ are either letters of Γ or the empty word ε. The command τ can only be executed from a 1-tape configuration $\mathbf{c}$ if uqv is a subword of the single tape of $\mathbf{c}$. If the command τ is executed, then the machine replaces the subword uqv with the subword $u'q'v'$. In any such command τ, $u = \alpha_1$ if and only if $u' = \alpha_1$. Also $v = \omega_1$ if and only if $v' = \omega_1$.

Formally, a single-tape command is a six-tuple of symbols from the finite alphabets Q and Γ, and $\{\varepsilon\}$. For example, the above command τ is the tuple (u, q, v, u', q', v').

For a general k-tape machine a command is a k-tuple of single-tape commands

$$\tau = (\tau_1, ..., \ \tau_k),$$

where τ_i is $u_i q_i v_i \rightarrow u_i' q_i' v_i'$. Formally then, such a command is a $6k$-tuple of symbols. In order to execute the command τ from a k-tape configuration $\mathbf{c}$, $u_i q_i v_i$ must be a subword of the configuration of the ith tape of $\mathbf{c}$. If this is the case, then the machine may execute the command τ, replacing each $u_i q_i v_i$ by $u_i' q_i' v_i'$. In any such command τ, $u_i = \alpha_i$ if and only if $u_i' = \alpha_i$. Also $v_i = \omega_i$ if and only if $v_i' = \omega_i$.

In later sections, we will want to represent each command of a Turing machine as a symbol. The formal symbol for a command will be $\tau_{\vec{x}}$, where $\vec{x}$ is the corresponding $6k$-tuple. The purpose of this notation is for a command symbol to contain a complete description of its corresponding command. When referring to a command symbol informally, we will often omit the $\vec{x}$ index.

A *computation* of a k-tape Turing machine M is a sequence of configurations $\mathbf{c}^1, ..., \mathbf{c}^n$ such that for every $i = 1, ..., n-1$, the machine passes from $\mathbf{c}^i$ to $\mathbf{c}^{i+1}$ by applying one of the commands from Θ. A configuration $\mathbf{c}$ is *acceptable* to M if there exists at least one computation that starts with $\mathbf{c}$ and ends with an accept configuration. Such a computation is called an *accepting computation* for $\mathbf{c}$.

An input word $\mathbf{u} \in X^*$ is said to be *acceptable* if the input configuration for $\mathbf{u}$ is an acceptable configuration. The set of all acceptable input words over the alphabet X is called the *language accepted by* M.

Let $C = (\mathbf{c}^1, ..., \mathbf{c}^n)$ be a computation of a machine M such that the configuraion $\mathbf{c}^{j+1}$ is obtained from $\mathbf{c}^j$ by the command $\tau_j \in \Theta$. Then we call the word $\tau_1 ... \tau_{n-1}$ the *history* of the computation. The number $(n-1)$ will be called the *time* or *length* of the computation. The sum $\sigma_{j=1}^{n} |\mathbf{c}^j|$ will be called the *area* of C and will be denoted by area(C).

With every Turing machine we associate five functions: the *time function* $T(n)$, the *space function* $S(n)$, the *generalized time function* $T'(n)$, the *generalized space function* $S'(n)$, and the *area function* $A(n)$. These functions will be called the *complexity functions* of the machine. They are defined below.

We define $T(n)$ as the minimal number such that every acceptable input configuration $\mathbf{c}$ with $|\mathbf{c}| \leq n$ is accepted by a computation of length at most $T(n)$. The number $S(n)$ is the minimal number such that every acceptable input configuration $\mathbf{c}$ with $|\mathbf{c}| \leq n$ is accepted by a computation which contains only configurations of length $\leq S(n)$. We define $T'(n)$ as the minimal number such that every acceptable configuration $\mathbf{c}$ with $|\mathbf{c}| \leq n$ is accepted by a computation of length at most $T'(n)$. The number $S'(n)$ is the minimal number such that every acceptable configuration $\mathbf{c}$ with $|\mathbf{c}| \leq n$ is accepted by a computation which contains only configurations of length $\leq S'(n)$. It is clear that $T(n) \leq T'(n)$ and $S(n) \leq S'(n)$ and it is easy to give examples where these inequalities are strict. The area function $A(n)$ is defined as the minimal number such that for every acceptable configuration $\mathbf{c}$ with $|\mathbf{c}| \leq n$ there exists at least one accepting computation with area at most $A(n)$.

Definition 1. Suppose M_1 and M_2 are Turing machines where, for $i = 1, 2$,

$$M_i = \langle A, \Gamma_i, Q_i, \Theta_i, \vec{s}, \vec{h} \rangle.$$

Note that M_1 and M_2 have identical $A, \vec{s}$, and $\vec{h}$. We write that $M_1 \sqsubseteq M_2$ if $\Gamma_1 \subseteq \Gamma_2$, $Q_1 \subseteq Q_2$ and $\Theta_1 \subseteq \Theta_2$.

Definition 2. Suppose $\{M_i | i \in \mathbf{N}\}$ is a set of Turing machines having identical A, $\vec{s}$, and $\vec{h}$. Let $M_i = \langle X, \Gamma_i, Q_i, \Theta_i, \hat{s}, \hat{h} \rangle$. We define $\bigcup_{i=1}^{\infty} M_i$ as follows:

$$\bigcup_{i=1}^{\infty} M_i = \langle A, \cup_{i=1}^{\infty} \Gamma_i, \cup_{i=1}^{\infty} Q_i, \cup_{i=1}^{\infty} \Theta_i, \vec{s}, \vec{h} \rangle.$$

We call $\bigcup_{i=1}^{\infty} M_i$ the union of the Turing machines M_i, or a *union machine*. Note that every Turing machine is a union machine, but not every union machine is a Turing machine, since the sets $\bigcup_{i=1}^{\infty} \Gamma_i, \bigcup_{i=1}^{\infty} Q_i$, and $\bigcup_{i=1}^{\infty} \Theta_i$ need not be finite. All the terms and notation defined in this section concerning Turing machines have identical interpretations for union machines.

We will denote a union machine $\bigcup_{i=1}^{\infty} M_i$ as M_∞.

We say that a union machine M_∞ is decidable if the sets $\bigcup_{i=1}^{\infty} \Gamma_i$, $\bigcup_{i=1}^{\infty} Q_i$, $\bigcup_{i=1}^{\infty} \Theta_i$ are decidable. A union machine M_∞ is enumerable if the sets $\bigcup_{i=1}^{\infty} \Gamma_i$, $\bigcup_{i=1}^{\infty} Q_i$, $\bigcup_{i=1}^{\infty} \Theta_i$ are enumerable.

For a union machine M_∞, we define the *configuration problem* to be the problem of deciding whether an arbitrary configuration of M_∞ is acceptable.

We will adopt the following convention for notational convenience. If $M_\infty = \bigcup_{i=1}^{\infty} M_i$ is a union machine, then we will assume that $M_i \sqsubseteq M_{i+1}$ for $i \in \mathbf{N}$. This can be assumed without loss of generality, since for each i, $M_1 \cup \ldots M_i$ is a Turing machine, $M_1 \cup \cdots \cup M_i \sqsubseteq M_1 \cup \ldots M_i \cup M_{i+1}$ and $M_\infty = \bigcup_{i=1}^{\infty} M_1 \cup \cdots \cup M_i$.

Chapter 3

Symmetrization of Turing Machines

For a Turing machine command τ of the form

$$(u_1 q_1 v_1 \to u_1' q_1' v_1', ..., u_k q_k v_k \to u_k' q_k' v_k'),$$

we write τ^{-1} to indicate the tuple

$$(u_1' q_1' v_1' \to u_1 q_1 v_1, ..., u_k' q_k' v_k' \to u_k q_k v_k).$$

Note that τ^{-1} has the form of a command of a Turing machine. These two commands are called *mutually inverse*. We say that a Turing machine is *symmetric* if every $\tau \in \Theta$ has an inverse command $\tau^{-1} \in \Theta$. The definition of a symmetric union machine is identical.

We disallow Turing machine/union machine commands τ for which $\tau = \tau^{-1}$. This can be done without loss of generality because when such a command τ is applied to a configuration $\mathbf{c}$, the resulting configuration is $\mathbf{c}$. Since these commands have no effect on the configurations they are applied to, we assume that our Turing machines do not contain such commands. Thus in a symmetric machine the set of commands can be partitioned into *positive* and *negative* commands such that if τ is positive, then τ^{-1} is negative.

The following lemma is nearly identical to Lemma 3.1[10]. The only difference is that the uniformity of the equivalences in property P3 is not explicitly mentioned in the statement of Lemma 3.1[10]. However, it is shown in the proof of Lemma 3.1[10] that the equivalences given in Lemma 3.1[10] are in fact uniform for all k-tape Turing machines. We therefore state the below lemma without proof.

Lemma 3.0.5. *For every k-tape Turing machine M accepting a language L, there exists a $2(k+1)$-tape Turing machine M' called the* symmetrization of M *with the following properties:*

(P1) The language accepted by M' is L.

(P2) M' is symmetric.

(P3) The time, generalized time, space, generalized space functions of M' are equivalent to the time function of M. The area function of M' is equivalent to the square of the time function of M. These equivalences are uniform for all k-tape union machines M.

(P4) The machine M' accepts only when all tapes are empty.

(P5) Each command of M' affects only a single tape of M'. Formally, every command of M' or its inverse has one of the following forms for some $i = 1, ..., 2(k+1)$.

$$(q_1\omega_1 \to q_1'\omega_1, ..., aq_i\omega_i \to q_i'\omega_i, ..., q_{2(k+1)}\omega_{2(k+1)} \to q'_{2(k+1)}\omega_{2(k+1)}). \tag{3.0.1}$$

$$(q_1\omega_1 \to q_1'\omega_1, ..., \alpha_i q_i\omega_i \to \alpha_i q_i'\omega_i, ..., q_{2(k+1)}\omega_{2(k+1)} \to q'_{2(k+1)}\omega_{2(k+1)}). \tag{3.0.2}$$

(P6) The letters used on different tapes are from disjoint alphabets. This includes the state letters.

In order to prove later results in this section we will require a detailed description of how the machine M' is constructed from M. Let M be a k-tape union machine. Recall that by our definition of Turing machines, the first tape of M is the *input* tape, which can only contain letters from the input alphabet. Also, in an *input configuration* of M, an input word is written on the first tape, all other tapes are empty, and the head observes the right end marker of each tape.

We will first describe a version of the construction M' that has $(k+1)$ tapes and satisfies properties P1-P4. We will then adjust this construction to produce a $2(k+1)$-tape version of M' that satisfies P1-P6.

The machine M' has $(k+1)$ tapes. The $(k+1)$st set of state letters Q_{k+1} contains three elements, $q(1), q(2), q(3)$. The machine M' is composed of three subroutines which we will call *phases 1, 2, and 3*. Positive phase one commands can only be applied to configurations whose $(k+1)$st state letter is $q(1)$, positive phase two commands can only be applied to configurations whose $(k+1)$-st state letter is $q(2)$, and positive phase three commands can only be applied to configurations whose $(k+1)$st state letter is $q(3)$.

The only symbols that will ever be written in the $(k+1)$st tape of M' are command symbols of M. The alphabet letters that are used in the first through kth tapes of M' are the letters of Γ, the work alphabet of M. The input alphabet of M' is identical to that of M. We will define M' by first describing the set of positive commands of M'. The description of M' will then be completed by including the inverses of the positive commands.

Input configurations of M' are phase 1 configurations. In a phase 1 configuration the $(k+1)$st head is in state $q(1)$, and the first through kth heads are in the start state of M. For each command letter τ of M, there is a positive phase 1 command of M' that writes the letter τ in the $(k+1)$st tape to the left of the $(k+1)$st head. These commands do not change the state of M'.

At the end of a phase 1 computation, the machine M' will have a sequence of command symbols of M written on the $(k+1)$st tape to the left of the $(k+1)$st head. In order to proceed to the second phase, the machine checks if all tapes except tapes 1 and $(k+1)$ are empty and then changes the $(k+1)$st state to $q(2)$. This is done by a single command of the form:

$$(q_1\omega_1 \to q_1'\omega_1, ..., \alpha_i q_i \omega_i \to \alpha_i q_i'\omega_i, ..., q(1)\omega_{k+1} \to q(2)\omega_{k+1}) \tag{3.0.3}$$

Note that in this command the $(k+1)$st state letter changes from $q(1)$ to $q(2)$. In the second phase M' attempts to use the first k tapes to execute the sequence of commands written on tape $(k+1)$. For every positive command τ of M of the form

$$(u_1q_1v_1 \to u_1'q_1'v_1', ..., u_kq_kv_k \to u_k'q_k'v_k'),$$

we include the following positive command τ' in M':

$$(u_1q_1v_1 \to u_1'q_1'v_1', ..., u_kq_kv_k \to u_k'q_k'v_k', \quad \tau q(2) \to q(2)\tau).$$

The command τ' executes the command τ on the first k tapes of M', checks if the command symbol τ is written to the left of the $(k+1)$st head (if τ is not written there, then the command τ' cannot be executed), and moves the $(k+1)$st head one letter to the left.

Suppose the $(k+1)$st head succeeds in moving all the way to the left end marker of the $(k+1)$st tape during phase 2. Then the machine may pass to phase 3 provided tapes 1 through k form an accept configuration of M (recall that accept configurations of M have all tapes except the input tape empty). In this case M' may pass to phase 3 via the below command (in which $\vec{h} = (h_1, ..., h_k)$ is the accept state of M).

$$(h_1\omega_1 \to h_1\omega_1, \alpha_2h_2\omega_2 \to \alpha_2h_2\omega_2, ..., \alpha_kh_k\omega_k \to \alpha_kh_k\omega_k, \alpha_{k+1}q(2) \to \alpha_{k+1}q(3)).$$

In the third phase the machine erases tapes 1 and $(k+1)$ and enters the accept state of M' once this erasing is complete. This accept state is $(h_1', ..., h_k', q(3))$, where each h_i' is a new state letter that is not a state letter of M. Note that in phase 3 of M' we make an exception to the rule that the input tape is read only. We now include the inverses of all commands described above.

The machine described so far does not yet satisfy properties P5 or P6. In order to get property 5, for each $i = 1, ..., k$, the ith tape of M is divided into two tapes. These new tapes are numbered i and $(i+1/2)$. The new tapes i and $(i+1/2)$ simulate the portions of the old ith tape that lay to the left and right of the head, respectively.

If a configuration of the old tape i was $\alpha_1 \mathbf{u} q_i \mathbf{v} \omega_i$ then the corresponding configurations of the new tapes i and $i + 1/2$ will be, respectively:

$$\alpha_i \mathbf{u} q_i \omega_i \quad \text{and} \quad \alpha_{(i+1/2)} \bar{\mathbf{v}} q_{(i+1/2)} \omega_{(i+1/2)},$$

where $\bar{\mathbf{v}}$ is the word $\mathbf{v}$ rewritten from right to left.

The set of commands is then adjusted so that each old command is replaced by $2k$ new commands that execute the old command one tape at a time. These adjustments (formally described in [10]) do not affect the language accepted by the machine. The complexity functions are changed by only a constant factor (since each old command has been turned into $2(k+1)$ new commands). After these adjustments the machine M' satisfies property P5.

To get property P6, one just needs to replace the alphabet of each tape by a disjoint copy of this alphabet, and the set of state letters on each tape by a disjoint copy of this set. Then one needs to change the commands of the machine accordingly. This completes the description of M'.

We define a *description* of a union machine M_∞ to be a list of the alphabets, commands, and states that comprise M_∞. In the case that M_∞ is a standard Turing machine, these sets are all finite and the description of M_∞ is a word in some finite alphabet.

Lemma 3.0.6. *There is a linear time algorithm that takes as input a Turing machine M and outputs M', the symmetrization of M.*

Proof. This lemma follows immediately from the above description of M'.

□

Suppose $M_\infty = \bigcup_{i=1}^{\infty} M_i$ is a union machine, and M_i' is the symmetrization of the Turing machine M_i. Note that each M_i' has the same input alphabet, the same start state, and the same accept state. We can therefore take the union of these symmetrizations, $M_\infty' = \bigcup_{i=1}^{\infty} M_i'$. We call this union machine the *symmetrization* of M_∞. We can now prove a result for union machines analogous to Lemma 3.0.5.

Lemma 3.0.7. *Suppose M_∞ is a k-tape union machine accepting a language L. The symmetrization M'_∞ of M_∞ has the following properties:*

(Q1) The language accepted by M'_∞ is L.

(Q2) M'_∞ is symmetric.

(Q3) The time, generalized time, space, generalized space functions of M'_∞ are equivalent to the time function of M_∞. The area function of M'_∞ is equivalent to the square of the time function of M_∞. These equivalences are uniform for all k-tape union machines M_∞.

(Q4) The machine M'_∞ accepts only when all tapes are empty.

(Q5) If a union machine M_∞ is enumerable then M'_∞ is enumerable.

(Q6) If the language accepted by M_∞ is decidable, then M'_∞ has solvable configuration problem.

(Q7) Each command of M'_∞ affects only a single tape of M'_∞. Formally, every command of M'_∞ or its inverse has one of the following forms for some $i = 1, ..., 2(k+1)$.

$$(q_1\omega_1 \to q'_1\omega_1, ..., aq_i\omega_i \to q'_i\omega_i, ..., q_{2(k+1)}\omega_{2(k+1)} \to q'_{2(k+1)}\omega_{2(k+1)}). \tag{3.0.4}$$

$$(q_1\omega_1 \to q'_1\omega_1, ..., \alpha_i q_i\omega_i \to \alpha_i q'_i\omega_i, ..., q_{2(k+1)}\omega_{2(k+1)} \to q'_{2(k+1)}\omega_{2(k+1)}). \tag{3.0.5}$$

(Q8) The letters used on different tapes are from disjoint alphabets. This includes the state letters.

Proof. Properties Q1, Q2, Q4, Q7, and Q8 follow immediately from Lemma 3.0.5.

To get property Q3, suppose T_∞ is the time function of M_∞ and T_j is the time function of M_j. Suppose J_∞ is the time, space, generalized time, or generalized space function of M'_∞ and J_j is the time, space, generalized time, or generalized space function of M'_j. By Property P3 of Lemma 3.0.5, $J_j \approx T_j$ and this equivalence is uniform for all $j \in \mathbf{N}$. Therefore there is a constant d such that, for all $j \in \mathbf{N}$, $J_j(x) \leq dT_j(dx)$ and $T_j(x) \leq dJ_j(dx)$.

For an arbitrary $y \in \mathbf{N}$, we will show that $J_\infty(y) \leq dT_\infty(dy)$ and $T_\infty(y) \leq dJ_\infty(dy)$. Since the input alphabet of M_∞ is finite, there are only finitely many input configurations of length $\leq dy$. Thus, for any sufficiently large j, $T_j(z) = T_\infty(z)$ for all $z \leq dy$. Therefore for every sufficiently large j, $J_j(y) \leq dT_\infty(dy)$ and $T_\infty(y) \leq dJ_j(dy)$. Therefore $J_\infty(y) \leq dT_\infty(dy)$ and $T_\infty(y) \leq dJ_\infty(dy)$.

Property Q5 follows directly from the definition of M'_∞ and Lemma 3.0.6.

We now prove Property Q6. In order to determine whether a phase 1 configuration $\mathbf{c}$ is acceptable, we first check if all tapes of $\mathbf{c}$ except the left half of the input tape (i.e. tape 1) and the left half of the history tape (i.e. tape $(2(k+1)-1)$ are empty. If not, then $\mathbf{c}$ is not an acceptable configuration because no computation starting with $\mathbf{c}$ can ever exit phase 1. If all tapes besides the first and the $(2(k+1)-1)$st are empty, then $\mathbf{c}$ is an acceptable configuration of M'_∞ if and only if the word w written on the input tape of $\mathbf{c}$ is an accepted input of M_∞. This is because there is a computation of M'_∞ beginning with $\mathbf{c}$ that simply erases the left half of the history tape of $\mathbf{c}$, leaving the machine in an input configuration $\mathbf{c}'$ for the input word $\mathbf{u}$. Since M_∞ and M'_∞ accept the same language, M'_∞ accepts $\mathbf{c}'$ (and therefore $\mathbf{c}$) if and only if $w \in L$ accepted by M_∞. If L is decidable, we can decide whether $\mathbf{c}$ is acceptable.

A phase 3 configuration is acceptable if and only if all tapes except the 1st (left half of input tape) and $2(k+1)$st (right half of the history tape) are empty.

If a phase 2 configuration $\mathbf{c}$ is an acceptable configuration of M_∞, then there must be a phase 2 computation that starts in $\mathbf{c}$ and ends in either a phase 1 configuration or a phase 3 configuration. A reduced Phase 2 computation beginning with $\mathbf{c}$ must execute one of the sequences of commands written in the two halves of the history tape in $\mathbf{c}$. Recall that each command symbol of M_∞ written in these history tapes contains a complete description of the corresponding M_∞ command. Therefore we can effectively recover the finite set of M'_∞ commands that can be executed in a reduced M'_∞ computation that begins with $\mathbf{c}$. We can then

effectively check if applying the either of the sequences of commands written in the history tapes ends in either a phase 1 or a phase 3 configuration. If not, then $\mathbf{c}$ is not an acceptable configuration. If a phase 3 configuration $\mathbf{c}'$ can be reached, then $\mathbf{c}$ is acceptable if and only if $\mathbf{c}'$ is acceptable, which is decidable by the previous paragraph. If a phase 1 configuration $\mathbf{c}'$ can be reached then $\mathbf{c}$ is acceptable if and only if $\mathbf{c}'$ is acceptable. Since $\mathbf{c}$ is a phase 1 configuration, we can effectively decide whether $\mathbf{c}'$ is acceptable.

□

Chapter 4

S-Machines

In [10], a model of computation called the S-machine is introduced. It is shown that, for any symmetrized Turing machine M' satisfying Lemma 3.1[10], an S machine $S(M')$ can be constructed that simulates M'. The purpose of this section is to define S-machines, and to state some facts about the construction of $S(M')$ that will be used later.

An S-machine is a group presentation of an HNN-extension of a free group that satisfies some additional conditions. Certain words in the generators of the base group of such an HNN-extension are thought of as configurations of the machine. As in [10], we call these configurations *admissible words*. The stable letters of the HNN-extension are thought of as the commands of the machine. These stable letters act on the set of admissible words by conjugation. We formalize this idea below.

A *hardware* of an S-machine is a free group $G = F(\hat{A} \cup \hat{Q})$, where $\hat{A}$ and $\hat{Q}$ are disjoint sets of positive generators. The hardware will be the base group of the HNN-extension. We will call $\hat{A}$ the set of tape letters, and $\hat{Q}$ the set of state letters. The set $\hat{Q}$ is the union of k disjoint sets: $\hat{Q} = \hat{Q}_1 \cup ... \cup \hat{Q}_k$.

A reduced word w in the generators of G is an *admissible word* of G if it has the form

$$w = r_1 w_1 r_2 w_2 ... r_{k-1} w_{k-1} r_k,$$

where $r_i \in \hat{Q}_i$ and $w_i \in F(\hat{A})$. If $i \leq j$, then a subword of an admissible word w of the form $r_i w_i ....r_j$ is called the (i, j)-*subword* of w. Note that if $i = j$ then an (i, j)-subword consists of only a single $\hat{Q}_i$ letter.

A necessary condition for an HNN extension $\mathcal{N}$ of a hardware G to be an S-machine is that $\mathcal{N}$ must be constructed by adjoining a finite set of stable letters to G. We will denote this set of stable letters $\hat{\Theta} = \{\rho_1, ..., \rho_n\}$, or the set of command letters of $\mathcal{N}$.

If $\mathcal{N}$ is an S-machine with hardware G, then $\mathcal{N}$ is of the form $\langle G, \rho_1, ..., \rho_n | H_1, ..., H_n \rangle$ where each H_i is a disjoint set of relators. The set H_i corresponds to the stable letter ρ_i: every relator in H_i has the form $\rho_i^{-1} x \rho_i = y$, where x, y are words in the generators of G. Also, the letter ρ_i does not appear in the relators of H_j if $j \neq i$. The set of relators H_i is called a *command* of $\mathcal{N}$.

There are two types of relators in each H_i: transition relators and auxiliary relators. The auxiliary relators are $\{\rho_i a \rho_i^{-1} = a | a \in \hat{A}\}$. The transition relators of H_i have the form $\rho_i^{-1} \mathbf{u} \rho_i = a \mathbf{v} b$ where, for some $m \leq n$, the words $\mathbf{u}, \mathbf{v}$ are both (m, n)-subwords of admissible words and $a, b \in \hat{A}^{\pm 1} \cup \{\varepsilon\}$.

For each $\hat{Q}_j$, there is exactly one transition relator in each H_i in which a $\hat{Q}_j$ letter appears. Furthermore, if $\rho_i^{-1} \mathbf{u} \rho_i = a \mathbf{v} b$ is the single H_i relator in which a $\hat{Q}_j$ letter appears then there is exactly one $\hat{Q}_j$ letter in each of the words $\mathbf{u}$ and $\mathbf{v}$ (this follows from the fact that both $\mathbf{u}$ and $\mathbf{v}$ are (m, n) subwords of admissible words).

Lemma 4.0.8. *Let $\mathcal{N} = \langle G, \rho_1, ..., \rho_n | H_1, ..., H_n \rangle$ be an S-machine as defined above. Then $\mathcal{N}$ is an HNN-extension of G.*

Proof. Suppose $\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_s$ are the relators of H_i. Then each $\mathbf{r}_\ell$ is of the form $\rho_i^{-1} \mathbf{x}_\ell \rho_i = \mathbf{y}_\ell$ where $\mathbf{x}_\ell, \mathbf{y}_\ell$ are words in the generators of G. Note that each $\mathbf{x}_\ell$, $\ell = 1, ..., s$, is either a letter of $\hat{A}$ (if $\mathbf{r}_\ell$ is an auxiliary relator) or an (m_ℓ, n_ℓ)-subword of an admissible word (if $\mathbf{r}_\ell$ is a transition relator). Each such (m_ℓ, n_ℓ)-subword contains at least one $\hat{Q}$ letter, and no two distinct such (m_ℓ, n_ℓ)-subwords share a common $\hat{Q}$ letter. Therefore the elements $\mathbf{x}_1, ..., \mathbf{x}_s$ are Nielsen reduced and they are free generators of a subgroup of G.

We will show that the elements $\mathbf{y}_1, ..., \mathbf{y}_s$ also freely generate a subgroup of G. For all $a \in \hat{A}$ there exists an auxiliary relator $\mathbf{r}_\ell \in H_i$ such that $\mathbf{y}_\ell = a$. If $\mathbf{r}_\ell$ is a transition relator then $\mathbf{y}_\ell$ is a word of the form $a \mathbf{v} b$, where $\mathbf{v}$ is an (m_ℓ, n_ℓ)-subword of an admissible word and $a, b \in \hat{A}^{\pm 1} \cup \{\varepsilon\}$. If, for a given $\mathbf{y}_\ell$, the letter a (or b) is not ε then we can perform a Neilson reduction on $\mathbf{y}_1, ..., \mathbf{y}_s$ to remove a (or b) from $\mathbf{y}_\ell$. After we remove all such a and b letters from the words $\mathbf{y}_1, ..., \mathbf{y}_s$, the resulting set of words is Neilson reduced by the argument in the above paragraph. Therefore the elements $\mathbf{y}_1, ..., \mathbf{y}_s$ freely generate a subgroup of G, and the map $\mathbf{x}_\ell \mapsto \mathbf{y}_\ell$ induces an isomorphism of subgroups. We conclude that $\mathcal{N}$ is an HNN-extension of G.

□

If $\mathcal{N}_1$ is an S-machine, we say that an S-machine $\mathcal{N}_2$ is a *submachine* of $\mathcal{N}_1$ if every generator of the hardware of $\mathcal{N}_2$ is a generator of the hardware of $\mathcal{N}_1$, every command letter of $\mathcal{N}_2$ is a command letter of $\mathcal{N}_1$, and every relator of $\mathcal{N}_2$ is a relator of $\mathcal{N}_1$. In this case we write $\mathcal{N}' \sqsubseteq \mathcal{N}$.

A *computation* of an S-machine $\mathcal{N}$ is a sequence of admissible words $W_1, ..., W_n$ such that for each $i \geq 2$, the equation $W_i = \rho^{\pm 1} W_{i-1} \rho^{\mp 1}$ holds in $\mathcal{N}$ for some command letter ρ of $\mathcal{N}$.

For an S-machine $\mathcal{N}$, we may designate a single admissible word W_0 as the "accept configuration" of the machine. We say that an admissible word W is *acceptable* by $\mathcal{N}$ if there is a computation of $\mathcal{N}$ that begins with W and ends with W_0.

We define the complexity functions of an S-machine the same way we defined them for Turing machines: simply replace the word "configuration" with "admissible word".

In [10], the authors construct an S-machine $S(M')$ to simulate the symmetrization M' of an arbitrary Turing machine M. We will state some facts about this construction, beginning with a precise description of what "simulates" means in this context.

The simulation of M' by $S(M')$ relies on an injective map σ from the set of configurations of M' to the set of admissible words of $S(M')$. For a configuration $c = (\alpha \mathbf{u}_1 q_1 \omega, ..., \alpha \mathbf{u}_k q_k \omega)$ of M', the admissible word $\sigma(\mathbf{c})$ is the concatenation of $(k+2)$ many subwords. The first subword is

$$E(0)\ \hat{\alpha}^n\ x(0)\ F(0).$$

In the above subword, the letter $\hat{\alpha}$ is a tape letter. All other letters in the above subword are head letters. Also, $n = |\mathbf{u}_1| + \cdots + |\mathbf{u}_k|$. The next k subwords have the form

$$E(i)\ \mathbf{u}_i\ x(i)\ F_{q_i}(i)\ E'(i)\ p(i)\ \delta^{|\mathbf{u}_i|}\ \Omega_i\ F'_{q_i}(i),$$

for $i = 1, ..., k$. The ith such subword corresponds to the ith tape in $\mathbf{c}$. The word $\mathbf{u}_i$ is identical to the word written in the ith tape of $\mathbf{c}$, and Ω_i is a word in the head letters of $S(M)$. The letter δ and the letters of $\mathbf{u}_i$ are tape letters. All other letters in the above word are head letters. The last subword of $\sigma(\mathbf{c})$ is

$$E(k+1)\ \hat{\omega}^n\ x(0)\ F(k+1)$$

.

The letter $\hat{\omega}$ is a tape letter. All other letters in the above word are head letters. Also, $n = |\mathbf{u}_1| + ... + |\mathbf{u}_k|$. This definition of $\sigma(\mathbf{c})$ appears on page 400 of [10]. We differ slightly from the notation of [10] by using the letters $\hat{\alpha}$ and $\hat{\omega}$ in $\sigma(\mathbf{c})$ where [10] used α and ω. The purpose of this is to differentiate $\hat{\alpha}$ and $\hat{\omega}$ from the α and ω letters that appear in $\mathbf{c}$.

Lemma 4.0.9. *Given an admissible word W of $S(M')$, it is possible to decide in linear time whether or not $W = \sigma(\mathbf{c})$ for some configuration $\mathbf{c}$ of M'. Also, in the case that $W = \sigma(\mathbf{c})$, it is possible to effectively recover $\mathbf{c}$ from W in linear time.*

Proof. The first sentence of this lemma follows immediately from the definition of $\sigma(\mathbf{c})$. If $W = \sigma(\mathbf{c})$, note that the word between $E(i)$ and $x(i)$ is $\mathbf{u}_i$, and the subscript indices of the $F_{q_1}(1), ..., F_{q_k}(k)$ letters are the states $q_1, ..., q_k$. Thus we can recover $\mathbf{c}$ from $\sigma(\mathbf{c})$ in linear time.

□

We can now state precisely what it means to say that $S(M')$ simulates M'. We first introduce the following notation. If w is a word of a free group $\langle Y \rangle$, then we use the notation $\|w\|$ to denote the sum of the exponents of the letters appearing in w. For example, if a, b, c are positive generators of $\langle Y \rangle$ then $\|ab^{-1}c\| = 1$.

Lemma 4.0.10. *Suppose M is a k-tape Turing machine with symmetrization M'. Then there is an S-machine $S(M')$ with the following properties:*

(S1) A configuration $\mathbf{c}$ of M' is accepted by M' if and only if there is a computation of $S(M')$ that begins with $\sigma(\mathbf{c})$ and ends with $W_0 := \sigma(\mathbf{c}_0)$, where $\mathbf{c}_0$ is the accept configuration of M'.

(S2) If $\mathbf{c}$ is an acceptable configuration of M' then there exists an injective map ψ from the set of accepting computations of M' starting with $\mathbf{c}$ to the set of accepting computations of $S(M')$ starting with $\sigma(\mathbf{c})$

with the following preperties. Suppose C' is an accepting computation of M' starting with $\mathbf{c}$, and C' has length T and space S. Suppose $\psi(C')$ has length L and area A. Then $\varepsilon_1 S^3 \leq L \leq \varepsilon_2 TS^2$ and $\varepsilon_3 S^4 \leq A \leq \varepsilon_4 TS^3$ for some positive constants $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4$.

Proof. Properties S1 and S2 are proven in Proposition 4.1[10].

□

Corollary 1. The S-machine $S(M')$ also fulfills the following three properties.

(R1) For Turing machines M_1' and M_2' satisfying Lemma 3.0.7, if $M_1' \sqsubseteq M_2'$ then $S(M_1') \sqsubseteq S(M_2')$.

(R2) There is a linear time algorithm that takes as input a description of a Turing machine M satisfying Lemma 3.0.7 and outputs a description of $S(M')$.

(R3) The constants $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4$ in property S2 of Lemma 4.0.10 are uniform for all k-tape Turing machines M' satisfying Lemma 3.0.7.

Proof. Properties R1 and R2 follow directly from the definition of $S(M')$ in [10] and from Lemma 3.0.6. Property R3 follows from inspection of the constants used in the proof of Proposition 4.1[10].

□

There are some additional facts about the $S(M')$ construction that we will require. Every command letter of $S(M')$ is indexed by a command letter of M'. For each positive command letter τ in M', there is a finite set of command letters of $S(M')$ that are indexed by τ. For each command τ of M' of the form (3.0.4), $S(M')$ contains a collection of submachines denoted in [10] by $S_4(\tau)$, $S_9(\tau)$, $R_4(\tau)$, $R_{4,9}(\tau)$, and $R_9(\tau)$. The command letters of these submachines are exactly the command letters of $S(M')$ that are indexed by τ. For each command τ' of M' of the form (3.0.5), $S(M')$ contains a submachine denoted in [10] by $P(\tau')$. The positive command letter (there is only one) of this submachine is the only command letter of $S(M')$ indexed by τ'. The purpose of these sub-machines is to allow $S(M')$ to pass from $\sigma(\mathbf{c})$ to $\sigma(\mathbf{c}')$ if and only if the M' command τ (or τ') takes $\mathbf{c}$ to $\mathbf{c}'$. A concise description of the function of each of these machines can be found on pages 397-398 of [10]. Their formal definitions are located on pages 374-396 of [10].

Lemma 4.0.11. *There is a linear-time algorithm that, when given as input an M' command τ of type (3.0.4) outputs the set of commands of the submachines $S_4(\tau)$, $S_9(\tau)$, $R_4(\tau)$, $R_{4,9}(\tau)$, $R_9(\tau)$ of $S(M')$. When given as input a command τ' of type (3.0.5), the algorithm outputs the set of commands of $P(\tau')$.*

Proof. This follows directly from the definition of the machines $S_4(\tau)$, $S_9(\tau)$, $R_4(\tau)$, $R_{4,9}(\tau)$, $R_9(\tau)$, and $P(\tau')$ in [10].

□

For any command τ or τ' in M' (of type (3.0.4) or (3.0.5) respectively), the submachines $R_4(\tau)$, $R_9(\tau)$, and $P(\tau')$ each contain a single positive command. In the case of each of these three submachines, we will use the same notation to denote both the machine and its single positive command. For example if we say that the command $R_4(\tau)^{-1}$ can be applied to an admissible word, then the command we are referring to is the inverse of the single positive command contained in the submachine $R_4(\tau)$.

The state letters of $S(M')$ are divided into two types: standard and non-standard. Each non-standard state letter of $S(M')$ is indexed by a positive command of M'. The standard state letters are not indexed by any commands of M'. A complete description of the state letters of $S(M')$ can be found on page 397 of [10].

Lemma 4.0.12. *Suppose W is an admissible word of $S(M')$ such that all state letters appearing in W are standard. Then the only commands of $S(M')$ that may be applied to W are $R_4(\tau)$, $R_9(\tau)^{-1}$, or $P(\tau')^{\pm 1}$ for commands τ of type (3.0.4) or τ' of type (3.0.5).*

Proof. This follows immediately from the description of the commands of $S(M')$ on pages 397-398 of [10].

□

In [10], the authors call an admissible word W *normal* if it fulfills certain properties (the definition is on page 400 of [10]). For our purposes, the details of this definition are not important. It will suffice to note that it is stated on page 403 of [10] that the commands of $S(M')$ take normal words to normal words, and that every admissible word $\sigma(\mathbf{c})$ is a normal word.

Lemma 4.0.13. *Let W be an admissible word of $S(M')$. Suppose that W is positive and normal. Suppose also that one of the commands $R_4(\tau)$, $R_9(\tau)^{-1}$, $P(\tau')$ can be applied to W. Then $W = \sigma(\boldsymbol{c})$ for some configuration $\boldsymbol{c}$ of M'.*

Proof. This is Lemma 4.15[10].

□

Corollary 2. Let W be an admissible word of $S(M')$. Suppose that W is positive and normal, and that a command $P(\tau')^{-1}$ can be applied to W. Then $W = \sigma(\mathbf{c})$ for some configuration $\mathbf{c}$ of M'.

Proof. In [10], Lemma 4.15[10] is stated without proof because it follows immediately from the definition of $S(M')$. Similarly, this corollary (not stated in [10]) follows immediately from the definition of $S(M')$ as well.

□

Lemma 4.0.14. *Suppose W is an admissible word of $S(M')$ and there is a computation C of $S(M')$ that starts with W_0 and ends with W. If every state letter appearing in W is standard, then $W = \sigma(\boldsymbol{c})$ for some configuration $\boldsymbol{c}$ of M'.*

Proof. Suppose Θ_1 is the set of all commands of M' of the form (3.0.4) and Θ_2 is the set of all commands of M' of the form (3.0.5). Any computation of $S(M')$ is a concatenation of computations of the submachines

$$\{S_4(\tau), S_9(\tau), R_4(\tau), R_{4,9}(\tau), R_9(\tau), P(\tau') \mid \tau \in \hat{\Theta}_1, \tau' \in \hat{\Theta}_2\}.$$

We write C as such a concatenation:

$$C = C_1 ... C_N.$$

Here each C_i is a nonempty computation of one of the above submachines of $S(M')$. The computations C_i and C_{i+1} are not computations of the same submachine.

The submachine that executes the computation C_i is denoted by $\chi(C_i)$. The proof of Proposition 4.1[10] describes the structure of the sequence of submachines $\chi(C) = \chi(C_i) ... \chi(C_N)$. It is stated on page 406 of [10] that the sequence $\chi(C)$ is a concatenation of subsequences $D_1(\tau_1), D_2(\tau_2), ..., D_\ell(\tau_\ell)$ where for each $i < \ell$, the subsequence $D_i(\tau_i)$ is a sequence of one of the following three forms:

$R_4(\tau_i),\ S_4(\tau_i),\ R_{4,9}(\tau_i),\ S_9(\tau_i),\ R_9(\tau_i)$

$R_9(\tau_i),\ S_9(\tau_i),\ R_{4,9}(\tau_i),\ S_4(\tau_i),\ R_4(\tau_i)$

$P(\tau_i)$.

The subsequence $D_l(\tau_l)$ is a prefix of one of these above forms. Since C ends in W, and W has all state letters standard, it follows from Lemma 4.0.12 that $\chi(C_N)$ must be either $P(\tau_i)$, $R_4(\tau_i)$ or $R_9(\tau_i)^{-1}$. Thus, by the above stated structure of $\chi(C)$, $\chi(C_{N-1})$ may not be $R_{4,9}$. It is stated on page 406 of [10] that if $\chi(C_{N-1}) \neq R_{4,9}$, then all words in $\bigcup_{n=1}^{N-1} C_i$ are positive. By definition, each of the rules $P(\tau_i)^{\pm 1}$, $R_4(\tau_i)$, and $R_9(\tau_i)^{-1}$ take positive admissible words to positive admissible words. We conclude that W is positive. Since C began with W_0, all words in C are normal. The result now follows from Lemma 4.0.13.

□

The following lemma is stated in the proof of Proposition 4.1[10] on page 408.

Lemma 4.0.15. *If W is an acceptable admissible word of $S(M')$, then there is a computation C_1'' of $S(M')$ that takes W to $\sigma(\boldsymbol{c})$ for some acceptable configuration $\boldsymbol{c}$ of M'. The computation C_1'' is composed of computations of $S_4(\tau)$, $S_9(\tau)$, $R_4(\tau)$, $R_{4,9}(\tau)$, $R_9(\tau)$ for some command τ of M'. Also, $|\boldsymbol{c}| < \|W\| < |W|$. The area of C_1'' does not exceed $O((\sigma(\boldsymbol{c}) + |W|)^3) \leq O(|W|)^3$.*

Corollary 3. The constants witnessing the use of the big-O notation in Lemma 4.0.15 are uniform for all k-tape machines M.

Proof. It is stated on page 408 of [10] that the constants witnessing this use of the big-O notation are derived from the bounds given in Lemmas 4.6[10], 4.12[10], and 4.16[10]. It is straightforward to check that all bounds appearing in these lemmas are uniform for all k-tape Turing machines M.

□

The following lemma follows directly from the construction of $S(M')$ in [10]. However there is no particular page number that can be cited to verify this lemma since it relies on the entirety of the construction of $S(M')$.

Lemma 4.0.16. *There is a constant bound b_k such that for any k-tape Turing machine M the relators in the presentation $S(M')$ have length less than b_k.*

Chapter 5

Union S-Machines

Given a union machine with symmetrization M'_∞, we define the group presentation $S(M'_\infty)$ as follows. The generating set of $S(M'_\infty)$ is equal to the union of the generating sets of the S-machines $\{S(\bigcup_{i=1}^{n} M'_i)|i \in \mathbf{N}\}$. Similarly, the set of relators of $S(M'_\infty)$ is equal to the union of the sets of relators of the S-machines $\{S(M'_i)|i \in \mathbf{N}\}$. Informally, this is the same presentation one would get by applying the S-machine construction in [10] to M'_∞ (this is possible since the construction of $S(M')$ does not rely at all on the finiteness of M). As the next lemma shows, this presentation behaves analogously to the standard S-machines defined in the previous section.

Lemma 5.0.17. *For a union machine M_∞ with symmetrization M'_∞ the group presentation $S(M'_\infty)$ satisfies the following properties:*

(U1) There is an injective map σ from the set of configurations of M'_∞ to the set of admissible words of $S(M'_\infty)$ such that σ satisfies the following requirement. A configuration $\boldsymbol{c}$ of M'_∞ is accepted by M'_∞ if and only if there is a computation of $S(M'_\infty)$ that takes $\sigma(\boldsymbol{c})$ to $\sigma(\boldsymbol{c}_0)$, where $\boldsymbol{c}_0$ is the accept configuration of M'_∞.

(U2) The area function of $S(M'_\infty)$ is equivalent to T^4 where T is the time function of M_∞.

(U3) If M_∞ is enumerable then $S(M'_\infty)$ is enumerable.

Proof. The map σ mentioned in this lemma is defined identically to the map σ defined in section 4 on page 15.

To prove property U1, note that a configuration $\mathbf{c}$ of M'_∞ is accepted by M'_∞ if and only if there is some Turing machine M'_i that accepts $\mathbf{c}$. By Lemma 4.0.10, the machine M'_i accepts $\mathbf{c}$ if and only if $S(M'_i)$ accepts

the admissible word $\sigma(\mathbf{c})$. The machine $S(M'_\infty)$ accepts $\sigma(\mathbf{c})$ if and only if there is some $S(M'_i)$ that accepts $\sigma(\mathbf{c})$ (because all accepting computations have finite length). Part 1 is proved.

To prove property U2, consider an arbitrary accepted admissible word W of $S(M'_\infty)$. Suppose $W = \sigma(\mathbf{c})$ for some configuration $\mathbf{c}$ of M'_∞. By property U1, $\mathbf{c}$ is an accepted configuration of M'_∞. Suppose C' is a minimal area accepting computation for $\mathbf{c}$ in M'_∞, with length $\mathcal{T}$ and space $\mathcal{S}$. There is a sufficiently large i that C' is a computation of M'_i. We can now apply Lemma 4.0.10 to M'_i. As in property S2 of Lemma 4.0.10, the computation $\psi(C')$ of $S(M'_i)$ begins with $\sigma(\mathbf{c})$ and ends with $\sigma(\mathbf{c}_0) = W_0$. If A is the area of $\psi(C')$ then $\varepsilon_3\mathcal{S}^4 \leq A \leq \varepsilon_4\mathcal{T}\mathcal{S}^3$. Since every M'_i has the same number of tapes, the constants ε_3 and ε_4 are uniform for all M'_i (by property R3 of Corollary 1). This proves that if f is the area function of $S(M'_\infty)$ then $f \succeq T(n)^4$ (since the space and time functions of M'_∞ are equivalent by Lemma 3.0.7). To get the other direction of the equivalence, we must examine the case in which W is not in the image of σ.

Suppose W is an acceptable admissible word of M'_∞ and W is not in the image of σ. We choose n sufficiently large that W is an acceptable admissible word of $S(M'_i)$. By Lemma 4.0.15, there is a computation C''_1 of $S(M'_i)$ that begins with W and ends with an accepted admissible word $\sigma(\mathbf{c})$ of $S(M'_i)$, and the area of C''_1 does not exceed $O((|W|)^3)$. Additionally, Lemma 4.0.15 states that $|\mathbf{c}| < |W|$, and therefore $|\sigma(\mathbf{c})| \leq O(|W|)$. We can now use the argument from the previous paragraph to conclude that there exists an accepting computation for W with area at most $O((|W|)^3) + \varepsilon_4 T^4(|\sigma(\mathbf{c})|)$. Recall that the constants witnessing this big O notation are uniform for all $S(M'_i)$, and that T grows at least as fast as the linear function $f(x) = x$. Therefore $f \preceq T(n)^4$ and part 2 is proved.

Part 3 follows immediately from Corollary 1, item 2.

□

Lemma 5.0.18. *If the configuration problem is solvable for M'_∞, then the configuration problem is solvable for $S(M'_\infty)$.*

Proof. Let W be an accepted admissible word of $S(M'_\infty)$. If $W = \sigma(\mathbf{c})$ for some configuration $\mathbf{c}$ of M'_∞, then by Lemma 4.0.9 we can recover $\mathbf{c}$ from W. Since $\mathbf{c}$ is accepted by M'_∞ if and only if $\sigma(c)$ is accepted by $S(M'_\infty)$, we can decide whether $W = \sigma(c)$ is accepted by $S(M'_\infty)$.

If W is not equal to $\sigma(\mathbf{c})$ for any configuration c of M'_∞, then by Lemma 4.0.14, not all state letters of W are standard. We choose a non-standard state letter of W and look at its Θ index τ. By Lemma

4.0.11, we can use τ to recover the set of commands of the sub-machines $S_4(\tau)$, $S_9(\tau)$, $R_4(\tau)$, $R_{4,9}(\tau)$, $R_9(\tau)$. We effectively construct all positive sequences of commands of these sub-machines of length not exceeding $O(|W|^2)$. This is possible because there are finitely many such sequences. We then check if any of these sequences of commands can be applied to W to reach an admissible word of the form $\sigma(\mathbf{c})$. If not, then by Lemma 4.0.15, W is not an accepted admissible word of $S(M'_\infty)$. If so, then we collect the finitely many configurations $\mathbf{c}$ of M'_∞ such that $\sigma(\mathbf{c})$ is reachable from W by applying such a sequence of commands. The admissible word W is accepted by $S(M'_\infty)$ if and only if at least one of these configurations $\mathbf{c}$ is an accepted configuration of M'_∞. Since the configuration problem is solvable for M'_∞, the proof is complete.

□

Chapter 6

Van Kampen Diagrams

In this section we define the terms and notation that we will use concerning van Kampen diagrams. We will keep our terms and notation as close as possible to those used in [10]. Let $P = \langle X \parallel R \rangle$ be a group presentation.

A *van Kampen diagram* (or often just a diagram) over the presentation P is a planar, finite, connected and simply connected 2-complex Δ. The edges of Δ are oriented, and each oriented edge of Δ is labeled by an element of $X^{\pm 1}$. The label of an oriented edge e is denoted $\mathrm{Lab}(e)$. If $\mathrm{Lab}(e) = x$, then $\mathrm{Lab}(e^{-1}) = x^{-1}$. If $p = e_1 \dots e_k$ is a path of edges in Δ then the label of p is $\mathrm{Lab}(p) = \mathrm{Lab}(e_1) \dots \mathrm{Lab}(e_k)$. We denote the boundary path of Δ by $\partial\Delta$. For a 2-cell π of Δ, we denote the boundary path of π by $\partial(\pi)$ (we will often write "boundary" for boundary path). We will always assume that boundary paths of diagrams and of 2-cells are oriented in the clockwise direction. For every 2-cell π of Δ, $\mathrm{Lab}(\partial\pi)$ is an element of the relator set R. The *contour* of a cell or diagram is the union of its boundary and the inverse of its boundary. An *undirected edge* of Δ is the union of an edge of Δ with its inverse. An edge e of Δ is a *boundary edge* of Δ if e is contained in $\partial\Delta$. If an edge e of Δ is not a boundary edge then e is an *interior edge*. We call the initial and final vertices of an edge e the *endpoints* of e.

A pair of 2-cells π_1, π_2 in a diagram Δ is called a *reducible pair* if there is an edge e in Δ such that $\partial\pi_1 = ve$, $\partial\pi_2 = e^{-1}v'$ and $\mathrm{Lab}(v) = \mathrm{Lab}(v')^{-1}$. A diagram is *reduced* if it contains no reducible pairs.

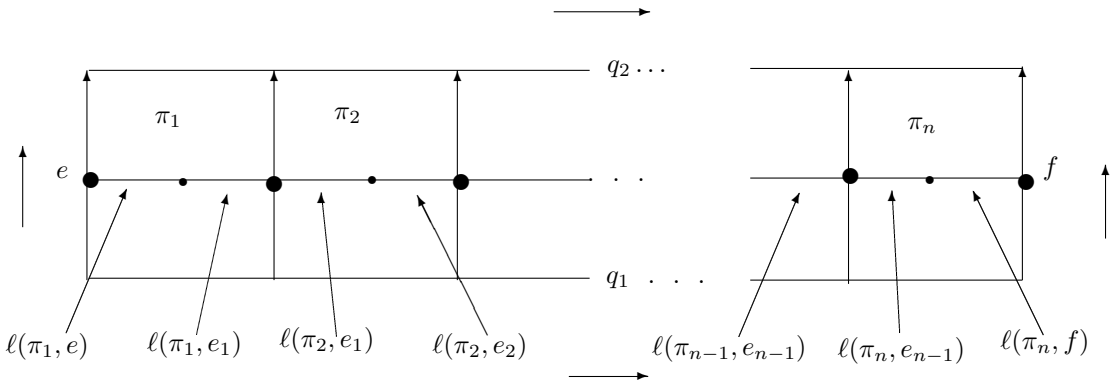
The length of a path p, denoted $|p|$, is the number of edges p contains. The *area* of a diagram Δ is the number of 2-cells Δ contains. If we say that Δ is a *minimal area* diagram over some presentation P, we mean that Δ is the minimal area P diagram with boundary label $\mathrm{Lab}(\partial\Delta)$.

A *spherical diagram* over the presentation P is defined almost identically to a diagram over P, with the one exception that a spherical diagram is spherical instead of planar.

Let $\mathbf{S}$ be a subset of the generating set of a presentation P. An $\mathbf{S}$-band $\mathcal{B}$ over P is a sequence of 2-cells $\pi_1, ..., \pi_n$ in a van Kampen diagram such that:

- For $i = 1, \ldots, (n+1)$, the boundaries $\partial\pi_i$ and $(\partial\pi_{i+1})^{-1}$ share a common edge labeled by a letter from $\mathbf{S}$.

- For $i = 1, ..., n$, the boundary $\partial\pi_i$ contains exactly two $\mathbf{S}$-edges (i.e. edges labeled by a letter from $\mathbf{S}$). The figure below illustrates this concept. In this figure, the edges $e, e_1, ..., e_{n-l}, f$ are $\mathbf{S}$-edges, and the line $\ell(\pi_i, e_i)$ connects a fixed point in the interior of π_i cells with a fixed point in the interior of e_i.



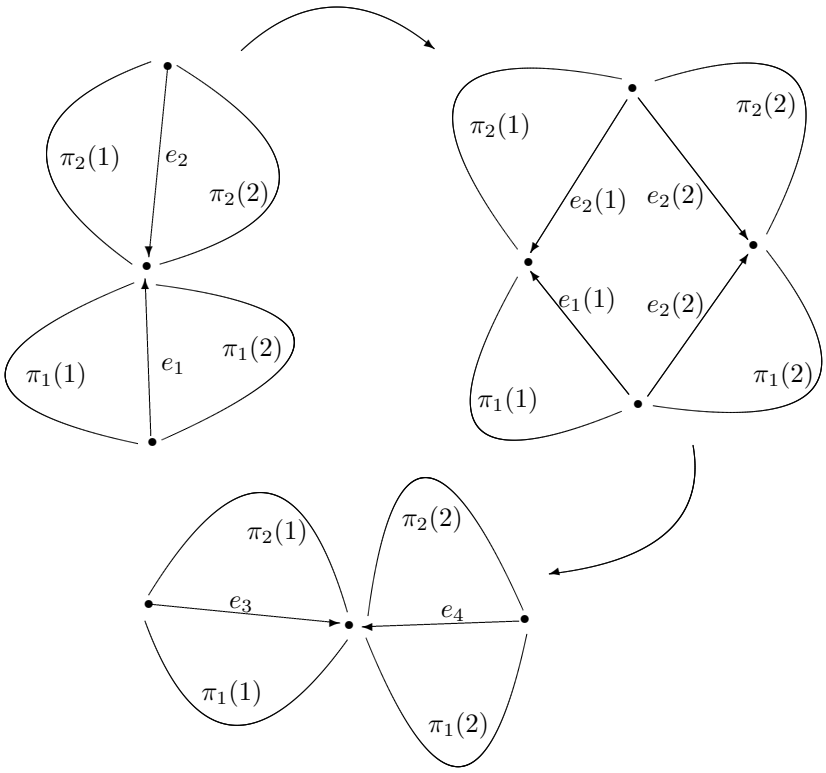
The broken line formed by the lines $\ell(\pi_i, e_i), \ell(\pi_i, e_{i-1})$ connecting points inside neighboring cells is called the *median* of the band $\mathcal{B}$. The $\mathbf{S}$-edges e and f are called the *start* and *end* edges of the band. If $\mathcal{B} = (\pi_1, ..., \pi_n)$ is an $\mathbf{S}$-band then $(\pi_n, \pi_{n-1}, ..., \pi_1)$ is also an $\mathbf{S}$-band. This band is called the *inverse* of $\mathcal{B}$ and is denoted $\mathcal{B}^{-1}$. The start edge of $\mathcal{B}$ is the end edge of $\mathcal{B}^{-1}$, and the end edge of $\mathcal{B}$ is the start edge of $\mathcal{B}^{-1}$.

The clockwise boundary of the diagram formed by the cells $\pi_1, ..., \pi_n$ of $\mathcal{B}$ has the form $eq_2f^{-1}q_1^{-1}$. We call q_1 the *bottom* of $\mathcal{B}$ and q_2 the *top* of $\mathcal{B}$, denoted $\mathbf{bot}(\mathcal{B})$ and $\mathbf{top}(\mathcal{B})$.

We say that two bands *cross* if their medians cross. We say that a band is an *annulus* if its median is a closed curve (i.e. if $e = f$).

We now define a type of surgery on diagrams called a *folding surgery*. Suppose that $p = e_1e_2^{-1}$ is a path in Δ such that the initial vertex of e_1 is distinct from both endpoints of e_2 and the initial vertex of e_2 is distinct

from both endpoints of e_1. Suppose also that both e_1 and e_2 have label x, so the label of p is xx^{-1}. Let $\mathbf{C}$ be the set which contains each 2-cell of Δ and which also contains the complement of Δ in the plane, denoted Δ^c. We define the contour of Δ^c to be identical to the contour of Δ. Let $\pi_1(1), \pi_1(2), \pi_2(1), \pi_2(2) \in \mathbf{C}$ be such that e_1 is in the contour of $\pi_1(1)$ and $\pi_1(2)$, while e_2 is in the contour of $\pi_2(1)$ and $\pi_2(2)$. Note that these elements of $\mathbf{C}$ are not necessarily all distinct. We define a *folding surgery* on Δ at p as follows.



We cut along the path $p = e_1 e_2^{-1}$ to create a hole in Δ. In the resulting annular diagram, the four edges adjacent to this hole form the cyclic path

$$e_1(1)e_2(1)^{-1}e_2(2)e_1(2)^{-1},$$

where the edge $e(i)_j$ is in the contour of $\pi(i)_j$. Note that the edges $e_1(1)$ and $e_2(1)$ have same label, as do the edges $e_1(2)$ and $e_2(2)$. We complete the folding surgery by gluing $e_1(1)$ to $e_2(1)$ and gluing $e_1(2)$ to $e_2(2)$.

If Δ is a spherical diagram, then we define a folding surgery at p identically.

In later sections we will use some results about presentations satisfying the small-cancelation condition

$C'(\lambda)$ for $0 < \lambda < 1$. Recall that a presentation $P = \langle X \| R \rangle$ satisfies $C'(\lambda)$ for $0 < \lambda < 1$ if for any two distinct elements $r_1, r_2 \in R$ with $r_1 = bc_1$ and $r_2 = bc_2$ the inequalities $|b| < \lambda |r_1|$ and $|b| < \lambda |r_2|$ hold. The following lemma is well-known. It follows immediately from the proof of Theorem 4.4 of [7].

Lemma 6.0.19. *Suppose a presentation $P = \langle X \| R \rangle$ satisfies the small-cancelation condition $C'(\lambda)$ for $\lambda \leq \frac{1}{6}$. If Δ is a reduced P diagram with cyclically reduced boundary label and area ≥ 1 then there is a 2-cell π of Δ such that $\partial\pi$ shares a common subpath p with $\partial\Delta$ and $|p| > \frac{1}{2}|\partial\pi|$.*

Lemma 6.0.20. *Suppose a presentation P satisfies the small-cancelation condition $C'(\lambda)$ for $\lambda \leq \frac{1}{6}$. Suppose Δ is a reduced diagram over P with area ≥ 1. Then the number of undirected edges in Δ is bounded above by $d|\partial\Delta|$ for some constant $d > 1$. Also, reduced spherical diagrams over P do not exist.*

Proof. If e is an edge of a diagram Δ such that ee^{-1} is a subpath of the boundary of Δ, we call e an *extremal edge*. If a diagram contains an extremal edge then its boundary label is not cyclically reduced.

We begin by observing that if a diagram Δ does not have a cyclically reduced boundary label then Δ can be transformed into a diagram with cyclically reduced boundary label by performing folding surgeries and/or removing extremal edges. Folding surgeries do not change the number of undirected edges contained in Δ nor do they change the boundary length of Δ. Removing an extremal edge reduces the number of undirected edges in Δ by one and the boundary length of Δ by two.

We now prove the first claim by induction on the area of Δ. If Δ contains only a single 2-cell then the claim is trivial. Suppose Δ contains at least two 2-cells. We transform Δ into a diagram Δ_1 with cyclically reduced boundary label by performing folding surgeries and/or removing extremal edges. Suppose n is the number of extremal edges removed in this fashion. Then $|\partial\Delta_1| = |\partial\Delta| - 2n$ and Δ_1 contains n fewer undirected edges than Δ

By Lemma 6.0.19 we can find a 2-cell π in Δ_1 such that $\partial\pi = pq$, p is a subpath of $\partial\Delta_1$, and $|p| > \frac{1}{2}|\partial\pi|$. Suppose we delete π from Δ_1 and call the resulting diagram Δ_π. Note that $|\partial\Delta_\pi| = |\partial\Delta_1| - |p| + |q|$, and that Δ_π contains $|p|$ fewer undirected edges than Δ_1. By the induction hypothesis, Δ_π contains at most $d|\partial\Delta_\pi|$ undirected edges. Therefore Δ_1 contains $d|\partial\Delta_\pi| + |p|$ undirected edges. Therefore Δ contains at most $d|\partial\Delta_\pi| + |p| + n < d(|\partial\Delta_\pi| + |p| - |q| + 2n) = d|\partial\Delta|$ undirected edges.

To prove the second claim, suppose that Δ is a reduced spherical diagram over P. Let π be a 2-cell in Δ. Suppose we delete π from Δ and call the resulting diagram Δ_π. By Lemma 6.0.19 there is some 2-cell π_1 of Δ_π such that $\partial\pi_1$ shares a common subpath p with $\partial(\Delta_\pi)$ such that $|p| > \frac{1}{2}|\partial\pi|$. Since P satisfies $C'(\lambda)$ for $\lambda \leq \frac{1}{6}$, this implies that π and π_1 form a reducible pair of 2-cells in Δ, contradicting the assumption that Δ is reduced.

□

Chapter 7

The Group Presentation

The goal of this section is to generalize Theorem 1.0.3 to a statement about union machines.

The presentation $P(M)$ from Theorem 1.0.3 is constructed by adding generators and relators to the presentation $S(M')$. The generating set of $P(M)$ includes all generators of $S(M')$ as well as the new generators $\{\kappa_i | i = 1, ..., 2N\}$ for some sufficiently large N. We call these new generators κ-letters.

There are three types of relators of $P(M)$: transition relators, auxiliary relators, and the hub relator. The transition relators of $P(M)$ are exactly the transition relators of $S(M')$. The auxiliary relators of $S(M')$ consist of the auxiliary relators of $S(M')$ as well as the relators $\{\rho\kappa_i = \kappa_i\rho | i = 1, ..., 2N\}$ for each command letter ρ of $S(M')$.

For a word w in the generators of $S(M')$, let $K(w)$ denote the following word:

$$K(w) := (w^{-1}\kappa_1 w\kappa_2 w^{-1}\kappa_3 w\kappa_4 ... w^{-1}\kappa_{2N-1} w\kappa_{2N})(\kappa_{2N} w^{-1}\kappa_{2N-1} w ... \kappa_2 w^{-1}\kappa_1 w)^{-1}.$$

The hub relator is $K(W_0) = 1$. These generators and relators constitute the presentation $P(M)$.

For every admissible word W of $S(M')$, the word $K(W)$ is trivial in $P(M)$ if and only if W is accepted by $S(M')$. If $\mathbf{u} \in A^*$, and $\mathbf{c_u}$ is the input configuration for $\mathbf{u}$ in M', then we define

$$\mathcal{K}(\mathbf{u}) := K(\sigma(\mathbf{c_u})). \tag{7.0.1}$$

Lemma 7.0.21. *There is a linear time algorithm that, when given as input a description of a Turing machine M, outputs the group presentation $P(M)$.*

Proof. This follows directly from Lemma 3.0.6, Corollary 1 part 2, and the definition of $P(M)$.

□

Lemma 7.0.22. *Suppose $K(W)$ is trivial in $P(M)$, and suppose that C is the minimal area accepting computation for W in $S(M')$. Then there is a $P(M)$ diagram $\Pi(C)$ with boundary label $K(W)$ and area equal to $O(area(C))$. The constants witnessing this big O are uniform for all k-tape Turing machines M.*

Proof. This is stated on page 426-427 of [10]. The constants witnessing the big O notation depend only on the choice of N.

□

The diagram $\Pi(C)$ is called a *computational disc* for C.

Lemma 7.0.23. *Let M be a Turing machine, $\boldsymbol{u}$ be an accepted input of M, and $\boldsymbol{c_u}$ be the corresponding input configuration of M. Let C be the minimal area accepting computation for $\sigma(\boldsymbol{c_u})$ in $S(M')$. Then the minimal area $P(M)$ diagram Δ with boundary label $K(\sigma(\boldsymbol{c_u})) = \mathcal{K}(\boldsymbol{u})$ has area not less than $O(area(C))$.*

Proof. This is stated in the proof of Proposition 12.1[10].

□

We will now generalize Theorem 1.0.3 to union machines. For a union machine M_∞ that decides a language $L \subseteq A^*$, we use the Turing machines M_i to construct the presentations $P(M_i)$. We define the generating set B of $P(M_\infty)$ to be the union of the generating sets of the presentations $P(M_i)$. The relator set of $P(M_\infty)$ is defined to be the union of the relator sets of the presentations $P(M_i)$.

Note that this presentation $P(M_\infty)$ is identical to the presentation we would get by applying the standard construction of $P(M)$ to a union machine M_∞ (this would be possible because the construction of $P(M)$ does not depend at all on the finiteness of M).

For M_∞, we define the injective map $\mathcal{K}$ identically to how we define $\mathcal{K}$ for a standard Turing machine, see (7.0.1).

Lemma 7.0.24. *Let $L \subset A^*$ be a language accepted by an enumerable k-tape union machine M_∞. Suppose $T(n)$ is the time function of M_∞, where $T(n)^4$ is superadditive. Let $P(M_\infty)$ and $\mathcal{K}$ be as defined above.*

1. $\boldsymbol{u} \in L$ *if and only if* $\mathcal{K}(\boldsymbol{u}) = 1$ *in* $P(M_\infty)$*;*

2. $\mathcal{K}(\boldsymbol{u})$ *has length* $O(|\boldsymbol{u}|)$*. There is a linear-time algorithm that takes as input a word* $\boldsymbol{u}$ *in* A^* *and outputs* $\mathcal{K}(\boldsymbol{u})$*.*

3. *The presentation* $P(M_\infty)$ *is enumerable.*

4. $P(M_\infty)$ *has Denh function equivalent to* $T(n)^4$*.*

Proof. Suppose $\mathbf{u} \in A^*$. Then $\mathbf{u} \in L$ if and only if there is some M_i that accepts $\mathbf{u}$. By Theorem 1.0.3, M_i accepts $\mathbf{u}$ if and only if $\mathcal{K}(\mathbf{u})$ is trivial in $P(M_i)$. By definition, $\mathcal{K}(\mathbf{u})$ is trivial in some $P(M_i)$ if and only if $\mathcal{K}(\mathbf{u})$ is trivial in $P(M_\infty)$. Part 1 is proved.

Part 2 follows immediately from the definition of the map $\mathcal{K}$ in (7.0.1). Part 3 follows from Lemma 7.0.21 and the fact that M_∞ is enumerable.

To prove part 4, consider an arbitrary trivial word w in $P(M_\infty)$. There is some i such that the minimal area $P(M_\infty)$ diagram Δ for w is also a $P(M_i)$ diagram. Let $|\Delta|_2$ denote the area of Δ. Let T_i be the time function of M_i. By Theorem 1.0.3 the Dehn function of $P(M_i)$ is equivalent to T_i^4. Therefore there is a constant d such that $|\Delta|_2 \leq dT_i^4(d|w|)$, and this constant is uniform for all k-tape Turing machines. Since the input alphabet of M_∞ is finite, we can assume that i is sufficiently large that $T_i(y) = T(y)$ for all $y \leq d|w|$.This implies that $|\Delta|_2 \leq dT^4(d|w|) = dT_i^4(d|w|)$. If f is the Dehn function of $P(M_\infty)$, we conclude that $f \preceq T^4$.

In order to prove that $T^4 \preceq f$, let $\mathbf{u}$ be any accepted input for M'_∞ and let $\mathbf{c_u}$ be the corresponding input configuration. Let C be the minimal area accepting computation for $\sigma(\mathbf{c_u})$ in $S(M'_\infty)$. Suppose the minimal area $P(M_\infty)$ diagram with boundary label $\mathcal{K}(\mathbf{u})$ is Δ. We can find i sufficiently large that C is a $S(M'_i)$ computation and Δ is a $P(M_i)$ diagram. Note that since C is the minimal area accepting computation for $\sigma(\mathbf{c_u})$ in $S(M'_\infty)$, C is also the minimal area accepting computation for $\sigma(\mathbf{c_u})$ in $S(M'_i)$ as well. By Lemma 7.0.23, Δ has area not less than $O(\text{area}(C))$. Therefore the minimal area $P(M_\infty)$ diagram Δ with boundary

label $\mathcal{K}(\mathbf{u})$ has area not less than $O(area(C))$. Since $|\mathcal{K}(\mathbf{u})| = O(|\mathbf{u}|)$, we conclude that $f(n) \succeq T(n)^4$.

□

Chapter 8

Solvability of the Word Problem

The purpose of this section is to prove the following lemma.

Lemma 8.0.25. *If M_∞ is enumerable and if the language accepted by M_∞ is decidable then the word problem for $P(M_\infty)$ is solvable.*

Since $P(M_\infty)$ may be an infinite presentation, being able to compute the Dehn function for $P(M_\infty)$ is not sufficient to solve the word problem. Our proof will rely on results proven in [10] about the structure of $P(M)$ diagrams.

If a diagram over a presentation $P(M)$ or $P(M_\infty)$ contains no hub relators, then we say the diagram is hub-free.

Lemma 8.0.26. *If w is a word in $P(M_\infty)$, it is decidable whether there is a reduced hub-free $P(M_\infty)$ diagram with boundary label w.*

Proof. It is stated in Lemma 8.1[10] that if Δ is a reduced diagram without hubs over a presentation $P(M)$ for a Turing machine M, then there exists a constant c such that the area of Δ is at most $c|\partial(\Delta)|^3$. It is stated in the proof of Lemma 8.1[10] that the constant c is determined by the length of the longest relator in the presentation $S(M')$. By Lemma 4.0.16, there is an upper bound b_k on the length of the longest relator of $S(M)$ for any k-tape M. The constant c is therefore uniform for all k-tape Turing machines M. It follows that Lemma 8.1[10] holds for $P(M_\infty)$ as well: if Δ is a reduced diagram without hubs over $P(M_\infty)$, then the area of Δ is at most $c|\partial(\Delta)|^3$.

We complete the proof by showing that given w, it is possible to effectively obtain a finite subset $J(w)$ of the relators of $P(M_\infty)$ such that the minimal area hub free diagram for w is a $J(w)$ diagram. By Lemma 7.6[10] if Δ is reduced and contains no hubs, then Δ contains no $\hat{\Theta}$ annuli. Therefore every $\hat{\Theta}$ band in Δ has its start and end edge on the boundary of Δ. Therefore, we can effectively recover from w every $\hat{\Theta}$ letter ρ

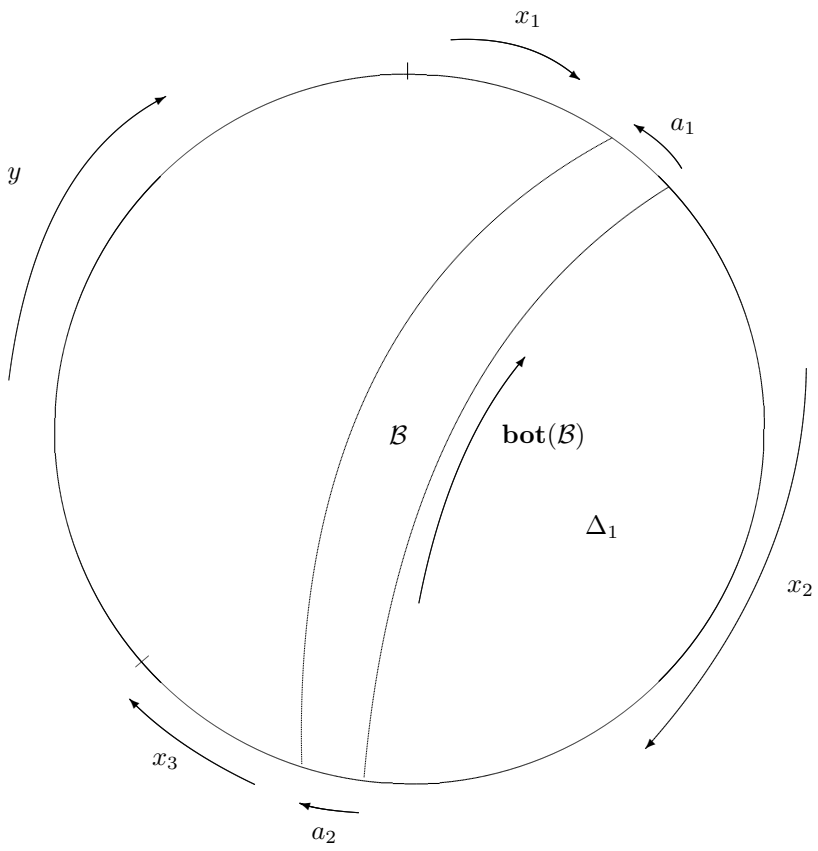
of $S(M'_\infty)$ that labels an edge in Δ. Recall that each $\hat{\Theta}$ letter ρ of $S(M'_\infty)$ is indexed by a command letter τ of M'_∞. By Lemma 4.0.11, given a command letter τ of M' we can effectively produce the finite set of all commands of $S(M'_\infty)$ (and therefore all transition relators of $P(M_\infty)$) whose command letters are indexed by τ. Therefore we can effectively produce a finite set of B of transition relators such that all transition relators that appear in Δ are elements of B.

As for auxiliary relators, each auxiliary relator is of the form $\rho x = x\rho$, where x is either a κ letter or a tape letter of $P(M_\infty)$. Minimal area hub free diagrams do not contain annuli composed of auxiliary relators, since the inner and outer boundary labels of such annuli are identical. Therefore every auxiliary relator in Δ is part of a κ band composed of auxiliary relators or an $\hat{A}$ band composed of auxiliary relators. In a hub free diagram such as Δ, κ bands start/end on the boundary of Δ, while $\hat{A}$ bands start/end either on the boundary of Δ or on the boundary of a transition cell. Therefore, using w and B we can produce the finite set of κ and $\hat{A}$ letters that label edges in Δ. Since we have already determined the finite set of $\hat{\Theta}$ edges that label edges in Δ, we can produce a finite set B' of auxiliary relators of $P(M_\infty)$ such that B' contains all relators appearing in Δ. This completes the proof.

□

Lemma 8.0.27. *Suppose that Δ is a $P(M_\infty)$ diagram with $\partial(\Delta) = xy$, where $Lab(x)$ is a subword of $K(W)$ for some admissible word W of $S(M'_\infty)$. Then Δ contains no $\hat{A}$ or κ band consisting entirely of auxiliary relators whose start and end edges are both contained in x.*

Proof. Suppose such a band $\mathcal{B}$ exists. Suppose that a_2 is the start edge of $\mathcal{B}$ and a_1^{-1} is the end edge of $\mathcal{B}$. Suppose $x = x_1 a_1^{-1} x_2 a_2 x_3$. We cut along $\mathbf{bot}(\mathcal{B})$ to separate Δ into two subdiagrams, Δ_1 and Δ_2. The diagram Δ_1 has boundary $\partial(\Delta_1) = \mathbf{bot}(\mathcal{B})x_2$.



We claim that for each component $\hat{Q}_i$ of $\hat{Q} = (\hat{Q}_1 \cup \hat{Q}_2 \cup ...)$, Δ_1 contains no $\hat{Q}_i$ edges. If Δ_1 did contain a $\hat{Q}_i$ edge then Δ_1 would contain a $\hat{Q}_i$ band $\mathcal{B}'$, since reduced hub-free $P(M_\infty)$ diagrams cannot contain $\hat{Q}$-annuli by Lemma 7.2[10]. Since $\mathbf{bot}(\mathcal{B})$ is composed entirely of $\hat{\Theta}$ edges, $\mathcal{B}'$ must begin and end on x_2. Assume without loss of generality that $\mathcal{B}'$ is an outermost $\hat{Q}_i$ band of Δ_1. Then $\mathcal{B}'$ must begin and end on consecutive $\hat{Q}$ letters of x_2. The word $\mathrm{Lab}(x_2)$ is a subword of $K(W)$, and consecutive $\hat{Q}$ letters in $K(W)$ come from different components of $\hat{Q}$. Therefore there is no $\hat{Q}_i$ such that $\mathcal{B}'$ is a $\hat{Q}_i$ band.

Since Δ_1 contains no $\hat{Q}$ edges, we conclude that Δ_1 is composed entirely of auxiliary relators. Since these relators are commutators of elements of $\hat{\Theta}$ and $\hat{A} \cup \kappa$, this means that boundary label of Δ_1 is trivial in the direct product of free groups $F(\hat{A} \cup \kappa) \times F(\hat{\Theta})$. Since $K(w)$ contains no $\hat{\Theta}$-edges, the path x_2 contains no $\hat{\Theta}$ edges. The path $\mathbf{bot}(\mathcal{B})$ is composed entirely of $\hat{\Theta}$ edges, which means that the label of $\mathbf{bot}(\mathcal{B})$ is not freely reduced. Therefore there are two adjacent 2-cells in $\mathcal{B}$ that form a reducible pair, since an auxiliary relator is uniquely determined by the $\hat{A} \cup \kappa$ letter and the $\hat{\Theta}$ letter that appear in its boundary label. This contradicts the assumption that Δ is reduced.

□

The following lemma is Lemma 7.1[10].

Lemma 8.0.28. *If Δ is a $P(M_\infty)$ diagram containing a κ-annulus then there exists another $P(M_\infty)$ diagram Δ_κ with identical boundary label and smaller area.*

Corollary 4. In the above lemma, Δ_κ does not contain more hubs than Δ.

Proof. Although this statement about the number of hubs is not explicitly stated in Lemma 7.1[10], it follows immediately from the proof of Lemma 7.1[10].

□

Proof of Lemma 8.0.25. In order to prove that it is decidable whether an arbitrary word w in $P(M_\infty)$ is trivial we use induction on the number of κ letters contained in w. If there are zero κ-letters in w then the minimal area diagram Δ with $\text{Lab}(\partial\Delta) = w$ can contain no hubs, since the boundary of the hub relator contains κ-letters and Δ contains no κ-annuli (by Lemma 8.0.28). Thus we conclude that if w contains no κ letters then w is trivial if and only if w has a hub free diagram (which is decidable by Lemma 8.0.26).

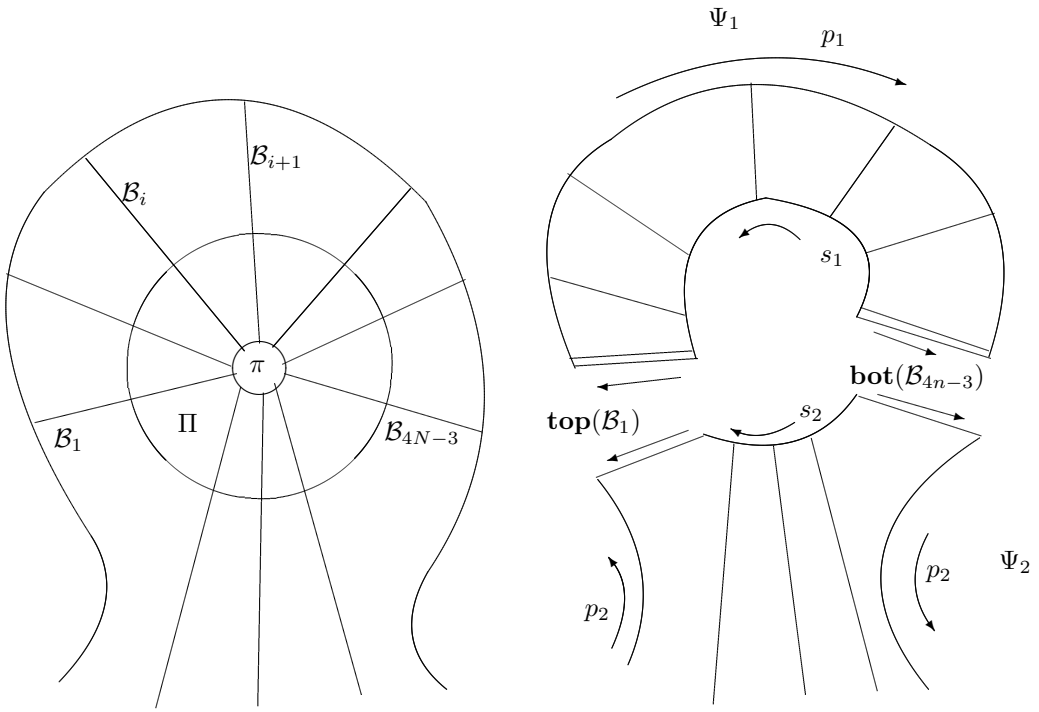
For the rest of this proof, we assume that w is a word in $P(M_\infty)$ and that Δ is a diagram with a minimal number of hubs. The results we will quote from [10] depend on this assumption. It should be noted that the results we will cite from [10] are about minimal hub diagrams over a $P(M)$ presentation for an arbitrary k-tape Turing machine M. Since every minimal hub diagram over a presentation $P(M_\infty)$ is also a minimal hub diagram over some $P(M_i)$ for sufficiently large i, the results we will cite hold for minimal hub $P(M_\infty)$ diagrams as well.

In [10] (on page 435), the authors define a special type of κ-band called a *dividing κ-band*. It is stated in [10] on page 435 that if Δ is a $P(M_\infty)$ diagram for w containing a dividing κ-band then there exists a pair of reduced diagrams (Δ', Ψ) fulfilling the following property. The diagrams Δ' and Ψ can be glued together to form a diagram with boundary label w such that two of the κ-edges in the boundary of this diagram are boundary edges of Ψ. Additionally, it is stated on page 435 of [10] that Ψ does not contain hubs, $|\partial\Psi| \leq 2|\partial\Delta|$, and every $\hat{\Theta}$-band in Ψ either begins or ends on $\partial\Delta$.

Since every relator in Ψ is contained in a $\hat{\Theta}$-band, if w is trivial in $P(M_\infty)$ then we can use the argument from the second paragraph of the proof of Lemma 8.0.26 to effectively construct from w a finite set J of

$P(M_\infty)$ relators such that every relator in Ψ is contained in J. Since Ψ is hub-free, it can have area at most $c|\partial\Psi| \leq c2|\partial\Delta|$ (as in the first paragraph of the proof of Lemma 8.0.26). We can therefore effectively produce from w a set B of diagrams which, if w is trivial, contains Ψ. We can then check if there is any way of multiplying a cyclic conjugate of w by a cyclic conjugate of the boundary label of an element of B to obtain a word w' with fewer κ-letters than w. If we can produce such a w' then by the induction hypothesis we can decide whether w' (and therefore w) is trivial. If not, then we conclude that if w is trivial then Δ contains no dividing κ-bands and we proceed.

Suppose we have determined that w has no hub-free diagram and that if w is trivial then a minimal hub diagram for w contains no dividing κ-bands. We can then use the following procedure to determine if w is trivial. It is proven in [10] that if Δ is a minimal-hub $P(M_\infty)$ diagram without dividing κ-bands and containing at least one hub then there is a triple of $P(M_\infty)$ diagrams (Ψ_1, Ψ_2, Π) associated with Δ satisfying the following properties.



- The diagram Π is a computational disk with hub π, and contains $4N$ κ-bands that end on the boundary of π, numbered $\mathcal{B}_1, ..., \mathcal{B}_{4N}$ as in the above figure.

- The diagram Ψ_1 contains $\mathcal{B}_1, ..., \mathcal{B}_{4N-3}$. The diagram Ψ_2 contains $\mathcal{B}_{4N-2}, \mathcal{B}_{4N-1}, \mathcal{B}_{4N}$.

- $\partial(\Delta) = p_1 p_2$

- The end edges of the κ bands $\mathcal{B}_1, ..., \mathcal{B}_{4N-3}$ are contained in the path p_1.

- $\partial\Psi_1 = \mathbf{top}(\mathcal{B}_1)p_1\mathbf{bot}(\mathcal{B}_{4N-3})^{-1}s_1$

- $\partial\Psi_2 = \mathbf{top}(\mathcal{B}_1)^{-1}s_2^{-1}\mathbf{bot}(\mathcal{B}_{4N-3})p_2$

- $\partial(\Pi) = s_1^{-1}s_2$.

These above properties are listed on page 438 of [10]. Additionally, the triple (Ψ_1, Ψ_2, Π) satisfies the following properties, listed on pages 439-440 of [10].

- (P1) By gluing Ψ_1, Ψ_2, and Π together one can get a diagram with the same boundary label as Δ.

- (P2) Ψ_1 is reduced and contains no hubs.

- (P3) There are no $\hat{\Theta}$ bands in Ψ_1 which cross all the bands $\mathcal{B}_1, ..., \mathcal{B}_{4N-3}$.

Since every diagram in $P(M_\infty)$ is also a diagram in some $P(M_i)$, we conclude that every $P(M_\infty)$ diagram Δ with at least one hub and no dividing κ-bands has an associated triple (Ψ_1, Ψ_2, Π) with the above properties.

If w is trivial (and, as assumed, if minimal hub diagrams for w contain no dividing κ-bands), then there exists a minimal hub $P(M_\infty)$ diagram Δ for w with associated triple (Ψ_1, Ψ_2, Π). We will show how to effectively construct a set of $P(M_\infty)$ diagrams such that if Δ and it's associated triple (Ψ_1, Ψ_2, Π) exist, then $\Psi_1 \in \Delta$.

We first show that it is possible to effectively produce a finite set $J(w)$ of trivial words in $P(M_\infty)$ such that if w is trivial then $\mathrm{Lab}(\partial\Psi_1) \in J(w)$. We begin by producing a finite set $B(w)$ of transition relators such that if w is trivial then all transition relators of Ψ_1 are in $B(w)$. Note that any $\hat{\Theta}$ band in Ψ_1 that begins/ends on $\mathbf{top}(\mathcal{B}_1)$ cannot end/begin on $\mathbf{bot}(\mathcal{B}_{4N-3})$ without violating property P3. Since s_1 contains no $\hat{\Theta}$ edges, no $\hat{\Theta}$ band can begin or end on s_1. Therefore every $\hat{\Theta}$ band in Ψ_1 must either begin or end on p_1. We can now use the argument from the second paragraph of the proof of Lemma 8.0.26 to produce $B(w)$.

Next we produce a finite set $B'(w)$ of auxiliary relators such that if w is trivial then all auxiliary relators of Ψ_1 are in $B'(w)$. Note that every $\hat{A}$ edge contained in the boundary of an auxiliary relator of Ψ_1 is part of an $\hat{A}$ band composed of auxiliary relators. All such $\hat{A}$ bands start/end either on $\partial(\Psi_1)$ or on the boundary of a transition relator of Ψ_1. No such bands can start or end on $\mathbf{top}(\mathcal{B}_1)$ or $\mathbf{bot}(\mathcal{B}_{4N-3})$, since these paths are composed of $\hat{\Theta}$ letters. By Lemma 8.0.27, no such band can both begin and end on s_1. So each such $\hat{A}$ band that both starts and ends on $\partial\Psi_1$ must either start or end on p_1 which is a subpath of $\partial\Delta$. We can now construct $B'(w)$ as follows. Suppose ρ is a $\hat{\Theta}$ letter that appears in $\partial\Delta$ or in the boundary label of a transition relator of $B(w)$. Suppose z is either a κ-letter or an $\hat{A}$ letter that appears either in w or in the boundary label of an element of $B(w)$. We then include the auxiliary relator $z\rho = \rho z$ in $B'(w)$. Since either the start or end edge of every $\hat{A}$ band in Ψ_1 must appear either in $\partial\Delta$ or in the boundary of a transition relator of Ψ_1, and since every $\hat{\Theta}$ edge of Ψ_1 must appear either in the boundary of a transition relator of Ψ_1 or in $\partial\Delta$, the set $B'(w)$ must include every auxiliary relator that appears in Ψ_1.

It follows from pages 454-455 of [10] there is some constant ϵ_1' such that $|\partial\Psi_1| \leq \epsilon_1'|\partial\Delta| = \epsilon_1'|w|$. Therefore, given w, we can effectively produce a finite set of words $J(w)$ that contains the boundary label of every possible $B(w) \cup B'(w)$ diagram with boundary length not exceeding $\epsilon_1'|w|$. If w is trivial, then the diagram Ψ_1 will be in $J(w)$, as will the mirror image of Ψ_1.

We now multiply every cyclic conjugate of w by every cyclic conjugate of the boundary label of each element of $J(w)$. We collect the finite set of words that result from these multiplications and call this set $\hat{J}(w)$.

If w is trivial, then boundary label of $\Psi_2 \cup \Pi$ is an element of $\hat{J}(w)$. Call this boundary label w'. If w is trivial then the word w' will have s_1 as a subword. Recall that s_1 is a subword of $K(W)^{\pm 1}$ that contains at least $(4N-3)$ κ-letters, where W is some acceptable admissible word of $S(M_\infty')$.

Since $S(M'_\infty)$ has solvable configuration problem, we can effectively check whether there exist any words v in the set $\hat{J}(w)$ that contain "large subwords" of $K(W)$ for any accepted admissible word W. Specifically, we search for a word $v \in \hat{J}(w)$ containing a subword $s_1(v)$ such that $s_1(v)$ contains $(4N-3)$ many κ-letters and $s_1(v)$ is a subword of $K(W)^{\pm 1}$ for some acceptable admissible word W of $S(M'_\infty)$. If there are no such words v in $\hat{J}(w)$ we conclude that w is not trivial in $P(M_\infty)$. In this case we output"no". If there are such words v in $J'(w)$ then for each such pair $(v, s_1(v))$, if $K(W)^{\pm 1} = s_1(v)s_2(v)$, we will replace the subword $s_1(v)$ of v by $s_2(v)^{-1}$ and add the resulting word to $\hat{J}(w)$. Provided w is trivial, the boundary label of Ψ_2 (which contains fewer κ-letters than w) will now be an element of $\hat{J}(w)$. By the induction hypothesis, we can decide whether any of the elements of $\hat{J}(w)$ containing fewer κ-letters than w are trivial. If any of these words are trivial, we conclude that w is trivial. If not, we conclude that w is not trivial.

□

Lemma 8.0.29. *The presentation $P(M_\infty)$ is minimal.*

Proof. Consider any non-hub relator $\rho w = w\rho$ of $P(M_\infty)$. Suppose we remove this relator from $P(M'_\infty)$ and then attempt to construct a diagram Δ with boundary label $\rho w \rho^{-1} w^{-1}$ using the remaining relators. If Δ contains at least one hub, then there are at least $(4N-3)$ κ-edges in $\partial\Delta$, by Lemma 8.0.28 and the properties of the decomposition (Ψ_1, Ψ_2, Π). Since no non-hub relator of $P(M_\infty)$ contains more than two κ letters, we conclude that Δ contains no hubs. By Lemma 7.6[10], Δ contains no $\hat{\Theta}$ annuli. Therefore Δ contains only a single $\hat{\Theta}$-band, and it is a ρ-band. For each $\hat{Q}_i$ in the hardware of $S(M'_\infty)$, there is only a single relator in $P(M_\infty)$ that contains both the letter ρ and a $\hat{Q}_i$-letter. Therefore, if w contains any $\hat{Q}$-letters, Δ can not exist. If w contains no $\hat{Q}$-letters then $\rho w = w\rho$ is an auxiliary relator and w consists of a single $\hat{A}$ or κ letter. By Lemma 7.2[10] Δ can contains no Q-annuli, so Δ contains no transition relators. Therefore Δ is composed of auxiliary relators. By Lemma 7.9[10], Δ cannot contain $\hat{A}$ or κ annuli (note that in [10], $\hat{A} \cup \kappa$ is denoted by $\bar{Y}$). Therefore Δ cannot contain any auxiliary relators besides $\rho w = w\rho$. Therefore Δ doesn't exist. We conclude that if we remove any non-hub relator from $P(M_\infty)$ then the group presented by this new presentation is not isomorphic to the group presented by $P(M_\infty)$.

If we remove the hub relator from $P(M_\infty)$ then the only diagrams we can make using the remaing relators contain no hubs. Such diagrams contain no $\hat{\Theta}$ annuli by Lemma 7.6[10]. Therefore it is impossible to construct a diagram with a reduced boundary label using only non-hub relators whose boundary contains no $\hat{\Theta}$ edges. Since the boundary label of the hub relator contains no $\hat{\Theta}$ edges, we conclude that if we remove the hub relator from $P(M_\infty)$ then the group presented by this new presentation is not isomorphic to the

group presented by $P(M_\infty)$.

□

Chapter 9

Proof of Theorem 1.0.4

In this section, we will always assume that M_∞ is an enumerable union machine. By Lemma 7.0.24 $P(M_\infty)$ is enumerable. Therefore we can effectively construct sequences E_1 and E_2 such that E_1 is a sequence of positive generators of $P(M_\infty)$, E_2 is a sequence of relators of $P(M_\infty)$, every positive generator of $P(M_\infty)$ appears exactly once in E_1, and every relator of $P(M_\infty)$ appears exactly once in E_2.

In this section when we refer to an arbitrary positive generator of $P(M_\infty)$ as g_n, we mean to indicate that g_n is the nth term to appear in the sequence E_1. We define $X := \{g_n | n \in \mathbf{N}\}$.

We denote an arbitrary presentation with generating set X by

$$P = \langle X \| R \rangle.$$

Let $\{b, a_1, ..., a_{100}\}$ be a set of generating symbols that do not appear in X. We will define a map h from $F(X)$ to $F(b, a_1, a_2, ..., a_{100})$. The map h will be given by

$$h(g_n) = a_1 b^{2n} a_2 b^{2n} a_3 b^{2n} ... a_{99} b^{2n} a_{100}. \tag{9.0.1}$$

We now define a new presentation $P' = \langle b, a_1, a_2, ..., a_{100} \| R_h \rangle$ where the relator set R_h of P' is the set of cyclically reduced h-images of relators of P. Such a presentation P' is called an *h-presentation.* In the special case where $P = P(M_\infty)$, we denote P' by $P'(M_\infty)$.

Lemma 9.0.30. *For an enumerable union machine M_∞, the presentation $P'(M_\infty)$ is decidable.*

Proof. Since the generating set of $P'(M_\infty)$ is finite, it will be sufficient to show that the relator set of $P'(M_\infty)$ is decidable. Suppose we are given a word w in the generators of $P'(M_\infty)$. We must show that it is possible

to decide whether w is a relator of $P'(M_\infty)$. There is only a single hub relator in $P(M_\infty)$, so we can decide immediately whether w is the cyclically reduced h-image of the hub relator. We must show that it is also possible to decide whether w is the cyclically reduced h-image of a transition or auxiliary relator of $P(M_\infty)$.

Suppose g_i, g_j are distinct positive generators of $P(M_\infty)$. Note that when we cyclically reduce the word $h(g_i)h(g_j)^{-1}$, none of the $a_2, ..., a_{99}$ letters appearing in either $h(g_i)^{\pm 1}$ or $h(g_j)^{\pm 1}$ is canceled. This means we can decide whether an arbitrary element $w \in F(b, a_1, ...a_{100})$ is equal in $F(b, a_1, ..., a_{100})$ to the cyclically reduced h-image of an element in $F(X)$. If w is not such an element then w is not a $P'(M_\infty)$ relator. If w is the cyclically reduced h-image of an element of $F(X)$, then by examining the powers of b that appear in w we can recover the tuple of indices $i_1, \ldots, i_n$ such that $g_{i_1} \ldots g_{i_n}$ are the generators of $F(X)$ that appear in $h^{-1}(w)$. We can then use the aforementioned effectively constructible sequence E_1 of generators of P to recover the letters $g_{i_1}, \ldots, g_{i_n}$.

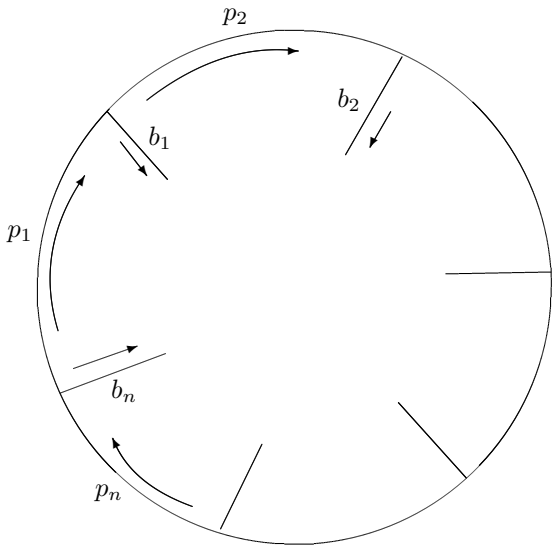
Recall that the command symbols of $P(M_\infty)$ contain a significant amount of information; in particular, there is an algorithm that takes as input a command symbol ρ of $P(M_\infty)$ and outputs the finite set of transition relators of $P(M_\infty)$ in which the letter ρ appears (as in the second paragraph of the proof of Lemma 8.0.26). We can use this algorithm to effectively determine from $g_{i_1}...g_{i_n}$ whether or not $h^{-1}(w)$ is a transition relator of $P(M_\infty)$. If w is not a transition relator of $P'(M_\infty)$, we continue.

If $h^{-1}(w)$ is a commutator of a command letter with a κ or $\hat{A}$ letter, then $h^{-1}(w)$ is an auxiliary relator of $P(M_\infty)$ and w is a relator of $P'(M_\infty)$. If not, and if $h^{-1}(w)$ is neither a transition nor hub relator of $P(M_\infty)$, then we conclude that w is not a relator of $P(M_\infty)$.

□

In order to analyze P' diagrams, we will consider 2-complexes that are similar to P' diagrams but are not P' diagrams. We call these 2-complexes *special P' 2-complexes*, and formally define them below.

Suppose that Δ' is a P' diagram and π' is a 2-cell of Δ'. Then there is a relator $r = g_{i_1}^{\epsilon_{i_1}} \ldots g_{i_n}^{\epsilon_{i_1}}$ of P such that boundary label of π' is the word obtained by cyclically reducing $h(r) = h(g_{i_1})^{\epsilon_{i_1}} \ldots h(g_{i_n})^{\epsilon_{i_n}}$. We draw paths $b_1, \ldots, b_n$ in the interior of π' as in the below figure.



The paths $b_1, \ldots, b_n$ are drawn and labeled such that $\text{Lab}(b_{(j-1)}^{-1} p_j b_j) = h(g_{i_j})^{\epsilon_{i_j}}$ for $j = 2, \ldots, n$ and $\text{Lab}(b_n^{-1} p_1 b_1) = h(g_{i_1})^{\epsilon_{i_1}}$. Thus the paths $b_1, \ldots, b_n$ can be thought of as geometric representations of the letters that are canceled when $h(r)$ is cyclically reduced.

Suppose we draw such paths in the interior of every 2-cell of an arbitrary P' diagram (or spherical P' diagram). We call the resulting 2-complex an *adjusted P' 2-complex* (or a *spherical adjusted P' 2-complex*). Note that the folding surgeries described in section 6 can be applied to (spherical) adjusted P' 2-complexes as well as diagrams. We define a *special P' 2-complex* to be either an adjusted P' 2-complex or any 2-complex obtained by applying folding surgeries to an adjusted P' 2-complex. Similarly, a *spherical special P' 2-complex* is either a spherical adjusted P' 2-complex or any 2-complex obtained by applying folding surgeries to a spherical adjusted P' 2-complex.

Suppose Δ' is a special P' 2-complex and π' is a 2-cell of Δ'. The *boundary* of Δ' is defined to be the clockwise oriented cyclic path of edges of Δ' that are adjacent to the complement of Δ' in the plane. The boundary of π' is defined identically. The boundary label of a special P' 2-complex or a 2-cell of a special P' 2-complex is the label of the boundary of that 2-complex or 2-cell. We define the *inner boundary* of π' to be the clockwise oriented cyclic path of edges adjacent to the interior of π'. For example, if π' is the 2-cell pictured in the above figure, the inner boundary of π' is the path $p_1 b_1 b_1^{-1} \ldots p_n b_n b_n^{-1}$, while the boundary of π' is the path $p_1 p_2 \ldots p_n$. The *inner boundary label* of a 2-cell π' of a special P' 2-complex is the label

of the inner boundary of π'. The area of a special P' 2-complex is the number of 2-cells it contains. This completes the definition of a special 2-complex.

A special P' 2-complex Δ' is a *minimal area* special P' 2-complex if there does not exist a special P' 2-complex with the same boundary label as Δ' and smaller area than Δ'.

We now define yet another class of 2-complexes that are similar to P' diagrams. Suppose that Δ is an arbitrary P diagram. For every edge e in Δ with label g_i, we replace e with a path p such that $\text{Lab}(p) = h(g_i)$. We call the resulting planar 2-complex $h(\Delta)$. Note that $h(\Delta)$ is not necessarily a $P'(M_\infty)$ diagram. This is because the boundary labels of the 2-cells of $h(\Delta)$ are not necessarily cyclically reduced, and are therefore not equal to relators of $P'(M_\infty)$. Instead, the boundary labels of the 2-cells of $h(\Delta)$ are non-reduced concatenations of words of the form $h(g_i)^{\pm 1}$.

If Δ is a P diagram, we call $h(\Delta)$ a *proper P' 2-complex*. The boundary of a proper P' 2-complex is defined identically as that of a special P' 2-complex. In order to prove Theorem 1.0.4, we will analyze an arbitrary P' diagram Δ' by first drawing paths in the interiors of its 2-cells to create a special P' 2-complex. We then show how to "decompose" this special P' 2-complex into subcomplexes that are proper P' 2-complexes. We will then use these proper P' 2-complexes to prove results about the original P' diagram Δ'.

Definition 3. Suppose Δ' is an arbitrary special P' 2-complex. If a path p is a subpath of the inner boundary of a 2-cell π' of Δ' and $\text{Lab}(p) = h(g_n)^{\pm 1}$ for $g_n \in X$, then we call p an *h-path* of Δ'.

Suppose Δ' is a special P' 2-complex containing distinct h-paths p_1 and p_2. If p_1 and p_2^{-1} share a common edge or a common vertex that is not an endpoint of both p_1 and p_2, then p_1 and p_2 are *adjacent* h-paths. If p_1 and p_2^{-1} share a common edge, then p_1 and p_2 are *edge adjacent* h-paths. Suppose p_1 and p_2 are adjacent h-paths where q is a common subpath of p_1 and p_2^{-1} such that $|q| > 0$, $p_1 = u_1 q x_1$ and $p_2^{-1} = u_2 q x_2$. If $\text{Lab}(u_1) = \text{Lab}(u_2)$ and $\text{Lab}(x_1) = \text{Lab}(x_2)$, then we call p_1 and p_2 *strongly adjacent* h-paths. If $p_1 = p_2^{-1}$, then p_1 and p_2 are *contiguous* h-paths.

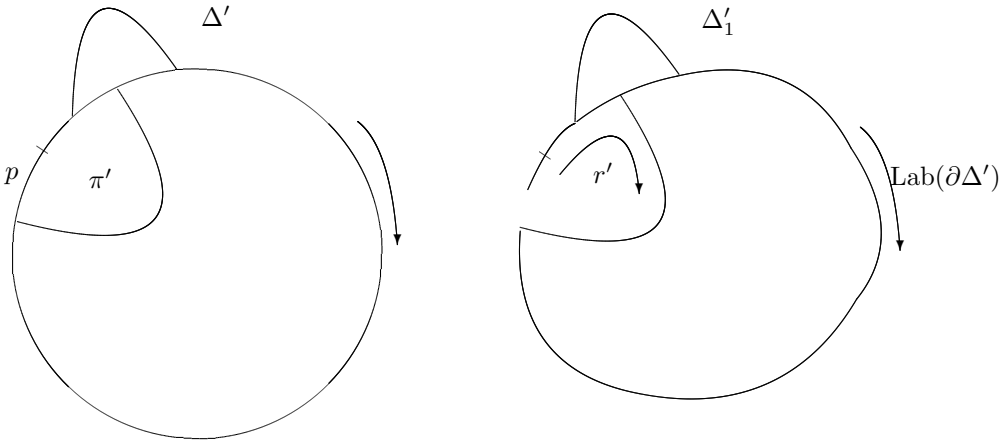
If w is a word in the generators of P', we say that w is *proper* if there exists a word $g_{i_1}^{\epsilon_{i_1}} \dots g_{i_n}^{\epsilon_{i_n}}$ in the generators of P such that w is the non-reduced product $h(g_{i_1})^{\epsilon_{i_1}} \dots h(g_{i_n})^{\epsilon_{i_n}}$.

Note that the inner boundary label of any 2-cell in a special P' diagram is a proper word.

Lemma 9.0.31. *Suppose Δ' is a special P' 2-complex containing no pairs of non-contiguous edge adjacent h-paths. Suppose also that Δ' has proper boundary label. Then there exists a proper P' 2-complex with the same area and boundary label as Δ'.*

Proof. We prove this lemma by induction on the area of Δ'. Suppose Δ' has zero area. Then $\mathrm{Lab}(\partial\Delta')$ is trivial in the free group $F(b, a_1, ..., a_{100})$. Since Δ' has proper boundary label, $\mathrm{Lab}(\partial\Delta') = h(g_{i_1})^{\epsilon_{i_1}} \dots h(g_{i_n})^{\epsilon_{i_n}}$. Note that the set of words in $F(b, a_1, ..., a_{100})$ of the form $h(g_j)$ is Neilson reduced. Therefore the map from $F(g_1, g_2, ...)$ to $F(h(g_1), h(g_2), ...) \subseteq F(b, a_1, ..., a_{100})$ is an isomorphism of free groups. Thus the word $g_{i_1}^{\epsilon_{i_1}} \dots g_{i_n}^{\epsilon_{i_n}}$ is trivial in $F(b, a_1, ..., a_{100})$, and there exists a P diagram Δ with zero area and boundary label $g_{i_1}^{\epsilon_{i_1}} \dots g_{i_n}^{\epsilon_{i_n}}$. The proper P' 2-complex $h(\Delta)$ has the same area and boundary label as Δ'.

Suppose Δ' has area greater than zero. Let π' be a 2-cell of Δ' that shares a boundary edge e with Δ'. Since $e \in \partial\pi'$, the edge e is contained in an h-path p of Δ'. Since p is not edge adjacent to any other h-paths, every edge of p is a boundary edge of Δ'. We remove the interior of π' from Δ', creating a hole in Δ'. We then cut the resulting annular 2-complex at the initial vertex of p, which creates a simply connected special P' 2-complex (call it Δ'_1) whose area is one less than that of the original Δ'.



Suppose that r' was the inner boundary label of π'. Then the boundary label of Δ'_1 is the non-reduced product $\mathrm{Lab}(\partial\Delta')(r')^{-1}$ (here we assume that the boundary label of Δ'_1 and the inverse of the inner boundary

label of π' are both read beginning at the initial vertex of p). Recall that the inner boundary label of any 2-cell in a special diagram is a proper word. Thus, the boundary label of Δ_1' is the non-reduced product of two proper words and is therefore proper. By the induction hypothesis, there exists a proper P' 2-complex Δ_2' with the same boundary label and area as Δ_1'. The P diagram $h^{-1}(\Delta_2')$ has boundary label

$$h^{-1}(\text{Lab}(\partial\Delta')(r')^{-1}) = h^{-1}(\text{Lab}(\partial\Delta'))h^{-1}((r')^{-1}).$$

Note that $h^{-1}(\Delta_2')$ has the same area as Δ_2' (that is, one less that the area of Δ'). Consider the P relator $r = h^{-1}(r')$. Note that the boundary label of $h^{-1}(\Delta_2')$ contains a subpath with label $(h^{-1}(r'))^{-1}$. Thus a 2-cell π with boundary label $r = h^{-1}(r')$ can be glued to $h^{-1}(\Delta_2')$ to form a single P diagram Δ such that $h(\Delta)$ has the same boundary label and area as Δ'. This completes the proof.

□

Lemma 9.0.32. *Suppose Δ' is a special P' 2-complex such for every edge $e \in \Delta'$, either e or e^{-1} is contained in an h-path of Δ'. Suppose also that there are no pairs of non-contiguous adjacent h-paths in Δ'. Then there exists a proper P' 2-complex with the same boundary label and area as Δ'.*

Proof. Suppose e is a boundary edge of Δ'. The edge e is contained in an h-path p. Since Δ' contains no pairs of non-contiguous adjacent h-paths, every edge of p is contained in the boundary of Δ', and no non-endpoint vertex of p is contained in any distinct h-path of Δ'. Since, for every edge $e \in \Delta'$, either e or e^{-1} is contained in an h-path of Δ', it follows that every vertex of Δ' is also contained in an h-path of Δ'. We conclude that p is a subpath of the boundary of Δ'. Therefore the boundary of Δ' is composed of whole and disjoint h-paths and/or inverses of h-paths, and Δ' has proper boundary label. The lemma now follows from Lemma 9.0.31.

□

Lemma 9.0.33. *Suppose Δ' is a minimal area P' diagram. Then there exists a special P' 2-complex with the same boundary label and area as Δ' that contains no pairs of non-contiguous strongly adjacent h-paths.*

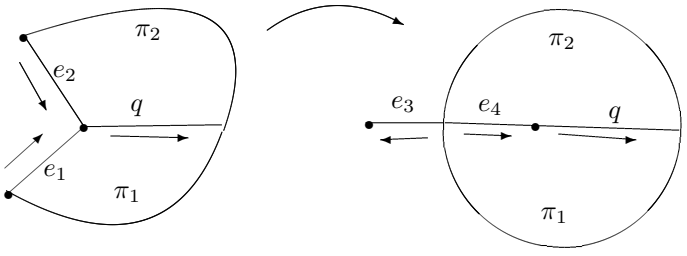
Proof. We first transform the P' diagram Δ' into a special P' 2-complex by drawing labeled paths in the interiors of the 2-cells of Δ'. For convenience, we will continue to call this special P' 2-complex Δ'. Suppose that p_1, p_2 are strongly adjacent h-paths in Δ'. Suppose $p_1 = \mu_1 q \beta_1$ and $p_2^{-1} = \mu_2 q \beta_2$, where q is a maximal common subpath of p_1 and p_2^{-1} such that $\text{Lab}(\mu_1) = \text{Lab}(\mu_2)$ and $\text{Lab}(\beta_1) = \text{Lab}(\beta_2)$.

We will identify p_1 and p_2^{-1} edge by edge by performing folding surgeries. We start by performing a folding surgery to identify the final edges of μ_1 and μ_2. We denote these edges by e_1 and e_2 respectively.

Since $\text{Lab}(\mu_1) = \text{Lab}(\mu_2)$, we know that e_1 and e_2 have the same label. Since μ_1 and μ_2 share the same final vertex, e_1 and e_2 share the same final vertex. In order to perform the folding surgery, the initial vertex of e_1 must be distinct from both endpoints of e_2 and the initial vertex of e_2 must be distinct from both endpoints of e_1.

Suppose toward a contradiction that the initial vertex of e_1 is the same as the final vertex of e_2. Then the initial and final vertices of e_1 are identical. Therefore e_1 is the boundary of a special P' 2-complex contained in Δ'. But this is impossible since the boundary of every special P' 2-complex has even length (this is because the boundary label of every 2-cell of a special P' 2-complex can be cyclically reduced to a relator of P', and every relator of P' has even length). Therefore the initial vertex of e_1 is distinct from the final vertex of e_2. A symmetric argument proves that the initial vertex of e_2 is distinct from the final vertex of e_1.

If e_1 and e_2 share the same initial vertex then $e_1^{-1}e_2$ is the boundary path of a special P' sub-2-complex of Δ' with area ≥ 1 and boundary label xx^{-1} for some generator $x \in \{b, a_1, ..., a_{100}\}$. In this case, we can excise this sub-2-complex of Δ' and identify the edges e_1 and e_2, which will produce a special P' 2-complex with boundary label equal to that of Δ' and with less area than Δ'. By then applying folding surgeries to this special P' 2-complex, we can produce an adjusted P' 2-complex. By removing the paths from the interiors of the 2-cells of this adjusted P' 2-complex, we can then produce a P' diagram with the same boundary label and less area than the original P' diagram Δ'. Since this contradicts the assumption that the original diagram Δ' is a minimal area P' diagram, we conclude that e_1 and e_2 do not share the same initial vertex. We can now perform a folding surgery at the path $e_1e_2^{-1}$ to increase by one the number of edges shared by p_1 and p_2^{-1}, as shown in the below figure.



We repeat this process until p_1 and p_2^{-1} have been identified. We claim that this process of identifying p_1 and p_2^{-1} decreases the number of h-paths in Δ' that are not contained in a contiguous pair of h-paths. To prove this claim, note that the folding surgery pictured above can only affect the edge adjacency of at most two h-paths of Δ' besides p_1 and p_2. We call these h-paths p_3 and p_4. Before the folding surgery, the paths p_3^{-1} and p_4 shared the edges e_1 and e_2 with p_1 and p_2^{-1}, respectively (note that p_3, p_4 do not exist if e_1, e_2, respectively, are boundary edges). Before the folding surgery, p_3 and p_4 were not contiguous to p_1 or p_2 since p_1 and p_2 were strongly adjacent to each other. Therefore the folding surgery pictured above does not increase the number of h-paths of Δ' that are not contained in a contiguous pair of h-paths. Thus when the process of identifying p_1 and p_2^{-1} is complete, the number of h-paths of Δ' that are not contained in a contiguous pair has decreased by at least two.

Since there are only finitely many h-paths in Δ', this process of transforming pairs of strongly adjacent h-paths into pairs of contiguous h-paths must terminate after some finite number of identifications. At this point Δ' will contain no more pairs of strongly adjacent h-paths.

□

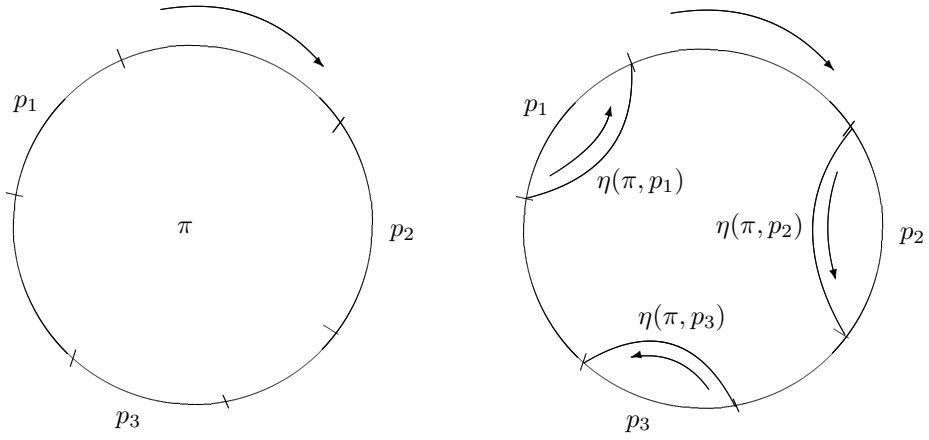
Suppose Δ' is a minimal area special P' 2-complex with proper boundary label. Suppose that $\hat{\Delta}'$ is the spherical 2-complex obtained by attaching a single new 2-cell π_0 to the boundary of Δ'. Note that since $\mathrm{Lab}(\partial\pi_0) = (\mathrm{Lab}(\partial\Delta')^{-1})$, the inner boundary label of π_0 is a proper word. Therefore the 2-complex $\hat{\Delta}'$ is a spherical special $\hat{P}'$ 2-complex, where $\hat{P}'$ is the h-presentation obtained by cyclically reducing $\mathrm{Lab}(\partial\Delta')$ and adding the resulting word to the relator set of P'. The proof of the following corollary is identical to that of Lemma 9.0.33.

Corollary 5. The spherical special $\hat{P}'$ 2-complex $\hat{\Delta}$ can be transformed via a sequence of folding surgeries into a spherical special $\hat{P}'$ 2-complex that contains no pairs of non-contiguous strongly adjacent h-paths.

Suppose Δ' is a minimal area special P' 2-complex that contains no pairs of strongly adjacent h-paths. We define an equivalence relation $\equiv_e$ on the set of h-paths of Δ'. If p_1 and p_n are h-paths in Δ', then $p_1 \equiv_e p_n$ if there exists a sequence of h-paths $p_1, p_2, ..., p_n$ such that for $i = 1, \ldots, (n-1)$, the paths p_i and p_{i+1} are edge adjacent. For an h-path p, denote the $\equiv_e$ equivalence class of p by $[p]$.

Suppose there is an h-path $p \in \Delta'$ such that p is edge adjacent but not contiguous to at least one other h-path of Δ'. Suppose π is a 2-cell of Δ', and $\{p_1, ..., p_n\}$ is the set of h-paths in the inner boundary of π that

are also contained in $[p]$. We can transform π into $(n+1)$ new 2-cells by placing new directed edges, labeled $\eta(\pi, p_1), ..., \eta(\pi, p_n)$, in the interior of π. We call these new edges *η-edges*. The initial and final vertices of the η-edge labeled $\eta(\pi, p_i)$ are the initial and final vertices of p_i, respectively. The η-edges are placed such that $\eta(\pi, p_i)$ and $\eta(\pi, p_j)$ do not intersect for $i \neq j$. All but one of the new 2-cells have boundary label of the form $\text{Lab}(p_i)\eta(\pi, p_i)^{-1}$. We call these new 2-cells *$[p]$-cells*. The single new 2-cell whose boundary contains each of the η-edges $\eta(\pi, p_1), ..., \eta(\pi, p_n)$ is called an *η-cell*.



We perform this operation on every 2-cell of Δ'. As a result, for each 2-cell $\pi' \subseteq \Delta'$, if there are n distinct h-paths $p_1, ..., p_n \in [p]$ contained in the inner boundary of π', then π' is transformed into $(n+1)$ new 2-cells, n of which are $[p]$-cells. We denote the minimal sub-complex of Δ' containing the $[p]$-cells of Δ' by $\Delta'([p])$.

The 2-complex $\Delta'([p])$ is planar. By the definition of the $\equiv_e$ relation and the fact that $\Delta'([p])$ is composed of $[p]$-cells, $\Delta'([p])$ is connected. Every edge of $\Delta'([p])$ is either an η-edge or is contained in an h-path of $[p]$. However $\Delta'([p])$ is not necessarily simply connected. The *boundary edges* of $\Delta'([p])$ are those edges of $\Delta'([p])$ that are adjacent to the complement of $\Delta'([p])$ in the plane. The *boundary* of $\Delta'([p])$, which we denote $\partial\Delta'([p])$, is the union of the clockwise oriented cyclic paths of boundary edges of $\Delta'([p])$.

Lemma 9.0.34. *If e is an edge of $\partial\Delta'([p])$ and e is not an η-edge, then e is contained in $\partial\Delta'$. Every η-edge in $\Delta'([p])$ is contained in $\partial\Delta'([p])$.*

Proof. Every edge in $\Delta'([p])$ is either an η-edge or is contained in some h-path of $[p]$. If p_1 is an h-path of $[p]$ then every edge e of p_1 is either a boundary edge of Δ' or there exists an h-path $p_2 \in [p]$ such that e

is contained in p_1 and p_2^{-1}. If e is contained in both p_1 and p_2^{-1} then e is contained in the contours of two distinct $[p]$-cells of $\Delta'([p])$ and is therefore an interior edge of $\Delta'([p])$. Thus if e is an edge of $\partial\Delta'([p])$ and e is not an η-edge, then $e \in \partial\Delta'$.

When we draw the η-edges inside a 2-cell π' of Δ', as described above, we transform π' into several new 2-cells. All but one of these new 2-cells are $[p]$-cells. Each η-edge drawn inside π' is contained in the contour of both a $[p]$-cell and an η-cell. Since $\Delta'([p])$ contains only $[p]$-cells and no η-cells, we conclude that every η-edge of $\Delta'([p])$ is a boundary edge of $\Delta'([p])$.

□

We now contract every η-edge in $\Delta'([p])$ to a point, and call the resulting 2-complex $\Upsilon([p])$. Note that there is a natural bijection between the edges of $\Upsilon([p])$ and the non-η edges of $\Delta'([p])$. We will often refer to an edge or an h-path of $\Delta'([p])$ as though it were contained in $\Upsilon([p])$ and vice versa. In these cases it should be understood that we are actually referring to the image of that edge or h-path under the aforementioned bijection.

Lemma 9.0.35. *The 2-complex $\Upsilon([p])$ is a reduced planar diagram over the presentation*

$$\langle b, a_1, ..., a_{100} | h(g_n), g_n \in X \rangle.$$

An edge e of $\Upsilon([p])$ is a boundary edge of $\Upsilon([p])$ if and only if e is a boundary edge of $\Delta'([p])$.

Proof. Since the boundary of every 2-cell of $\Upsilon([p])$ is an h-path, if $\Upsilon([p])$ is planar and simply connected then $\Upsilon([p])$ is a diagram over the presentation

$$\langle b, a_1, ..., a_{100} | h(g_n), g_n \in X \rangle.$$

It follows from Lemma 9.0.34 that if $\Delta'([p])$ is not simply connected, and if the boundary of $\Delta'([p])$ contains non-η edges, then these non-η edges are all contained in the same connected component of the boundary of $\Delta'([p])$. Therefore, if the boundary of $\Delta'([p])$ is not composed entirely of η-edges, then there is exactly one component of the boundary of $\Delta'([p])$ that contains non-η edges. The rest of the components of the boundary of $\Delta'([p])$ are composed entirely of η-edges. In this case, it follows that $\Upsilon([p])$ is planar and simply connected.

If the boundary of $\Delta'([p])$ is composed entirely of η-edges then $\Upsilon([p])$ is a spherical diagram over the

presentation $\langle b, a_1, ..., a_{100} | h(g_n), g_n \in X \rangle$. The original special P' 2-complex Δ' contained no pairs of strongly adjacent h-paths. Therefore $\Delta'([p])$ also contained no pairs of strongly adjacent h-paths. This implies that $\Upsilon([p])$ (whether spherical or planar) is reduced, since if $\Upsilon([p])$ contained a pair of reducible 2-cells it would imply that $\Delta'([p])$ contained a pair of strongly adjacent h-paths. Since the presentation $\langle b, a_1, ..., a_{100} | h(g_n), g_n \in X \rangle$ fulfills the small cancelation criteria $C'(\frac{1}{99})$, and since reduced spherical diagrams over such presentations do not exist by Lemma 6.0.20, we conclude that $\Upsilon([p])$ is not spherical.

To complete the proof, we observe that passing from $\Delta'([p])$ to $\Upsilon([p])$ has no effect on whether an edge e of an h-path in $[p]$ is a boundary edge.

□

Lemma 9.0.36. *Suppose Δ' is a special P' 2-complex which does not contain pairs of strongly adjacent h-paths. Then there exists a special P' 2-complex Δ'' with the following properties. The boundary labels $Lab(\partial\Delta'')$ and $Lab(\partial\Delta')$ are equal in the free group $F(b, a_1, a_2, ..., a_{100})$, the special diagram Δ'' has the same area as Δ', there is a constant d such that $|\partial\Delta''| \leq 2d|\partial\Delta'|$, and Δ'' contains no pairs of non-contiguous edge adjacent h-paths.*

Proof. We define an interior edge e of Δ' to be *improper* if e is contained in an h-path p and e^{-1} is contained in an h-path p' where p and p' are not contiguous. Note that if a special P' 2-complex contains no improper edges, then it contains no pairs of edge adjacent h-paths that are not contiguous. Suppose that $[p_1], \dots [p_n]$ are all the $\equiv_e$ equivalence classes in Δ' that contain pairs of edge adjacent h-paths that are not contiguous. Note that every improper edge of Δ' is contained in an h-path of one of the equivalence classes $[p_1] \dots [p_n]$.

We first show that the number of improper edges of Δ' is "small" compared to the boundary length of Δ'. We start by noting that each improper edge of Δ' is contained in $\Upsilon([p_i])$, for some $1 \leq i \leq n$. Each 2-complex $\Upsilon([p_i])$ is a diagram over the $C'(\frac{1}{99})$ presentation

$$\langle b, a_1, ..., a_{100} | h(g_n), g_n \in X \rangle.$$

By Lemma 6.0.20, the number of edges in each $\Upsilon([p_i])$ does not exceed a constant multiple of $|\partial\Upsilon([p_i])|$. By Lemmas 9.0.34 and 9.0.35, the sum of the boundary lengths of all of the $\Upsilon([p_i])$ diagrams does not exceed the boundary length of Δ'. Therefore there is some constant d such that the number of edges of Δ' contained in h-paths of the $[p_1], \dots [p_n]$ is less than $d|\partial\Delta'|$. It follows that the number of improper edges of Δ' is less

than $d|\partial\Delta'|$.

We now show how to construct Δ'' from Δ'. Suppose p is an h-path contained in some $[p_i]$. We consider $\Upsilon([p_i])$. Choose an h-path p' of $\Upsilon([p_i])$ that contains a boundary edge of $\Upsilon([p_i])$. By Lemmas 9.0.34 and 9.0.35, the h-path p' contains a boundary edge of Δ'. By the definition of $[p_i]$, at least one edge of p' must be an interior edge of Δ'. Therefore in the diagram Δ', there is an improper edge $e \in p'$ that is an interior edge of Δ' such that one of the endpoints v of e lies in the boundary of Δ' (assume without loss of generality that v is the final point of e). We draw a new extremal edge e' that has v as its final point such that the path $e'e^{-1}$ has label xx^{-1} where $x = \text{Lab}(e)$. This increases the boundary length of Δ' by 2 and inserts the subword $x^{-1}x$ into the boundary label of Δ'. We can now perform a folding surgery on the path $e'e^{-1}$, which affects neither the boundary label nor the area of Δ', but does decrease the number of improper edges contained in Δ by 2. We can repeat this procedure until no more improper edges remain. We call the resulting diagram Δ''.

Note that $\text{Lab}(\partial\Delta'')$ and $\text{Lab}(\partial\Delta')$ are equal in the free group $F(b, a_1, a_2, ..., a_{100})$, and Δ'' has the same area as Δ'. For each new extremal edge that is added, the boundary length of the special diagram increases by two. The number of extremal edges added does not exceed the number of edges contained in the h-paths of $[p_1], \ldots, [p_n]$. As stated above, the number of such edges is less than $d|\partial\Delta'|$. Therefore $|\partial\Delta''| \leq 2d|\partial\Delta'|$.

□

Lemma 9.0.37. *Suppose Δ' is a minimal area P' diagram with proper boundary label. Then there exists a special P' 2-complex with the same area and boundary label as Δ' which contains no pairs of non-contiguous edge adjacent h-paths.*

Proof. We first transform Δ' into an adjusted P' 2-complex by drawing labeled paths in the interiors of the 2-cells of Δ'. For convenience, we will continue to denote this adjusted P' 2-complex by Δ'. We then construct a spherical 2-complex $\hat{\Delta}'$ by gluing a 2-cell π_0 to the boundary of Δ'. Note that since $\text{Lab}(\partial\pi_0) = (\text{Lab}(\partial\Delta')^{-1})$, the 2-complex $\hat{\Delta}'$ is a spherical special $\hat{P}'$ 2-complex, where $\hat{P}'$ is the h-presentation obtained by cyclically reducing $\text{Lab}(\partial\Delta')$ and adding the resulting word to the relator set of P'. By Corollary 5, $\hat{\Delta}'$ can be transformed via a sequence of folding surgeries into a spherical special $\hat{P}'$ 2-complex containing no pairs of non-contiguous strongly adjacent h-paths.

After these folding surgeries, we claim that $\hat{\Delta}'$ also contains no pairs of non-contiguous edge adjacent

h-paths. To prove this claim we will use a construction similar to the construction of the Υ diagrams described earlier in this section.

Suppose toward a contradiction that there exists an h-path p in $\hat{\Delta}'$ that is edge adjacent but not contiguous to another h-path of $\hat{\Delta}'$. As we did earlier in this section, we can consider the $\equiv_e$ equivalence class $[p]$ of p. Suppose π is a 2-cell of $\hat{\Delta}'$ and $\{p_1, ..., p_n\}$ is the set of h-paths in $\partial\pi$ that are also in $[p]$. As before, we can transform π into $(n+1)$ new 2-cells by placing n new directed edges, labeled $\eta(\pi, p_1), ..., \eta(\pi, p_n)$, in the interior of π. The initial and final vertices of the edge labeled $\eta(\pi, p_i)$ are the initial and final vertices of p_i respectively. Just as before, all but one of these $(n+1)$ new 2-cells are $[p]$-cells.

We perform this operation on every 2-cell of $\hat{\Delta}'$. As a result, for each 2-cell $\pi' \subseteq \hat{\Delta}'$, if $p_1, ..., p_n$ are all the h-paths in $\partial(\pi)$ that are also contained in $[p]$, then π' is transformed into $(n+1)$ new 2-cells, n of which are $[p]$-cells. We consider the 2-complex $\hat{\Delta}'([p])$ consisting of only the $[p]$-cells of $\hat{\Delta}'([p])$.

Since $\hat{\Delta}'([p])$ is a subcomplex of a spherical 2-complex, $\hat{\Delta}'([p])$ is homeomorphic to a sphere with $m \geq 1$ holes. Every edge of $\hat{\Delta}'([p])$ is either an η-edge or is contained in an h-path of $[p]$. Since $\hat{\Delta}'$ is spherical, by the definition of $[p]$, every edge of every h-path $[p]$ is contained in the inverse of another h-path of $[p]$. This means that every non-η-edge of $\hat{\Delta}'([p])$ is an interior edge of $\hat{\Delta}'([p])$. By the argument in the second paragraph of Lemma 9.0.34, every η-edge of $\hat{\Delta}'([p])$ is a boundary edge of $\hat{\Delta}'([p])$. Therefore the set of boundary edges of $\hat{\Delta}'([p])$ is exactly the set of η-edges of $\hat{\Delta}'([p])$. If we contract every η-edge of $\hat{\Delta}'([p])$ to a point, the resulting 2-complex $\hat{\Upsilon}([p])$ is a spherical diagram over the presentation $\langle b, a_1, ..., a_{100} | h(g_n), g_n \in X \rangle$. Since $\hat{\Delta}'$ contains no pairs of strongly adjacent h-paths, $\hat{\Upsilon}([p])$ is reduced. This is a contradiction since such a diagram cannot exist by Lemma 6.0.20. Therefore $\hat{\Delta}'$ contains no pairs on non-contiguous edge adjacent paths.

Thus if we delete the 2-cell π_0 from $\hat{\Delta}'$ then the resulting 2-complex will be a special P' 2-complex with the same area and boundary label as Δ', and will contain no pairs of non-contiguous edge adjacent h-paths.

□

We can now use Lemma 9.0.31 to show that the map $h : G \to H$ is a group embedding. For a word w in the generators of $P(M_\infty)$, suppose the non-reduced word $h(w)$ is trivial in $P'(M_\infty)$. Then there is a $P'(M_\infty)$ diagram Δ' with the proper boundary label $h(w)$. We transform Δ into a special P' 2-complex with identical area and boundary label by drawing labeled directed paths in the interiors of the 2-cells of Δ'. We continue to call this special P' 2-complex Δ'. By Lemma 9.0.37 we can transform Δ' into a special

P' 2-complex containing no pairs of edge adjacent h-paths. By Lemma 9.0.37, this transformation does not affect the boundary label or area of Δ'. Since Δ' has proper boundary label, we can assume by Lemma 9.0.31 that Δ' is proper. Therefore there is a $P(M_\infty)$ diagram Δ such that $h(\Delta) = \Delta'$. The diagram Δ has boundary label w, so w is trivial in $P(M_\infty)$.

Lemma 9.0.38. *The Dehn function f' of $P'(M_\infty)$ is equivalent to the Dehn function f of $P(M_\infty)$.*

Proof. Let Δ' be an arbitrary minimal area $P'(M_\infty)$ diagram. By Lemma 9.0.33 there exists a special $P'(M_\infty)$ 2-complex with the same boundary label and area as Δ' that contains no pairs of strongly adjacent h-paths. For convenience, we also call this special diagram Δ'. By Lemma 9.0.36, we can then transform Δ' into a special $P'(M_\infty)$ diagram Δ'' with the same area as Δ' such that Δ'' contains no pairs of edge adjacent h-paths and $|\partial\Delta''| \leq 2d|\partial\Delta'|$.

Let E be the set of edges of Δ'' such that $e \in E$ if and only if neither e nor e^{-1} is contained in the inner boundary of any 2-cell of Δ''. Let $\Delta''_1, \ldots \Delta''_n$ be the maximal special P' sub-2-complexes of Δ'' that contain a 2-cell, do not contain any E edges, and do not contain pairs of non-contiguous adjacent h-paths. Note that for every edge $e \in \Delta''_i$, either e or e^{-1} is contained in the inner boundary of a 2-cell of Δ''_i (and therefore in an h-path of Δ''_i). Thus by Lemma 9.0.32, we can assume that $\Delta''_1, \ldots \Delta''_n$ are proper P' 2-complexes. Since Δ'' contains no pairs of non-contiguous edge adjacent h-paths, every boundary edge of Δ''_i is a boundary edge of Δ. Therefore the sum of the boundary lengths of $\Delta''_1, \ldots \Delta''_n$ does not exceed $|\partial\Delta''| \leq 2d|\partial\Delta'|$. Since $\Delta''_1, \ldots, \Delta''_n$ are proper P' 2-complexes, we can consider the $P(M_\infty)$ diagrams $h^{-1}(\Delta''_1), \ldots, h^{-1}(\Delta''_n)$. Let $\Delta_i = h^{-1}(\Delta''_i)$. We observe that

$$\Sigma_{i=1}^n |\partial\Delta_i| \leq \Sigma_{i=1}^n |\partial\Delta''_i| \leq 2d|\partial\Delta'|.$$

Suppose f is the Dehn function of $P(M_\infty)$ and f' is the Dehn function of $P'(M_\infty)$. Since f is equivalent to a superadditive function, the sum of the areas of $\Delta_1, \ldots, \Delta_n$ is bounded above by $d_1 f(d_1(\Sigma_{i=1}^n |\partial\Delta_i|))$ for some constant d_1. Since Dehn functions are non-decreasing, $d_1 f(d_1(\Sigma_{i=1}^n |\partial\Delta_i|)) \leq d_1 f(d_1(\Sigma_{i=1}^n |\partial\Delta''_i|))$, and since the area of Δ''_i equals the area of Δ_i, we conclude that the sum of the areas of $\Delta''_1, \ldots, \Delta''_n$ is bounded above by $d_1 f(d_1(\Sigma_{i=1}^n |\partial\Delta''_i|))$. Therefore the area of Δ' is bounded above by $d_1 f(d_1(2d|\partial\Delta'|))$. This proves that $f' \preceq f$.

To prove that $f \preceq f'$, consider the set of computational discs of $P(M_\infty)$. Each of these computational discs has boundary label of the form $K(W)$ for some admissible word W of $S(M'_\infty)$. Some admissible words of $S(M'_\infty)$ are of the form $\sigma(\mathbf{c})$ for a configuration $\mathbf{c}$ of M'_∞. Since the input alphabet of M'_∞ is finite, and every input configuration of M'_∞ has the same state, there are only finitely many state and tape letters that appear in the input configurations of M'_∞. Therefore, by the definition of σ, there are only finitely many generators of $S(M'_\infty)$ that appear in admissible words of the form $\sigma(\mathbf{c})$, where $\mathbf{c}$ is an input configuration of M'_∞. Therefore there are only finitely many generators of $P(M_\infty)$ that appear in the boundary labels of computational discs of the form $K(\sigma(\mathbf{c}))$ where $\mathbf{c}$ is an input configuration of M'_∞. We call these boundary labels *input labels*.

Since the input labels are composed of a finite set of generators of $P(M_\infty)$, there is a constant bound on the difference in length between an input label $K(\sigma(\mathbf{c}))$ and its h-image. To be precise, there is some constant d_2 such that for any input label $K(\sigma(\mathbf{c}))$, we have $|h(K(\sigma(\mathbf{c})))| \leq d_2|K(\sigma(\mathbf{c}))|$. Suppose $h(K(\sigma(\mathbf{c})))$ is trivial in $P'(M_\infty)$. By Lemmas 9.0.31 and 9.0.37, there is a proper diagram Δ' with boundary label $h(K(\sigma(\mathbf{c})))$. Since Δ' is proper, the area of Δ' is equal to the area of $h^{-1}(\Delta')$, which has boundary label $K(\sigma(\mathbf{c}))$. By Lemma 7.0.23, we conclude that $f' \succeq T^4$, where T^4 is the area function of $S(M'_\infty)$. Since $f \approx T^4$, we conclude that $f' \succeq f$.

□

Lemma 9.0.39. *If the word problem for $P(M_\infty)$ is solvable, then the word problem for $P'(M_\infty)$ is solvable.*

Proof. Suppose that the word problem for $P(M_\infty)$ is solvable. Let w be an arbitrary word in the generators of $P'(M_\infty)$.

If w is trivial in $P'(M_\infty)$ then there exists a special $P'(M_\infty)$ 2-complex Δ' with boundary label w. By Lemmas 9.0.33 and 9.0.36, Δ' can be transformed into a special $P'(M_\infty)$ 2-complex Δ'' in which there are no pairs of non-contiguous edge adjacent h-paths and $|\partial\Delta''| \leq 2d|\partial\Delta'|$. Let $\Delta''_1, ..., \Delta''_i \subseteq \Delta''$ be defined as in the proof of Lemma 9.0.38. By the arguments of the proof of Lemma 9.0.38, each Δ''_i is a proper $P'(M_\infty)$ 2-complex, every 2-cell of Δ'' is contained in some Δ''_i, and every boundary edge of each Δ''_i is a boundary edge of Δ''. Therefore the boundary label of Δ'' is a product of conjugates of the boundary labels of the Δ''_i special P' 2-complexes.

Since $P'(M_\infty)$ is finitely generated we can consider the finite set of words of the form $w' = h(g_{i_1})...h(g_{i_n})$ such that $|w'| \leq O(|\partial\Delta'|)$ (i.e., all the potential proper boundary labels for each Δ_i'') . Since the word problem for $P(M_\infty)$ is solvable, and since h is a group embedding, we can decide which of these words are trivial in $P'(M'_\infty)$. We can then effectively decide whether there is any tuple of conjugates of such words in $P'(M_\infty)$ such that their reduced product is w. The word w is trivial in $P'(M_\infty)$ if and only if there exists such a tuple.

□

Lemma 9.0.40. *The presentation $P'(M_\infty)$ is minimal.*

Proof. Suppose r is a relator of $P(M_\infty)$ and r' is the relator of $P'(M_\infty)$ obtained by cyclically reducing $h(r)$. Suppose we remove r' from $P'(M_\infty)$ and call the resulting presentation $P'(M_\infty) \setminus \{r'\}$. We then attempt to construct a $P'(M_\infty) \setminus \{r'\}$ diagram whose boundary label is the non-reduced word $h(r)$. If such a diagram Δ' exists, then by Lemmas 9.0.31 and 9.0.37, we can transform Δ' into a proper $P'(M_\infty) \setminus \{r'\}$ 2-complex. The h^{-1} image of this proper 2-complex will be a $P(M_\infty) \setminus \{r\}$ diagram with boundary label r. By Lemma 8.0.29, this is impossible. Therefore $P'(M_\infty)$ is minimal.

□

Proof of Theorem 1.0.4. Part 1 follows from Lemma 7.0.24 and the fact that h is a group embedding. Part 2 follows from Lemma 8.0.25 and Lemma 9.0.39. Part 3 follows from Lemma 7.0.24 and Lemma 9.0.38.

□

Chapter 10

Conclusion

We can now use Theorem 1.0.4 to create examples of finitely generated minimal decidable group presentations satisfying the cases **3**, **4**, **6**, **7**, and **8**. Let $\mathbf{B} \subseteq \{0,1\}^*$ be the set of binary representations of natural numbers. In this section, when we want to indicate a particular element $x \in \mathbf{B}$, we will often simply refer to the natural number for which x is the binary representation. For example, if we refer to the set of elements in $\mathbf{B}$ that are < 1000, we mean to indicate the set elements in $\mathbf{B}$ that are binary representations of natural numbers < 1000.

Let $\mathbf{K} \subset \mathbf{B}$ be a language over alphabet $\{0,1\}^*$ that is enumerable but not decidable. It is possible to program a union machine that can query the membership problem of $\mathbf{K}$. Such a union machine behaves similarly to the oracle Turing machines commonly used in computability theory [11].

We will denote such a union machine by $M_\infty^{\mathbf{K}}$. The machine $M_\infty^{\mathbf{K}}$ will query the membership problem of $\mathbf{K}$ using its kth tape. The kth set of state letters is $Q_k = \{q_k^{\mathbf{y}} | \mathbf{y} \in \{0,1\}^*\}$. The other sets of state letters $Q_1, ..., Q_{k-1}$ are required to be finite. The machine never writes any letters in the kth tape. Instead, it will use the upper index of the kth state letter as the kth tape. We explain this formally below.

The commands of $M_\infty^{\mathbf{K}}$ come in two types: query commands and non-query commands. We require that the kth component of a non-query command has one of the three following forms:

- $\alpha_k q_k^{\mathbf{y}} \omega_k \rightarrow \alpha_k q_k^{\mathbf{y}a} \omega_k$,
- $\alpha_k q_k^{\mathbf{y}a} \omega_k \rightarrow \alpha_k q_k^{\mathbf{y}} \omega_k$,
- $\alpha_k q_k^{\mathbf{y}} \omega_k \rightarrow \alpha_k q_k^{\mathbf{y}} \omega_k$,

where $\mathbf{y} \in \{0,1\}^*$ and $a \in \{0,1\}$. Thus, the upper index of the kth state letter behaves like a one-sided

tape. We also require that the set of non-query commands be decidable. Note that this is equivalent to requiring that the algorithm consisting of the non-query commands of $M_\infty^{\mathbf{K}}$ could be performed by a standard non-deterministic Turing machine.

The query commands of $M_\infty^{\mathbf{K}}$ are used to ask whether the upper index of the kth state letter in a given configuration is in $\mathbf{K}$. For each $\mathbf{y} \in \mathbf{K}$, there is a query command of the form

$$\tau_{\mathbf{y}} = (q_1 \to q_1', ..., q_{k-1} \to q_{k-1}', q_k^{\mathbf{y}} \to q_k^{\mathbf{y}}).$$

The state $(q_1', ..., q_{k-1}', q_k^{\mathbf{y}})$ is a "yes" state, indicating that the element $\mathbf{y}$ is a member of $\mathbf{K}$. If $\mathbf{y} \notin \mathbf{K}$, then no query command $\tau_{\mathbf{y}}$ exists in $M_\infty^{\mathbf{K}}$. In this way the machine $M_\infty^{\mathbf{K}}$ is different from an oracle Turing machine: while oracle machines may receive negative answers to queries, there is no way for $M_\infty^{\mathbf{K}}$ to receive a negative answer to a query.

The machine $M_\infty^{\mathbf{K}}$ is enumerable. This is because the input and work alphabets of $M_\infty^{\mathbf{K}}$ are finite, the set of states of $M_\infty^{\mathbf{K}}$ is decidable, the set of non-query commands of $M_\infty^{\mathbf{K}}$ is decidable, and (since $\mathbf{K}$ is enumerable) the set of query commands is enumerable.

We can now use Theorem 1.0.4 to construct group presentations with desired properties from union machines of the form $M_\infty^{\mathbf{K}}$. We will first construct a presentation to satisfy case **7**. For the following construction, we will assume that $\mathbf{K}$, in addition to being enumerable and undecidable, contains no elements less than 1000. Also we assume that $\mathbf{K}$ contains every even natural number ≥ 1000.

We begin by describing a union machine $M_\infty^{\mathbf{K}}$. If $M_\infty^{\mathbf{K}}$ is given $\mathbf{u}$ as an input word then $M_\infty^{\mathbf{K}}$ writes the binary representation of $|\mathbf{u}|$ in the upper index of the kth state letter. If $|\mathbf{u}| \in \mathbf{K}$, there will be a query command in $M_\infty^{\mathbf{K}}$ that can then be executed. If this query command is executed then $M_\infty^{\mathbf{K}}$ continues to run, eventually entering the accept state when the total number of steps in the computation reaches $|\mathbf{u}|^2$. If there is no query command that can be applied (i.e. if $|\mathbf{u}|$ is not in $\mathbf{K}$), then there is no way to reach the accept state. Thus the language accepted by $M_\infty^{\mathbf{K}}$ is the set of input words whose lengths are in $\mathbf{K}$.

We claim that the time function T of $M_\infty^{\mathbf{K}}$ (and therefore T^4 as well) is superadditive. Note that for

$n < 1000$, $T(n) = 0$. For every acceptable input word $\mathbf{u}$ of $M_\infty^{\mathbf{K}}$, the minimal length accepting computation for $\mathbf{u}$ has length $|\mathbf{u}|^2$. For any input length $n \geq 1000$, there is an acceptable input word $\mathbf{u}$ for $M_\infty^{\mathbf{K}}$ with $(n-1) \leq |\mathbf{u}| \leq n$ (this is because $\mathbf{K}$ contains every even natural number ≥ 1000). Therefore, for $n \geq 1000$, $T(n)$ is either n^2 or $(n-1)^2$. Suppose $n_1, n_2, n_3 \in \mathbf{N}$ and $n_1 = n_2 + n_3$. Then $T(n_2) + T(n_3) \leq n_2^2 + n_3^2$ and $T(n_1) \geq (n_2 + n_3 - 1)^2$. Therefore if either n_2 or n_3 is less that 1000, $T(n_2) + T(n_3) \leq T(n_1)$. If $n_2, n_3 > 1000$ then $n_2^2 + n_3^2 < (n_2 + n_3 - 1)^2$, and therefore $T(n_2) + T(n_3) \leq T(n_1)$. We conclude that T is superadditive and we can therefore apply Theorem 1.0.4 to $M_\infty^{\mathbf{K}}$.

Note that T is equivalent to x^2. By Theorem 1.0.4, the Dehn function f of $P'(M_\infty^{\mathbf{K}})$ is equivalent to x^8. Since $\mathbf{K}$ is undecidable, the language L accepted by $M_\infty^{\mathbf{K}}$ is not decidable. Therefore, by Theorem 1.0.4 part 1, the word problem for $P'(M_\infty^{\mathbf{K}})$ is not solvable.

Theorem 1.0.4 tells us the Dehn function f of $P'(M_\infty^{\mathbf{K}})$ up to equivalence, but this is not sufficient to conclude that f is computable. For that, we need to prove the following lemma. Consider the Baulmslag-Solitar presentation $H = \langle s, t \| sts^{-2}t^{-1} \rangle$, where s, t are not among the generators of $P'(M_\infty^{\mathbf{K}})$.

Definition 4. Let P be an arbitrary presentation. For a word w trivial in P, we define $L_P(w)$ to be the area of the minimal area P diagram for w.

Lemma 10.0.41. *Suppose H is the Baulmslag-Solitar presentation defined above and P is a finitely generated presentation whose Dehn function is equivalent to a superadditive polynomial function. If $J = H * P$ then the Dehn function f_J of J is computable.*

Proof. Let f_P be the Dehn function of P, and f_H be the Dehn function of H. It is well known that the Dehn function f_H of the Baulmslag-Solitair presentation H is equivalent to the exponential function 2^x. Let f_J denote the Dehn function of J.

Let w be an arbitrary reduced trivial word in J. Let $|w|_H$ denote the number of H letters in w, and $|w|_P$ denote the number of P letters in w. Note that, since f_P and f_H are both equivalent to superadditive functions, there is a constant b_1 such that

$$L_J(w) \leq b_1 f_P(b_1(|w|_P)) + b_1 f_H(b_1(|w|_H)). \tag{10.0.1}$$

Let $n, m \in \mathbf{N}$. We will compare the quantities $f_H(n+m)$ and $f_H(n) + f_P(m)$. Since f_P is equivalent to a polynomial function and f_H is equivalent to an exponential function, there exists a constant b_2 such that

if $m > b_2$ then the following inequality holds:

$$f_H(n+m) \geq f_H(n) + f_P(m). \tag{10.0.2}$$

Using (10.0.2) and (10.0.1), we conclude that if a word w trivial in J has $|w|_P \geq b_2$, then $L_J(w) \leq b_1 f_H(b_1|w|)$. This means that for any word w trivial in J with $|w|_P \geq b_2$, there exists a word w' trivial in J with $|w'| = |w|$ and $L_J(w') \geq L_J(w)$. Informally, this means that if w is trivial in J and $|w|_P \geq b_2$ then w need not be considered when computing f_J.

If w is a word in J with $L_J(w) \geq b_1 f_H(b_1|w|)$, then it follows that $|w|_P < b_2$. Since P is finitely generated, there are only finitely many words trivial in P of length $\leq b_2$. We can collect the finitely many minimal area P diagrams for such words and then consider the finite set R_1 of P relators that appear in these diagrams. The only P relators that appear in a minimal area J diagram for a word w with $|w|_P < b_2$ will be members of R_1. Therefore we can compute $f_J(x)$ as follows. By (10.0.1), $f_J(x) \leq b_1 f_H(b_1 x) + b_1 f_P(b_1 x)$. We can therefore effectively compute an upper bound for $f_J(x)$. We then effectively construct all minimal area diagrams over the presentation $H * R_1$ with boundary length not exceeding x, and with area not exceeding the upper bound for $f_J(x)$. The area of the largest such diagram is $f_J(x)$.

□

By this lemma, the Dehn function of $H * P'(M_\infty^{\mathbf{K}})$ is computable. Since the word problem for $H * P(M_\infty^{\mathbf{K}})$ is not solvable, and since no finitely generated decidable group presentation exists satisfying case **5** (by Lemma 1.0.1), the bounded word problem for $H * P(M_\infty^{\mathbf{K}})$ is unsolvable. By Lemma 9.0.40, the fact that H only contains one relator, and the obvious fact that a free product of two minimal presentations is a minimal presentation, $H * P'(M_\infty^{\mathbf{K}})$ is minimal.

Next we consider case **3**. For the following construction, we will assume that $\mathbf{K}$, in addition to being enumerable and undecidable, contains no even numbers.

We create a union machine $M_\infty^{\mathbf{K}}$ such that when $M_\infty^{\mathbf{K}}$ is given an input word $\mathbf{u}$, if $|\mathbf{u}| < 1000$ then $M_\infty^{\mathbf{K}}$ does not accept. If $|\mathbf{u}| \geq 1000$ then $M_\infty^{\mathbf{K}}$ writes the binary representation of $|\mathbf{u}|$ in the upper index of the kth state letter and attempts to execute a query command. If a query command is executed (which can only happen if $|\mathbf{u}| \in \mathbf{K}$) then $M_\infty^{\mathbf{K}}$ accepts immediately. Otherwise, $M_\infty^{\mathbf{K}}$ continues to run until the total number of steps in the computation reaches $|\mathbf{u}|^{10}$ and then accepts.

We claim that the time function T (and therefore T^4) is supperadditive. For any input length $n > 1000$, there is an acceptable input word $\mathbf{u}$ for $M_\infty^{\mathbf{K}}$ with $(n-1) \leq |\mathbf{u}| \leq n$ such that $|\mathbf{u}|$ is not in $\mathbf{K}$ (this is because $\mathbf{K}$ contains no even natural numbers). Therefore for $n < 1000$, $T(n) = 0$ and for $n \geq 1000$, $T(n)$ is either n^{10} or $(n-1)^{10}$. Suppose $n_1, n_2, n_3 \in \mathbf{N}$, $n_1 \geq 1000$, and $n_1 = n_2 + n_3$. Note that if either n_2 or n_3 is < 1000, then $T(n_2) + T(n_3) \leq T(n_1)$. Also, note that $T(n_2) + T(n_3) \leq n_2^{10} + n_3^{10}$ and $T(n_1) \geq (n_2 + n_3 - 1)^{10}$. If $n_2, n_3 > 1000$, then $n_2^{10} + n_3^{10} < (n_2 + n_3 - 1)^{10}$. We conclude that T is superadditive and we can therefore apply Theorem 1.0.4 to $M_\infty^{\mathbf{K}}$. Note also that T^4 is equivalent to the superadditive polynomial function $(x^{10})^4$.

By Theorem 1.0.4 part 2, since $M_\infty^{\mathbf{K}}$ accepts every input of length ≥ 1000, the word problem for $P'(M_\infty^{\mathbf{K}})$ is solvable. If the bounded word problem for $P'(M_\infty^{\mathbf{K}})$ were solvable then it would be possible to decide $\mathbf{K}$ as follows. For $n > 1000$, to decide if $n \in \mathbf{K}$, pick an input word $\mathbf{u}$ with $|\mathbf{u}| = n$. Consider the non-reduced word $h(\mathcal{K}(\mathbf{u}))$. Since every input word of length > 1000 is accepted by $M_\infty^{\mathbf{K}}$, the word $h(\mathcal{K}(\mathbf{u}))$ is trivial in $P'(M^{\mathbf{K}})$. If we begin solving the bounded word problem on inputs $(h(\mathcal{K}(\mathbf{u})), 1), (h(\mathcal{K}(\mathbf{u})), 2), ...$, then we can find the size of the minimal area diagram with boundary label $h(\mathcal{K}(\mathbf{u}))$. If $n \in \mathbf{K}$, by Lemmas 9.0.33, 9.0.36, and 7.0.23 this area is at least $O((|\mathbf{u}|^{10})^4)$. Otherwise, by Lemma 7.0.22, it will be at most $O(|\mathbf{u}|^4)$. Since $\mathbf{K}$ is undecidable, we conclude that the bounded word problem is not solvable for $P'(M_\infty^{\mathbf{K}})$.

Now we consider the presentation $H * P'(M_\infty^{\mathbf{K}})$, which (by Lemma 10.0.41) will have computable Dehn function. Since the bounded word problem is unsolvable for $P'(M_\infty^{\mathbf{K}})$, the bounded word problem is also unsolvable for $H * P'(M_\infty^{\mathbf{K}})$. Since the word problem is solvable for H and $P'(M_\infty^{\mathbf{K}})$, the word problem is also solvable for $H * P'(M_\infty^{\mathbf{K}})$. By Lemma 9.0.40, the fact that H only contains one relator, and the fact that a free product of two minimal presentations is a minimal presentation, $H * P'(M_\infty^{\mathbf{K}})$ is minimal.

To provide an example for case **4** we must construct a function f such that f^4 is superadditive, f is the time function of some union machine of the form $M_\infty^{\mathbf{K}}$, and f^4 is not equivalent to any computable function. To construct such an f, we first partition $\mathbf{N}$ into disjoint subsets I_n, where I_n is the set of integers contained in the interval $(10^{(n-1)^2}, 10^{n^2}]$. The function f will be constructed such that the values that f takes on I_n will depend on whether or not $n \in K$.

We now define the function f. For $x \in I_n$, if $n \in \mathbf{K}$ then $f(x) = 10^{2n^2}x$. If $n \notin \mathbf{K}$ then $f(x) = 10^{2n^2+3n}x$. We claim that f (and therefore f^4) is superadditive. To prove this claim, suppose that x_1, x_2, x_3 are positive

integers such that $x_2 + x_3 = x_1$. Also suppose that $x_1 \in I_n$ and $n \in \mathbf{K}$. Then $f(x_1) = 10^{2n^2}x_1$. Note that for all $y \leq x_1$, the following inequality holds: $f(y) \leq 10^{2n^2}y$. Therefore $f(x_2)+f(x_3) \leq 10^{2n^2}x_2+10^{2n^2}x_3 = 10^{2n^2}(x_2 + x_3) = f(x_1)$. A similar argument proves the same result if $n \notin \mathbf{K}$ and $f(x_1) = 10^{2n^2+3n}x$. We conclude that f is superadditive.

We now prove that f^4 is not equivalent to any computable function. For any $x \in I_n$, if $n \in \mathbf{K}$, then $f(x) \leq 10^{2n^2}10^{n^2} = 10^{3n^2}$. Alternately, if $n \notin \mathbf{K}$ then $f(x) > 10^{2n^2+3n}10^{(n-1)^2} = 10^{3n^2}10^{n+1}$. Suppose toward a contradiction that f^4 is equivalent to a computable function f_c. Then there exists a constant b_c such that for all $x \in \mathbf{N}$, $f_c(x) \leq b_c f^4(b_c x)$ and $f^4(x) \leq b_c f_c(b_c x)$. Define $x_n = \lfloor (10^{n^2})/(b_c^2) \rfloor$. Note that for all sufficiently large n, $x_n \geq 10^{(n-1)^2}$. Therefore, for sufficiently large n, the integers $x_n, b_c x_n, b_c^2 x_n$ are all contained in I_n. Suppose that n is sufficiently large that $x_n, b_c x_n, b_c^2 x_n \in I_n$. If $n \notin \mathbf{K}$, then $(10^{3n^2}10^{n+1})^4 < f^4(x_n) \leq b_c f_c(b_c x_n)$. Alternately, if $n \in \mathbf{K}$, then $f_c(b_c x_n) \leq b_c f^4(b_c^2 x_n) \leq b_c(10^{3n^2})^4$.

Thus if $n \notin \mathbf{K}$, then $\frac{1}{b_c}(10^{3n^2}10^{n+1})^4 \leq f_c(b_c x_n)$. If $n \in \mathbf{K}$, then $f_c(b_c x_n) \leq b_c(10^{3n^2})^4$. We note that if n is sufficiently large then $\frac{1}{b_c}(10^{3n^2}10^{n+1})^4 > b_c(10^{3n^2})^4$. Thus for sufficiently large n, we can decide whether $n \in \mathbf{K}$ by evaluating $f_c(b_c x_n)$. We conclude that f^4 is not equivalent to any computable function.

We now construct a union machine $M_\infty^{\mathbf{K}}$ with time function f as follows. The machine $M_\infty^{\mathbf{K}}$ accepts every input. When given input $\mathbf{u}$ the machine $M_\infty^{\mathbf{K}}$ writes $|\mathbf{u}|$ in the upper index of the kth state letter and attempts to execute a query command. If a query command is executed (which can only happen if $|\mathbf{u}| \in \mathbf{K}$) then $M_\infty^{\mathbf{K}}$ continues to run until the total number of steps in the computation reaches $10^{2n^2}|\mathbf{u}|$ and then accepts. Otherwise, $M_\infty^{\mathbf{K}}$ continues to run until the total number of steps in the computation reaches $10^{2n^2+3n}|\mathbf{u}|$ and then accepts. Note that $M_\infty^{\mathbf{K}}$ accepts every input and that the time function of $M_\infty^{\mathbf{K}}$ is equal to the function f defied above.

By Theorem 1.0.4 part 2, since $M_\infty^{\mathbf{K}}$ accepts every input, the word problem for $P'(M_\infty^{\mathbf{K}})$ is solvable. By Theorem 1.0.4 part 3, the Dehn function of $P'(M_\infty^{\mathbf{K}})$ is equivalent to f^4, and is therefore not computable. If the bounded word problem for $P'(M_\infty^{\mathbf{K}})$ were solvable then $P'(M_\infty^{\mathbf{K}})$ would satisfy case **2**, which is impossible by Lemma 1.0.1.

To provide an example of case **6**, we can simply use the machinery from [10]. We let $\hat{M}$ be a Turing machine that accepts the language $\mathbf{K}$ (since $\mathbf{K}$ is undecidable, the time function of $\hat{M}$ is not bounded by any

computable function). For an input word $\mathbf{u}$ of $\hat{M}$, the word $\mathcal{K}(\mathbf{u})$ is trivial in $P(\hat{M})$ if and only if $\mathbf{u} \in \mathbf{K}$. Since $\mathbf{K}$ is undecidable, the word problem for $P(\hat{M})$ is not solvable. Since $P(\hat{M})$ is finite, it follows that the Dehn function of $P(\hat{M})$ is not computable. Additionally, the bounded word problem is solvable for $P(\hat{M})$ because $P(\hat{M})$ is finite.

Finally, for case **8**, if P_1 and P_2 are the examples satisfying cases **6** and **7** respectively, then $P = P_1 * P_2$ satisfies case **8**. Since the word problem for P_1 is not solvable and P_1 is a finite presentation, the Dehn function for P_1 is not bounded above by any computable function. Therefore P does not have computable Dehn function. Since the word problem and bounded word problem are not solvable for P_2, they are also not solvable for P. By Lemma 9.0.40 and the fact that a free product of two minimal presentations is a minimal presentation, $P = P_1 * P_2$ is minimal.

References

[1] G. Baumslag, C. F. Miller, III, and H. Short. Isoperimetric inequalities and the homology of groups. *Invent. Math.*, 113(3):531–560, 1993.

[2] J.-C. Birget, A. Yu. Ol′shanskii, E. Rips, and M. V. Sapir. Isoperimetric functions of groups and computational complexity of the word problem. *Ann. of Math. (2)*, 156(2):467–518, 2002.

[3] S. M. Gersten. Dehn functions and l_1-norms of finite presentations. In *Algorithms and classification in combinatorial group theory (Berkeley, CA, 1989)*, volume 23 of *Math. Sci. Res. Inst. Publ.*, pages 195–224. Springer, New York, 1992.

[4] Rostislav I. Grigorchuk and Sergei V. Ivanov. On Dehn functions of infinite presentations of groups. *Geom. Funct. Anal.*, 18(6):1841–1874, 2009.

[5] M. Gromov. Hyperbolic groups. In *Essays in group theory*, volume 8 of *Math. Sci. Res. Inst. Publ.*, pages 75–263. Springer, New York, 1987.

[6] M. Gromov. Asymptotic invariants of infinite groups. In *Geometric group theory, Vol. 2 (Sussex, 1991)*, volume 182 of *London Math. Soc. Lecture Note Ser.*, pages 1–295. Cambridge Univ. Press, Cambridge, 1993.

[7] Roger C. Lyndon and Paul E. Schupp. *Combinatorial group theory*. Classics in Mathematics. Springer-Verlag, Berlin, 2001. Reprint of the 1977 edition.

[8] Klaus Madlener and Friedrich Otto. Pseudonatural algorithms for finitely generated presentations of monoids and groups. *J. Symbolic Comput.*, 5(3):339–358, 1988.

[9] Joseph J. Rotman. *An introduction to the theory of groups*. Allyn and Bacon Inc., Boston, MA, third edition, 1984.

[10] Mark V. Sapir, Jean-Camille Birget, and Eliyahu Rips. Isoperimetric and isodiametric functions of groups. *Ann. of Math. (2)*, 156(2):345–466, 2002.

[11] Robert I. Soare. *Recursively enumerable sets and degrees*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.