Journal of Algebra 320 (2008) 4030–4062

Contents lists available at ScienceDirect

Journal of Algebra

www.elsevier.com/locate/jalgebra

One-way permutations, computational asymmetry and distortion ☆

Jean-Camille Birget

Department of Computer Science, Rutgers University at Camden, Camden, NJ 08102, USA

ARTICLE INFO

Article history:
Received 6 May 2008
Available online 26 July 2008
Communicated by Derek Holt

Keywords:
Combinatorial group theory
Complexity

ABSTRACT

Computational asymmetry, i.e., the discrepancy between the complexity of transformations and the complexity of their inverses, is at the core of one-way transformations. We introduce a computational asymmetry function that measures the amount of one-wayness of permutations. We also introduce the word-length asymmetry function for groups, which is an algebraic analogue of computational asymmetry. We relate combinational circuits to words in a Thompson monoid, over a fixed generating set, in such a way that circuit size is equal to word-length. Moreover, combinational circuits have a representation in terms of elements of a Thompson group, in such a way that circuit size is polynomially equivalent to word-length. We show that circuits built with gates that are not constrained to have fixed-length inputs and outputs, are at most quadratically more compact than circuits built from traditional gates (with fixed-length inputs and outputs). Finally, we show that the computational asymmetry function is closely related to certain distortion functions: The computational asymmetry function is polynomially equivalent to the distortion of the path length in Schreier graphs of certain Thompson groups, compared to the path length in Cayley graphs of certain Thompson monoids. We also show that the results of Razborov and others on monotone circuit complexity lead to exponential lower bounds on certain distortions.

☆ Supported by NSF grant CCR-0310793. Some of the results of this paper were presented at the AMS Section Meeting, October 21–23, 2005, Lincoln, Nebraska (http://www.ams.org/amsmtgs/2117_program.html), and at the conference "Various Faces of Cryptography," 10 November 2006 at City College of CUNY, New York. The first version of the paper appeared in http://arxiv.org/abs/0704.1569 (12 April 2007).

E-mail address: birget@camden.rutgers.edu.

0021-8693/$ – see front matter
doi:10.1016/j.jalgebra.2008.05.035

1. Introduction

The existence of one-way permutations or of one-way functions (i.e., permutations or functions that are “easy to evaluate” but “hard to invert”) is a major open problem. In this paper we give some connections between this question and some group-theoretic concepts:

(1) We continue the work of [7–9] on the relation between combinational circuits, on the one hand, and the Thompson–Higman groups and monoids on the other hand. We give a representation of any combinational circuit by a word over the Thompson group, such that circuit size is polynomially equivalent to word-length.
(2) We establish connections between the existence of one-way permutations and the distortion function in a certain Thompson group. Distortion is an important concept in metric spaces (e.g., Bourgain [11]) and in combinatorial group theory (e.g., Gromov [18], Farb [15]).

As in [7–9], we treat the Thompson–Higman groups and monoids as *models of computation*, rather than just a source of computational problems; every combinational circuit can be represented by a word over group or monoid generators. This enables us to place open problems from computational complexity into an algebraic setting: On these groups and monoids we define functions that have an algebraic and geometric meaning and, in addition, are polynomially related to some traditional complexity functions.

Overview. Subsections 1.1–1.6 of the present section define and motivate the concepts used: One-way functions and one-way permutations; computational asymmetry; word-length asymmetry; reversible computing; distortion; Thompson groups and monoids. In Section 2 we show that circuits can be represented by elements of Thompson monoids: A combinational circuit is equivalent to a word over a fixed generating set of a Thompson monoid, with circuit size being equal (or linearly equivalent) to word-length over the generating set. The Thompson monoids that appear here are monoid generalizations of the Thompson group $G_{2,1}$, obtained when bijections are generalized to partial functions [9]. Section 3 shows that computational asymmetry and word-length asymmetry (for the Thompson groups and monoids) are linearly related. In Section 4 we give a representation of arbitrary (not necessarily bijective) circuits by elements of the Thompson group $G_{2,1}$; circuit size is polynomially equivalent to word-length over a certain generating set in the Thompson group. In Section 5 we show that the computational asymmetry function of permutations is polynomially related to a certain distortion in a Thompson group. Section 6 contains miscellaneous results, in particular that the work of Razborov and others on monotone circuit complexity leads to exponential lower bounds on certain distortion functions. Finally, we state some open problems about the distortion of the Thompson–Higman groups within the Thompson–Higman monoids, related to the existence of one-way permutations.

1.1. One-way functions and one-way permutations

Intuitively, a one-way function is a function f (mapping words to words, over a finite alphabet), such that f is “easy to evaluate” (i.e., given x_0 in the domain, it is “easy” to compute $f(x_0)$), but “hard to invert” (i.e., given y_0 in the range, it is “hard” to find any x_0 such that $f(x_0) = y_0$). The concept was introduced by Purdy [31] and Diffie and Hellman [14].

There are many ways of defining the words “easy” and “hard,” and accordingly there exist many different rigorous notions of a one-way function, all corresponding to a similar intuition. It remains an open problem whether one-way functions exist, for any “reasonable” definition. Moreover, for certain definitional choices, this problem is a generalization of the famous question whether $\mathsf{P} \neq \mathsf{NP}$ [12,17,36].

We will base our one-way functions on combinational circuits and their size. The size of a circuit will also be called its complexity. Below, $\{0,1\}^n$ (for any integer $n \geqslant 0$) denotes the set of all bitstrings of length n. A *combinational* circuit with input–output function $f : \{0,1\}^m \to \{0,1\}^n$ is an acyclic boolean circuit with m input wires (or “input ports”) and n output wires (or “output ports”).

The circuit is made from gates of type not, and, or, fork, as well as wire-crossings or wire-swappings. These gates are very traditional and are defined as follows.

and: $(x_1, x_2) \in \{0,1\}^2 \longmapsto y \in \{0,1\}$, where $y = 1$ if $x_1 = x_2 = 1$, and $y = 0$ otherwise.
or: $(x_1, x_2) \in \{0,1\}^2 \longmapsto y \in \{0,1\}$, where $y = 0$ if $x_1 = x_2 = 0$, and $y = 1$ otherwise.
not: $x \in \{0,1\} \longmapsto y \in \{0,1\}$, where $y = 0$ if $x = 1$, $y = 1$ otherwise.
fork: $x \in \{0,1\} \longmapsto (x, x) \in \{0,1\}^2$.

Another gate that is often used is the exclusive-or gate,

xor: $(x_1, x_2) \in \{0,1\}^2 \longmapsto y \in \{0,1\}$, where $y = 1$ if $x_1 \neq x_2$, and $y = 0$ otherwise.

The wire-swapping of the ith and jth wire ($i < j$) is described by the *bit transposition* (or bit position transposition)

$$\tau_{i,j} : ux_i vx_j w \in \{0,1\}^\ell \longmapsto ux_j vx_i w \in \{0,1\}^\ell, \quad \text{where } |u| = i-1,\ |v| = j-i-1,\ |w| = \ell - j - 1.$$

The fork and wire-swapping operations, although heavily used, are usually not explicitly called "gates"; but because of their important role we will need to consider them explicitly. Other notations for the gates: $\mathsf{and}(x_1, x_2) = x_1 \wedge x_2$, $\mathsf{or}(x_1, x_2) = x_1 \vee x_2$, $\mathsf{not}(x) = \bar{x}$, $\mathsf{xor}(x_1, x_2) = x_1 \oplus x_2$.

A combinational circuit for a function $f : \{0,1\}^m \to \{0,1\}^n$ is defined by an acyclic directed graph drawn in the plane (with crossing of edges allowed). In the circuit drawing, the m input ports are vertices lined up in a vertical column on the left end of the circuit, and the n output ports are vertices lined up in a vertical column on the right end of the circuit. The input and output ports and the gates of the circuit (including the fork gates, but not the wire transpositions) form the vertices of the circuit graph. We often view the circuit as cut into vertical *slices*. A slice can be any collection of gates and wires in the circuit such that no gate in a slice is an ancestor of another gate in the same slice, and no wire in a slice is an ancestor of another wire in the same slice (unless these two wires are an input wire and an output wire of a same gate). Two slices do not overlap, and every wire and every gate belongs to some slice. For more details on combinational circuits, see [12,34,45].

The *size* of a combinational circuit is defined to be the number of gates in the circuit, including forks and wire-swappings, as well as the input ports and the output ports. For a function $f : \{0,1\}^m \to \{0,1\}^n$, the *circuit complexity* (denoted $C(f)$) is the smallest size of any combinational circuit with input–output function f.

A cause of confusion about gates in a circuit is that gates of a certain type (e.g., and) are traditionally considered the same, no matter where they occur in the circuit. However, gates applied to different wires in a circuit are different functions; e.g., for the and gate, $(x_1, x_2, x_3) \mapsto (x_1 \wedge x_2, x_3)$ is a different function than $(x_1, x_2, x_3) \mapsto (x_1, x_2 \wedge x_3)$.

1.2. Computational asymmetry

Computational asymmetry is the core property of one-way functions. Below we will define computational asymmetry in a quantitative way, and in a later section we will relate it to the group-theoretic notion of distortion.

For the existence of one-way functions, it is mainly the relation between the circuit complexity $C(f)$ of f and the circuit complexity $C(f^{-1})$ of f^{-1} that matters, not the complexities of f and of f^{-1} themselves. Indeed, a classical *padding argument* can be used: If we add $C(f)$ "identity wires" to a circuit for f, then the resulting circuit has linear size as a function of its number of input wires; see Proposition 1.2 below. (An identity wire is a wire that goes directly from an input port to an output port, without being connected to any gate.)

In [12, p. 230] Boppana and Lagarias considered $\log C(f')/\log C(f)$ as a measure of one-wayness; here, f' denotes an inverse of f, i.e., any function such that $f \circ f' \circ f = f$. Massey and Hiltgen [20,26] introduced the phrases *complexity asymmetry* and *computational asymmetry* for injective functions, in reference to the situation where the circuit complexities $C(f)$ and $C(f^{-1})$ are very different. The concept of computational asymmetry can be generalized to arbitrary (non-injective) functions, with the meaning that for every inverse f' of f, $C(f)$ and $C(f')$ are very different.

In [26] Massey made the following observation. For any large-enough fixed m and for almost all permutations f of $\{0,1\}^m$, the circuit complexities $C(f)$ and $C(f^{-1})$ are very similar:

$$\frac{1}{10}C(f) \leqslant C\bigl(f^{-1}\bigr) \leqslant 10C(f).$$

Massey's proof is adapted from the Shannon lower bound [37] and the Lupanov upper bound [24] (see also [20,21,34]), from which it follows that almost all functions and almost all permutations (and their inverses) have circuit complexity close to the Shannon bounds. Massey's observation can be extended to the set of all functions $f:\{0,1\}^m \to \{0,1\}^n$, i.e., for almost all f and for every inverse f' of f, the complexities $C(f)$ and $C(f')$ are within constant factors of each other.

Hence, computationally asymmetric permutations are rare among the boolean permutations overall (and similarly for functions). This is an interesting fact about computational asymmetry, but by itself it does not imply anything about the existence or non-existence of one-way functions, not even heuristically. Indeed, Massey proved his linear relation $C(f) = \Theta(C(f'))$ in the situation where $C(f) = \Theta(2^m)$, and then uses the fact that the condition $C(f) = \Theta(2^m)$ holds for almost all boolean permutations and for almost all boolean functions. But there also exist functions with $C(f) = O(m^k)$, with k a small constant. In particular, one-way functions (if they exist) have small circuits; by definition, one-way functions violate the condition $C(f) = \Theta(2^m)$.

A well-known candidate for a one-way permutation is the following. For a large prime number p and a primitive root r modulo p, consider the map $x \in \{0,1,\dots,p-2\} \longmapsto r^x - 1 \in \{0,1,\dots,p-2\}$. This is a permutation whose inverse, known as the *discrete logarithm*, is believed to be difficult to compute.

Measuring computational asymmetry. Let $\mathfrak{S}_{\{0,1\}^m}$ denote the set of all permutations of $\{0,1\}^m$, i.e., $\mathfrak{S}_{\{0,1\}^m}$ is the symmetric group. We will measure the computational asymmetry of all permutations of $\{0,1\}^m$ (for all $m > 0$) by defining a *computational asymmetry function*, as follows. A function $a:\mathbb{N} \to \mathbb{N}$ is an upper bound on the computational asymmetry function iff for all $m > 0$ and all permutations f of $\{0,1\}^m$ we have: $C(f^{-1}) \leqslant a(C(f))$. The computational asymmetry function α of the boolean permutations is the least such function $a(.)$. Hence:

Definition 1.1. The computational asymmetry function α of the boolean permutations is defined as follows for all $s \in \mathbb{N}$: $\alpha(s) = \max\{C(f^{-1}):\ C(f) \leqslant s,\ f \in \mathfrak{S}_{\{0,1\}^m},\ m > 0\}$.

Note that in this definition we look at all combinational circuits, for all permutations in $\bigcup_{m>0} \mathfrak{S}_{\{0,1\}^m}$; we do not need to work with non-uniform or uniform families of circuits.

Computational asymmetry is closely related to one-wayness, as the next proposition shows.

Proposition 1.2.

(1) *For infinitely many n we have: There exists a permutation f_n of $\{0,1\}^n$ such that f_n is computed by a circuit of size $\leqslant 3n$, but f_n^{-1} has no circuit of size $< \alpha(n)$.*

(2) *Suppose that α is exponential, i.e., there is $k > 1$ such that for all n, $\alpha(n) \geqslant k^n$. Then $k \leqslant 2$, and there is a constant $c > 1$ such that we have: For every integer $n \geqslant 1$ there exists a permutation F_n of $\{0,1\}^n$ which is computed by a circuit of size $\leqslant cn$, but F_n^{-1} has no circuit of size $< k^n$.*

Proof. (1) By the definition of α, for every $m > 0$ there exists a permutation F of $\{0,1\}^m$ such that F is computed by a circuit of some size C_F, but F^{-1} has no circuit of size $< \alpha(C_F)$. Let $n = C_F$, and let us consider the function $f_n:\{0,1\}^{C_F} \to \{0,1\}^{C_F}$ defined by $f_n:(x,w) \longmapsto (F(x),w)$, for all $x \in \{0,1\}^m$ and $w \in \{0,1\}^{C_F - m}$.

Then $f_n(x,w)$ is computed by a circuit of size $C_F + 2(C_F - m)$; the term "$2(C_F - m)$" comes from counting the input–output wires of w. Hence f_n has a circuit of size $\leqslant 3n$. On the other hand, $(y,w) \longmapsto f_n^{-1}(y,w) = (F^{-1}(y),w)$ is not computed by any circuit of size $< \alpha(C_F)$, so f_n^{-1} has no circuit of size $< \alpha(n)$.

(2) For every $n \geqslant 1$ there exists a permutation F of $\{0,1\}^n$ such that F is computed by a circuit of some size C_F, and F^{-1} has a circuit of size $C_{F^{-1}} = \alpha(C_F) \geqslant k^{C_F}$; moreover, F^{-1} has no circuit of size $< \alpha(C_F)$. Thus, $k^{C_F} \leqslant C_{F^{-1}} \leqslant 2^n(1 + c_o \frac{\log n}{n})$, for some constant $c_o > 1$; the latter inequality comes from the Lupanov upper bound [24] (or see Theorem 2.13.2 in [34]). Hence, $k \leqslant 2$ and $n \leqslant C_F \leqslant \frac{1}{\log_2 k} n + c_1 \frac{\log n}{n}$, for some constant $c_1 > 0$. Hence, for all $n \geqslant 1$ there exists a permutation F of $\{0,1\}^n$ with circuit size $C_F \in [n, \frac{1}{\log_2 k} \cdot n + c_1 \cdot \frac{\log n}{n}]$, such that $C_{F^{-1}} = \alpha(C_F) \geqslant k^{C_F} \geqslant k^n$. $\square$

We will show later that the computational asymmetry function is closely related to the distortion of certain groups within certain monoids.

Remarks. Although in this paper we only use the computational asymmetry function of the boolean permutations, the concept can be generalized. Let $\mathsf{Inj}(\{0,1\}^m, \{0,1\}^n)$ denote the set of all *injective functions* $\{0,1\}^m \to \{0,1\}^n$. The computational asymmetry function α_{inj} of the injective boolean functions is defined by

$$\alpha_{\mathsf{inj}}(s) = \max\bigl\{C\bigl(f^{-1}\bigr):\ C(f) \leqslant s,\ f \in \mathsf{Inj}\bigl(\{0,1\}^m, \{0,1\}^n\bigr),\ m > 0,\ n > 0\bigr\}.$$

More generally, let $(\{0,1\}^n)^{\{0,1\}^m}$ denote the set of all *functions* $\{0,1\}^m \to \{0,1\}^n$. The computational asymmetry of all finite boolean functions is defined by

$$\alpha_{\mathsf{func}}(s) = \max\bigl\{C(f'):\ C(f) \leqslant s,\ ff'f = f,\ f, f' \in \bigl(\{0,1\}^n\bigr)^{\{0,1\}^m},\ n > 0,\ m > 0\bigr\}.$$

When we compare functions we will be mostly interested in their asymptotic growth pattern. Hence we will often use the big-O notation, and the following definitions.

By definition, two functions $f_1 : \mathbb{N} \to \mathbb{N}$ and $f_2 : \mathbb{N} \to \mathbb{N}$ are *linearly equivalent* iff there are constants $c_0, c_1, c_2 > 0$ such that for all $n \geqslant c_0$: $f_1(n) \leqslant c_1 f_2(c_1 n)$ and $f_2(n) \leqslant c_2 f_1(c_2 n)$. Notation: $f_1 \simeq_{\mathsf{lin}} f_2$.

Two functions f_1 and f_2 (from $\mathbb{N}$ to $\mathbb{N}$) are called *polynomially equivalent* iff there are constants $c_0, c_1, c_2, d, e > 0$ such that for all $n \geqslant c_0$: $f_1(n) \leqslant c_1 f_2(c_1 n^d)^d$ and $f_2(n) \leqslant c_2 f_1(c_2 n^e)^e$. Notation: $f_1 \simeq_{\mathsf{poly}} f_2$.

1.3. Word-length asymmetry

For a monoid M with generating set Γ, the *word-length* of an element $x \in M$ is defined to be the length of a shortest word over Γ, representing x; we use the notation $|x|_\Gamma$. We introduce an algebraic notion that looks very similar to computational asymmetry:

Definition 1.3. Let G be a group, let M be a monoid with generating set Γ (finite or infinite), and suppose $G \subseteq M$. The *word-length asymmetry function* of G within M (over Γ) is

$$\lambda(n) = \max\bigl\{\bigl|g^{-1}\bigr|_\Gamma:\ |g|_\Gamma \leqslant n,\ g \in G\bigr\}.$$

The word-length asymmetry function λ depends on G, M, Γ, and the embedding of G in M.

Consider the right Cayley graph of the monoid M with generating set Γ; its vertex set is M and the edges have the form $x \xrightarrow{\gamma} \gamma x$ (for $x \in M$, $\gamma \in \Gamma$). For $x, y \in M$, the *directed distance* $d(x, y)$ in the Cayley graph is the shortest length over all paths from x to y in the Cayley graph; if no path from x to y exists, the directed distance is infinite. By "path" we always mean directed path.

Lemma 1.4. *Under the above conditions on G, M, Γ, we have for every $g \in G$: $d(\mathbf{1}, g^{-1}) = d(g, \mathbf{1})$ and $d(\mathbf{1}, g) = d(g^{-1}, \mathbf{1})$.*

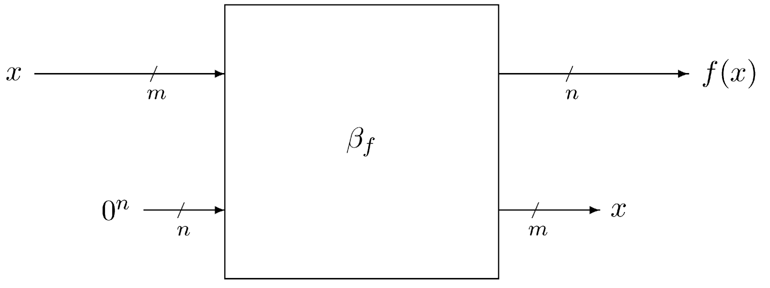

Fig. 1. Toffoli representation of the function f.

Proof. This is straightforward; see [10]. $\square$

Since $|g|_\Gamma$ is the distance $d(\mathbf{1}, g)$ in the graph of M, and since $|g^{-1}|_\Gamma = d(\mathbf{1}, g^{-1}) = d(g, \mathbf{1})$, the word-length asymmetry also measures the asymmetry of the directed distance, to or from the identity element $\mathbf{1}$ in the Cayley graph of M, restricted to vertices in the subgroup G.

For distances to or from the identity element of M it does not matter whether we consider the left Cayley graph or the right Cayley graph.

1.4. Computational asymmetry and reversible computing

Reversible computing deals with the following questions: If a function f is injective (or bijective) and computable, can f be computed in such a way that each elementary computation step is injective (respectively bijective)? And if such injective (or bijective) computations are possible, what is their complexity, compared to the usual (non-injective) complexity?

One of the main results is the following (Bennett's theorem [4,5], and earlier work of Lecerf [23]): Let f be an injective function, and assume f and f^{-1} are computable by deterministic Turing machines with time complexity $T_f(.)$, respectively $T_{f^{-1}}(.)$. Then f (and also f^{-1}) is computable by a *reversible Turing machine* (in which every transition is deterministic and injective) with time complexity $O(T_f + T_{f^{-1}})$. Note that only injectiveness (not bijectiveness) is used here.

Bennett's theorem has the following important consequence, which relates reversible computing to one-way functions: *Injective one-way functions exist iff there exist injective functions that have efficient traditional algorithms but that do not have efficient reversible algorithms.*

Toffoli representation. Remarkably, it is possible to "simulate" any function $f : \{0,1\}^m \to \{0,1\}^n$ (injective or not, one-way or not) by a bijective circuit; a circuit is called bijective iff the circuit is made from bijective gates. Here, bijective circuits will be built from the wire-swapping operations and the following bijective gates: not (negation), c-not (the Controlled Not, also called "Feynman gate") defined by $(x_1, x_2) \in \{0,1\}^2 \longmapsto (x_1, x_1 \oplus x_2) \in \{0,1\}^2$, and cc-not (the Doubly Controlled Not, also called "Toffoli gate") defined by $(x_1, x_2, x_3) \in \{0,1\}^3 \longmapsto (x_1, x_2, (x_1 \wedge x_2) \oplus x_3) \in \{0,1\}^3$.

Theorem 1.5. *(See Toffoli [42,43].) For every boolean function $f : \{0,1\}^m \to \{0,1\}^n$ there exists a bijective combinational circuit β_f (over the bijective gates* not, c-not, cc-not, *and wire-transpositions), with input–output function $\beta_f : x0^n \in \{0,1\}^{m+n} \longmapsto f(x)x \in \{0,1\}^{n+m}$.*

In other words, $f(x)$ consists of the projection onto the first n bits of $\beta_f(x0^n)$; equivalently, $f(.) = \mathsf{proj}_n \circ \beta_f \circ \mathsf{concat}_{0^n}(.)$, where proj_n projects a string of length $n+m$ to the first n bits, and concat_{0^n} concatenates 0^n to the right of a string. See Theorems 4.1, 5.3 and 5.4 of [42], and see Fig. 1.

The Toffoli representation contains two non-bijective actions: The projection at the output, and the forced setting of the value of some of the input wires.

Toffoli's proofs and constructions are based on truth tables, and he does not prove anything about the circuit size of β_f (counting the bijective gates), compared to the circuit size of f. The following gives a polynomial bound on the size of the bijective circuit, at the expense of a large number of input- and output-wires.

Theorem 1.6. *(See E. Fredkin, T. Toffoli [16].) For every boolean function $f:\{0,1\}^m \to \{0,1\}^n$ with circuit size $C(f)$ there exists a bijective combinational circuit B_f (over a bounded collection of bijective gates, e.g.,* not, c-not, cc-not, *and wire-transpositions), with input–output function*

$$B_f : x0^{n+C(f)} \in \{0,1\}^{m+n+C(f)} \longmapsto f(x)z(x) \in \{0,1\}^{m+n+C(f)}$$

for some $z(x) \in \{0,1\}^{m+C(f)}$. The number of gates of B_f has a polynomial upper bound in terms of $C(f)$.

If $g:\{0,1\}^m \to \{0,1\}^m$ is a permutation then there exists a bijective combinational circuit U_g (over bijective gates), with input–output function

$$U_g : x1^m0^{m+C} \in \{0,1\}^{3m+C} \longmapsto g(x)\overline{g(x)}x0^C \in \{0,1\}^{3m+C}$$

where $C = \max\{C(g), C(g^{-1})\}$, and $\overline{g(x)}$ is the bitwise complement of $g(x)$. The number of gates of U_g has a polynomial upper bound in terms of C.

Later we will introduce another reversible representation of boolean functions by bijective gates; we will need only one 0-wire, but the gates will be taken from the Thompson group $G_{2,1}$, i.e., we will also use non-length-preserving transformations of bitstrings (Theorems 4.1 and 4.2 below).

1.5. Distortion

We will prove later (Theorem 5.10) that computational asymmetry has a lot to do with distortion, a concept introduced into group theory by Gromov [18] and Farb [15]. Distortion is already known to have connections with isoperimetric functions (see [25,29,30]). A somewhat different problem about distortion (for finite metric spaces) was tackled by Bourgain [11].

We will use a slightly more general notion of distortion, based on (possibly directed) countably infinite rooted graphs, and their (directed) path metric.

A weighted directed graph is a structure (V, E, ω) where V is a set (called the vertex set), $E \subseteq V \times V$ (called the edge set), and $\omega : E \longmapsto \mathbb{R}_{>0}$ is a function (called the weight function); note that every edge has a strictly positive weight. It is sometimes convenient to define $\omega(u, v) = \infty$ when $(u, v) \in V \times V - E$. A path in (V, E) is a sequence of edges (u_i, v_i) $(1 \leqslant i \leqslant n)$ such that $u_{i+1} = v_i$ for all $i < n$, and such that all elements in $\{u_i: 1 \leqslant i \leqslant n\} \cup \{v_n\}$ are distinct; u_1 is called the start vertex of this path, and v_n is called the end vertex of this path; the sum of weights $\sum_{i=1}^{n} \omega(u_i, v_i)$ over the edges in the path is called the *length* of the path. Here we do not consider any paths with infinitely many edges; but we allow V and E to be countably infinite. A vertex w_2 is said to be reachable from a vertex w_1 in (V, E) iff there exists a path with start vertex w_1 and end vertex w_2. If w_2 is reachable from w_1 then the minimum length over all paths from w_1 to w_2 is called the *directed distance* from w_1 to w_2, denoted $d(w_1, w_2)$; since we only consider finite paths here, this minimum exists. If w_2 is not reachable from w_1 then we define $d(w_1, w_2)$ to be ∞. Clearly we have $w_1 = w_2$ iff $d(w_1, w_2) = 0$, and for all $u, v, w \in V$, $d(u, w) \leqslant d(u, v) + d(v, w)$. In a directed graph, the function $d(.,.)$ need not be symmetric. The function $d : V \times V \to \mathbb{R}_{\geqslant 0} \cup \{\infty\}$ is called the directed path metric of (V, E, ω). A rooted directed weighted graph is a structure (V, E, ω, r) where (V, E, ω) is a directed weighted graph, $r \in V$, and all vertices in V are reachable from r.

A set M with a function $d : M \times M \to \mathbb{R}_{\geqslant 0} \cup \{\infty\}$, satisfying the two axioms $w_1 = w_2$ iff $d(w_1, w_2) = 0$, and $d(u, w) \leqslant d(u, v) + d(v, w)$, will be called *directed metric space* (a.k.a. quasi-metric space).

Any subset G embedded in a directed metric space M becomes a directed metric space by using the directed distance of M. We call this the *directed distance on G inherited from M*.

If $G \subseteq V$ for a rooted directed weighted graph (V, E, ω, r), we consider the function $\ell : g \in G \longmapsto d(r, g) \in \mathbb{R}_{\geqslant 0}$, which we call the *directed length function* on G inherited from (V, E, ω, r). (The value ∞ will not appear here since all of G is reachable from r.)

We now define distortion in a very general way. Intuitively, distortion in a set is a quantitative comparison between two (directed) length functions that are defined on the same set.

Definition 1.7. Let G be a set, and let ℓ_1 and ℓ_2 be two functions $G \to \mathbb{R}_{\geqslant 0}$. The **distortion** of ℓ_1 with respect to ℓ_2 is the function $\delta_{\ell_1,\ell_2} : \mathbb{R}_{\geqslant 0} \to \mathbb{R}_{\geqslant 0}$ defined by

$$\delta_{\ell_1,\ell_2}(n) = \max\{\ell_1(g):\ g \in G,\ \ell_2(g) \leqslant n\}.$$

We will also use the notation $\delta[\ell_1, \ell_2](.)$ for $\delta_{\ell_1,\ell_2}(.)$. When we consider a distortion $\delta_{\ell_1,\ell_2}(.)$ we often assume that $\ell_2 \leqslant \ell_1$ or $\ell_2 \leqslant O(\ell_1)$; this insures that the distortion is at least linear, i.e., $\delta_{\ell_1,\ell_2}(n) \geqslant cn$, for some constant $c > 0$. We will only deal with functions obtained from the lengths of finite paths in countable directed graphs, so in that case the functions ℓ_i are discrete, and the distortion function exists. The next lemma generalizes the distortion result of Proposition 4.2 of [15].

Lemma 1.8. *Let G be a set and consider three functions $\ell_3, \ell_2, \ell_1 : G \to \mathbb{R}_{\geqslant 0}$ such that $\ell_1(.) \geqslant \ell_2(.) \geqslant \ell_3(.)$. Then the corresponding distortions satisfy*: $\delta_{\ell_1,\ell_3}(.) \leqslant \delta_{\ell_1,\ell_2} \circ \delta_{\ell_2,\ell_3}(.)$.

Proof. The inequalities $\ell_1(.) \geqslant \ell_2(.) \geqslant \ell_3(.)$ guarantee that the three distortions δ_{ℓ_1,ℓ_3}, δ_{ℓ_1,ℓ_2}, and δ_{ℓ_2,ℓ_3} are at least as large as the identity map. By definition,

$$\begin{aligned}
\delta_{\ell_1,\ell_2}\big(\delta_{\ell_2,\ell_3}(n)\big) &= \max\{\ell_1(x):\ x \in G,\ \ell_2(x) \leqslant \delta_{\ell_2,\ell_3}(n)\} \\
&= \max\{\ell_1(x):\ x \in G,\ \ell_2(x) \leqslant \max\{\ell_2(z):\ z \in G,\ \ell_3(z) \leqslant n\}\} \\
&= \max\{\ell_1(x):\ x \in G,\ (\exists z \in G)\ \big(\ell_2(x) \leqslant \ell_2(z) \text{ and } \ell_3(z) \leqslant n\big)\} \\
&\geqslant \max\{\ell_1(x):\ x \in G,\ \ell_3(x) \leqslant n\} = \delta_{\ell_1,\ell_3}(n).
\end{aligned}$$

The last inequality follows from the fact that if $\ell_3(x) \leqslant n$ then for some z (e.g., for $z = x$): $\ell_2(x) \leqslant \ell_2(z)$ and $\ell_3(z) \leqslant n$. □

Examples of distortion. Distortion and asymmetry are unifying concepts that apply to many fields.

1. Gromov distortion. Let G be a subgroup of a group H, with generating sets Γ_G, respectively Γ_H, such that $\Gamma_G \subseteq \Gamma_H$, and such that $\Gamma_G = \Gamma_G^{-1}$ and $\Gamma_H = \Gamma_H^{-1}$. This determines a Cayley graph for G and a Cayley graph for H. Now we have two distance functions on G, one obtained from the Cayley graph of G itself (based on Γ_G), and the other inherited from the embedding of G in H. See [11,15,18].

The Gromov distortion function is a natural measure of the difficulty of the *generalized word problem*. A very important case is when both Γ_G and Γ_H are finite. Here are some results for that case:

Theorem of Ol'shanskii and Sapir [30] (making precise and proving the outline on pp. 66–67 in [18]): All *Dehn functions* of finitely presented groups (and "approximately all" *time complexity* functions of *non-deterministic* Turing machines) are Gromov distortion functions of finitely generated subgroups of $\mathrm{FG}_2 \times \mathrm{FG}_2$; here, FG_2 denotes the 2-generated free group. Moreover, in [6] it was proved that $\mathrm{FG}_2 \times \mathrm{FG}_2$ is embeddable with *linear* distortion in the Thompson group $G_{2,1}$. So the theorem of Ol'shanskii and Sapir also holds for the finitely generated subgroups of $G_{2,1}$.

Actually, Gromov [18] and Bourgain [11] defined the distortion to be $\frac{1}{n} \cdot \max\{|g|_{\Gamma_G}:\ |g|_{\Gamma_H} \leqslant n,\ g \in G\}$, i.e., they use an extra factor $\frac{1}{n}$. However, the connections between distortion, the generalized word problem, and complexity (as we just saw, and will further see in the present paper) are more direct without the factor $\frac{1}{n}$.

2. Bourgain's distortion theorem. Given a finite metric space G with n elements, the aim is to find embeddings of G into a finite-dimensional euclidean space. The two distances of G are its given distance and the inherited euclidean distance. In this problem the goal is to have small distortion, as a function of the cardinality of G, while also keeping the dimension of the euclidean space small. Bourgain [11] found a bound $O(n \log n)$ for the distortion (or "$O(\log n)$" in Bourgain's and Gromov's terminology). This is an important result. See also [2,3,22].

3. Generator distortion. A variant of Gromov's distortion is obtained when $G = H$, but $\Gamma_G \subsetneqq \Gamma_H$. So here we look at the distorting effect of a change of generators in a given group. When Γ_G and Γ_H are both finite the generator distortion is linear; however, when Γ_G is finite and Γ_H is infinite the distortion becomes interesting. E.g., for the Thompson group $G_{2,1}$ let us take Γ_G to be any finite generating set, and for Γ_H let us take $\Gamma_G \cup \{\tau_{i,j}\colon 1 \leqslant i < j\}$; here $\tau_{i,j}$ is the position transposition defined earlier. Then the generator distortion is exponential (see [7]). Also, the word problem of $G_{2,1}$ over any finite generating set Γ_G is in P, but the word problem of $G_{2,1}$ over $\Gamma_G \cup \{\tau_{i,j}\colon 1 \leqslant i < j\}$ is coNP-complete (see [7] and [8]).

4. Monoids and directed distance. Gromov's distortion and the generator distortion can be generalized to monoids. We repeat what we said about Gromov distortion, but G and H are now monoids, and Γ_G, respectively Γ_H, are monoid generating sets which are used to define monoid Cayley graphs. We will use the left Cayley graphs. We assume $\Gamma_G \subseteq \Gamma_H$. In each Cayley graph there is a directed distance, defined by the lengths of directed paths. The monoid G now has two directed distance functions, the distance in the Cayley graph of G itself, and the directed distance that G inherits from its embedding into the Cayley graph of H. We denote the word-length of $g \in G$ over Γ_G by $|g|_G$; this is the minimum length of all words over Γ_G that represent g; it is also the length of a shortest path from the identity to g in the Cayley graph of G. Similarly, we denote the word-length of $h \in H$ over Γ_H by $|h|_H$. The definition of the distortion becomes: $\delta(n) = \max\{|g|_G\colon\ g \in G,\ \ |g|_H \leqslant n\}$.

5. Schreier graphs. Let G, H, and F be groups, where F is a subgroup of H. Let Γ_H be a generating set of Γ_H, and assume $\Gamma_H = \Gamma_H^{-1}$. We can define the Schreier left coset graph of H/F over the generating set Γ_H, and the distance function $d_{H/F}(.,.)$ in this coset graph. By definition, this Schreier graph has vertex set H/F (i.e., the left cosets, of the form $h \cdot F$ with $h \in H$), and it has directed edges of the form $h \cdot F \xrightarrow{\gamma} \gamma h \cdot F$, for $h \in H$, $\gamma \in \Gamma_H$. The graph is symmetric; for every edge as above there is an opposite edge $\gamma h \cdot F \xrightarrow{\gamma^{-1}} h \cdot F$. Because of symmetry the Schreier graph has a (symmetric) distance function based on path length, $d_{H/F}(.,.) : H/F \times H/F \to \mathbb{N}$.

Next, assume that G is embedded into H/F by some injective function $G \hookrightarrow H/F$. Such an embedding happens, e.g., if G and F are subgroups of H such that $G \cap F = \{\mathbf{1}\}$. Indeed, in that case each coset in H/F contains at most one element of G (since $g_1 F = g_2 F$ implies $g_2^{-1} g_1 \in F \cap G = \{\mathbf{1}\}$).

The group G now inherits a distance function from the path length in the Schreier graph of H/F. Comparing this distance with other distances in G leads to distortion functions. E.g., if the group G is also embedded in a monoid M with monoid generating set Γ_M, this leads to the following distortion function: $\delta_G(n) = \max\{d_{H/F}(F, gF)\colon\ g \in G,\ \ |g|_M \leqslant n\}$.

It will turn out that for appropriate choices of G, F, H, Γ_H, and Γ_M, this last distortion is polynomially related to the computational asymmetry function α of boolean permutations (Theorem 5.10).

6. Asymmetry functions. We already saw the computational asymmetry function of combinational circuits, and the word-length asymmetry function of a group embedded in a monoid. More generally, in any quasi-metric space (S, d), where $d(.,.)$ is a directed distance function, an asymmetry function $A : \mathbb{R}_{\geqslant 0} \to \mathbb{R}_{\geqslant 0}$ can be defined by $A(n) = \max\{d(x_2, x_1)\colon\ x_1, x_2 \in S,\ \ d(x_1, x_2) \leqslant n\}$.

This asymmetry function can also be viewed as the distortion of d^{rev} with respect to d in S; here d^{rev} denotes the reverse directed distance, defined by $d^{\mathrm{rev}}(x_1, x_2) = d(x_2, x_1)$.

7. Other distortions.

- Distortion can compare lengths of proofs (or lengths of expressions) in various, more or less powerful proof systems (respectively description languages). Distortion can also compare the duration

of computations or of rewriting processes in various models of computation. Hence, many (perhaps all) notions of complexity are examples of distortion. Distortion is an algebraic or geometric representation (or cause) of complexity.
– Instead of length and distance, other measures (e.g., volumes in higher dimension, energy, action, entropy, etc.) could be used.

1.6. Thompson–Higman groups and monoids

The Thompson groups, introduced by Richard J. Thompson [27,40,41], are finitely presented infinite groups that act as bijections between certain subsets of $\{0,1\}^*$. So, the elements of the Thompson groups are transformations of bitstrings, and hence they are related to input–output maps of combinational circuits. In this subsection we define the Thompson group $G_{2,1}$ (also known as "V"), as well as its generalization (by Graham Higman [19]) to the group $G_{k,1}$ that partially acts on A^*, for any finite alphabet A of size $k \geqslant 2$. We will follow the presentation of [6] (see also [8] and [7]); another reference is [35], which is also based on string transformations but with a different terminology; the classical references [13,19,27,40,41] do not describe the Thompson groups by transformations of finite strings. Because of our interest in strings and in circuits, we also use generalizations of the Thompson groups to monoids, as introduced in [9].

Some preliminary definitions, all fairly standard, are needed in order to define the Thompson–Higman group $G_{k,1}$. First, we pick any alphabet A of cardinality $|A| = k$. By A^* we denote the set of all finite words (or "strings") over A; the empty word ε is also in A^*. We denote the length of $w \in A^*$ by $|w|$ and we let A^n denote the set of words of length n. We denote the concatenation of two words $u, v \in A^*$ by uv or by $u \cdot v$; the concatenation of two subsets $B, C \subseteq A^*$ is defined by $BC = \{uv\colon\ u \in B,\ v \in C\}$. A *right ideal* of A^* is a subset $R \subseteq A^*$ such that $RA^* \subseteq R$. A generating set of a right ideal R is, by definition, a set C such that R is equal to the intersection of all right ideals that contain C; equivalently, C generates R (as a right ideal) iff $R = CA^*$. A right ideal R is called *essential* iff R has a non-empty intersection with every right ideal of A^*. For $u, v \in A^*$, we call u a *prefix* of v iff there exists $z \in A^*$ such that $uz = v$. A *prefix code* is a subset $C \subseteq A^*$ such that no element of C is a prefix of another element of C. A prefix code C over A is maximal iff C is not a strict subset of any other prefix code over A. It is easy to prove that a right ideal R has a unique minimal (under inclusion) generating set C_R, and that C_R is a prefix code; moreover, C_R is a maximal prefix code iff R is an essential right ideal.

For a partial function $f : A^* \to A^*$ we denote the domain by $\mathsf{Dom}(f)$ and the image (range) by $\mathsf{Im}(f)$. A restriction of f is any partial function $f_1 : A^* \to A^*$ such that $\mathsf{Dom}(f_1) \subseteq \mathsf{Dom}(f)$, and such that $f_1(x) = f(x)$ for all $x \in \mathsf{Dom}(f_1)$. An extension of f is any partial function of which f is a restriction. An *isomorphism* between right ideals R_1, R_2 of A^* is a bijection $\varphi : R_1 \to R_2$ such that for all $r_1 \in R_1$ and all $z \in A^*$: $\varphi(r_1 z) = \varphi(r_1) \cdot z$. The isomorphism φ is uniquely determined by a bijection between the prefix codes that minimally generate R_1, respectively R_2. One can prove [6,35,41] that every isomorphism φ between essential right ideals has a *unique maximal extension* (within the category of isomorphisms between essential right ideals of A^*); we denote this unique maximal extension by $\max(\varphi)$.

Now, finally, we define the *Thompson–Higman group* $G_{k,1}$: It consists of all maximally extended isomorphisms between finitely generated essential right ideals of A^*. The multiplication consists of composition followed by maximum extension: $\varphi \cdot \psi = \max(\varphi \circ \psi)$. Note that $G_{k,1}$ acts partially and faithfully on A^* on the *left*.

Every element $\varphi \in G_{k,1}$ can be described by a bijection between two finite maximal prefix codes; this bijection can be described concretely by a finite function *table*. When φ is described by a maximally extended isomorphism between essential right ideals, $\varphi : R_1 \to R_2$, we call the minimum generating set of R_1 the domain code of φ, and denote it by $\mathsf{domC}(\varphi)$; similarly, the minimum generating set of R_2 is called the image code of φ, denoted by $\mathsf{imC}(\varphi)$. See the beginning of the proof of Theorem 4.1 for examples of tables of elements of $G_{2,1}$. More examples appear in [6].

Thompson and Higman proved that $G_{k,1}$ is finitely presented. Also, when k is even $G_{k,1}$ is a simple group, and when k is odd $G_{k,1}$ has a simple normal subgroup of index 2. In [6] it was proved that the word problem of $G_{k,1}$ over any finite generating set is in P (in fact, more strongly, in the parallel complexity class AC_1). In [7,8] it was proved that the word problem of $G_{k,1}$ over $\Gamma \cup \{\tau_{i,j}\colon 1 \leqslant i < j\}$ is coNP-complete, where Γ is any finite generating set of $G_{k,1}$, and where $\tau_{i,j}$ is the position transposition introduced in Subsection 1.1.

Because of connections with circuits we consider the subgroup $lp\,G_{k,1}$ of all length-preserving elements of $G_{k,1}$; more precisely, $lp\,G_{k,1} = \{\varphi \in G_{k,1}\colon \forall x \in \mathsf{Dom}(\varphi),\ |x| = |\varphi(x)|\}$. See [8] for a study of $lp\,G_{k,1}$ and some of its properties. In particular, it was proved that $lp\,G_{k,1}$ is a direct limit of finite alternating groups, and that $lp\,G_{2,1}$ is generated by the set $\{N, C, T\} \cup \{\tau_{i,i+1}\colon 1 \leqslant i\}$, where $N : x_1 w \mapsto \bar{x}_1 w$, $C : x_1 x_2 w \mapsto x_1 (x_2 \oplus x_1) w$, and $T : x_1 x_2 x_3 w \mapsto x_1 x_2 (x_3 \oplus (x_2 \wedge x_1)) w$ (for $x_1, x_2, x_3 \in \{0,1\}$ and $w \in \{0,1\}^*$). Thus (recalling Subsection 1.4), N, C, T are the not, c-not, cc-not gates, applied to the first (left-most) bits of a binary string. It is known that the gates not, c-not, cc-not, together with the wire-swappings, form a complete set of gates for bijective circuits (see [16,38,42]); hence, $lp\,G_{2,1}$ is closely related to the field of reversible computing.

It is natural to generalize the bijections between finite maximal prefix codes to functions between finite prefix codes. Following [9] we will define below the *Thompson–Higman monoids* $M_{k,1}$. First, some preliminary definitions. A *right-ideal homomorphism* of A^* is a total function $\varphi : R_1 \to A^*$ such that R_1 is a right ideal, and such that for all $r_1 \in R_1$ and all $z \in A^*$: $\varphi(r_1 z) = \varphi(r_1) \cdot z$. It is easy to prove that $\mathsf{Im}(\varphi)$ is then also a right ideal of A^*. From now on we will write a right-ideal homomorphism as a total surjective function $\varphi : R_1 \to R_2$, where both R_1 and R_2 are right ideals. The homomorphism φ is uniquely determined by a total surjective function $f : P_1 \to S_2$, with $P_1, S_2 \subset A^*$ where P_1 is the prefix code (not necessarily maximal) that generates R_1 as a right ideal, and where S_2 is a set (not necessarily a prefix code) that generates R_2 as a right ideal; f can be described by a finite function table.

For two sets X, Y, we say that X and Y "intersect" iff $X \cap Y \neq \varnothing$. We say that a right ideal R_1' is *essential in* a right ideal R_1 iff R_1' intersects every right ideal that R_1 intersects. An *essential restriction* of a right-ideal homomorphism $\varphi : R_1 \to R_2$ is a right-ideal homomorphism $\Phi : R_1' \to R_2'$ such that R_1' is essential in R_1, and for all $x_1' \in R_1'$: $\varphi(x_1') = \Phi(x_1')$. In that case we also say that φ is an *essential extension* of Φ. If Φ is an essential restriction of φ then $R_2' = \mathsf{Im}(\Phi)$ will automatically be essential in $R_2 = \mathsf{Im}(\varphi)$. Indeed, if I is any non-empty right subideal of R_1 then $I \cap R_1' \neq \varnothing$, hence $\varnothing \neq \Phi(I \cap R_1') \subseteq \Phi(I) \cap \Phi(R_1') = \Phi(I) \cap R_2'$; moreover, any non-empty right subideal J of R_2 is of the form $J = \Phi(I)$, where $I = \Phi^{-1}(J)$ is a non-empty right subideal of R_1; hence, for any non-empty right subideal J of R_2, $\varnothing \neq J \cap R_2'$.

The free monoid A^* can be pictured by its right Cayley graph, which is easily seen to be the infinite regular k-ary tree with vertex set A^* and edge set $\{(v, va)\colon v \in A^*,\ a \in A\}$. We simply call this the *tree of* A^*. It is a directed, rooted tree, with all paths directed away from the root ε (the empty word); by "path" we will always mean a directed path. Many of the previously defined concepts can be reformulated more intuitively in the context of the tree of A^*: A word v is a prefix of a word w iff v is an ancestor of w in the tree. A set P is a prefix code iff no two elements of P are on a common path. A set R is a right ideal iff any path that starts in R has all its vertices in R. The prefix code that generates R consists of the elements of R that are maximal (within R) in the prefix order, i.e., maximally close (along paths) to the root ε. A finitely generated right ideal R is essential iff every infinite path eventually reaches R (and then stays in it from there on). Similarly, a finite prefix code P is maximal iff any infinite path starting at the root eventually intersects P. For two finitely generated right ideals R', R with $R' \subset R$ we have: R' is essential in R iff any infinite path starting in R eventually reaches R' (and then stays in it from there on).

Assume now that a total order $a_1 < a_2 < \cdots < a_k$ has been chosen for the alphabet A; this means that the tree of A^* is now an *oriented* rooted tree, i.e., the children of each vertex v have a total order $va_1 < va_2 < \cdots < va_k$. The following can be proved (see [9, Prop. 1.4(1)]): Φ is an essential restriction of φ iff Φ can be obtained from φ by starting from the table of φ and applying a finite number of *restriction steps* of the following form: "*replace* (x, y) *in a table by* $\{(xa_1, ya_1), \ldots, (xa_k, ya_k)\}$." In the tree of A^* this means that x and y are replaced by their children $xa_1, \ldots, xa_k$, respectively $ya_1, \ldots, ya_k$, paired according to the order on the children. One can also prove (see [9, Remark after Prop. 1.4]): Every right-ideal homomorphism φ with table $P \to S$ has an essential restriction φ' that has a table $P' \to Q'$ such that both P' and Q' are prefix codes. See [9] for examples of tables of elements of right-ideal homomorphisms.

An important fact is the following (see [9, Prop. 1.4(2)]): Every homomorphism between finitely generated right ideals of A^* has a *unique maximal* essential extension; we call it the maximum essential extension of Φ and denote it by $\max(\Phi)$.

Finally here is the definition of the *Thompson–Higman monoid*: $M_{k,1}$ consists of all maximum essential extensions of homomorphisms between finitely generated right ideals of A^*. The multiplication is composition followed by maximum essential extension.

One can prove the following, which implies associativity: For all right-ideal homomorphisms φ_1, φ_2: $\max(\varphi_2 \circ \varphi_1) = \max(\max(\varphi_2) \circ \varphi_1) = \max(\varphi_2 \circ \max(\varphi_1))$.

In [9] the following are proved about the Thompson–Higman monoid $M_{k,1}$:

- The Thompson–Higman group $G_{k,1}$ is the group of invertible elements of the monoid $M_{k,1}$.
- $M_{k,1}$ is finitely generated.
- The word problem of $M_{k,1}$ over any finite generating set is in P.
- The word problem of $M_{k,1}$ over a generating set $\Gamma \cup \{\tau_{i,j}\colon 1 \leqslant i < j\}$, where Γ is any finite generating set of $M_{k,1}$, is coNP-complete.

2. Boolean functions as elements of Thompson monoids

The input–output functions of combinational circuits map bitstrings of some fixed length to bitstrings of a fixed length (possibly different from the input length). In other words, circuits have input–output maps that are total functions of the form $f : \{0,1\}^m \to \{0,1\}^n$ for some $m, n > 0$. The Thompson–Higman monoid $M_{k,1}$ has an interesting submonoid that corresponds to fixed-length maps, defined as follows.

Definition 2.1 *(The submonoid $lep\, M_{k,1}$).* Let $\varphi : PA^* \to QA^*$ be a right-ideal homomorphism, where $P, Q \subset A^*$ are finite prefix codes, and where P is a *maximal* prefix code. Then φ is called *length-equality preserving* iff for all $x_1, x_2 \in \mathsf{Dom}(\varphi)$: $|x_1| = |x_2|$ implies $|\varphi(x_1)| = |\varphi(x_2)|$.

The submonoid $lep\, M_{k,1}$ of $M_{k,1}$ consists of those elements of $M_{k,1}$ that can be represented by length-equality preserving right-ideal homomorphisms.

It is easy to check that an essential restriction of an element of $lep\, M_{k,1}$ is again in $lep\, M_{k,1}$, so $lep\, M_{k,1}$ is well defined as a subset of $M_{k,1}$; moreover, one can easily check that $lep\, M_{k,1}$ is closed under composition, so $lep\, M_{k,1}$ is indeed a submonoid of $M_{k,1}$.

For $\varphi \in M_{k,1}$ we have $\varphi \in lep\, M_{k,1}$ iff there exist $m > 0$ and $n > 0$ such that $A^m \subset \mathsf{Dom}(\varphi)$ and $\varphi(A^m) \subseteq A^n$. So (by means of an essential restriction, if necessary), φ can be represented by a function table $A^m \to Q \subseteq A^n$ with a *fixed input length and a fixed output length* (but the input and output lengths can be different).

The motivation for studying the monoid $lep\, M_{k,1}$ is the following. Every boolean function $f : \{0,1\}^m \to \{0,1\}^n$ (for any $m, n > 0$) determines an element of $lep\, M_{k,1}$, and conversely, this element of $lep\, M_{k,1}$ determines f when restricted to $\{0,1\}^m$. By considering all boolean functions as elements of $lep\, M_{k,1}$ we gain the ability to compose arbitrary boolean functions, even if their domain and range "do not match." Moreover, in $lep\, M_{k,1}$ we are able to generate all boolean functions from gates by using ordinary *functional composition* (instead of graph-based circuit lay-outs). The following remains open:

Question. Is $lep\, M_{k,1}$ finitely generated?

However we can find nice infinite generating sets, in connection with circuits.

Proposition 2.2 *(Generators of $lep\, M_{k,1}$). The monoid $lep\, M_{k,1}$ has a generating set of the form $\Gamma \cup \{\tau_{i,i+1}\colon 1 \leqslant i\}$, for some finite subset $\Gamma \subset lep\, M_{k,1}$.*

Proof. We only prove the result for $k = 2$; a similar reasoning works for all k (using k-ary logic).

It is a classical fact that any function $f : \{0,1\}^m \to \{0,1\}^n$ can be implemented by a combinational circuit that uses copies of and, or, not, fork and wire-crossings. So all we need to do is to express these gates, at any place in the circuit, by a finite subset of $lep\, M_{2,1}$ and by position transpositions $\tau_{i,i+1}$. For each gate $g \in \{\text{and}, \text{or}\}$ we define an element $\gamma_g \in lep\, M_{k,1}$ by

$$\gamma_g : x_1 x_2 w \in \{0,1\}^m \ \longmapsto\ g(x_1, x_2) w \in \{0,1\}^{m-1}.$$

Similarly we define $\gamma_{\text{not}}, \gamma_{\text{fork}} \in lep\, M_{k,1}$ by

$$\gamma_{\text{not}} : x_1 w \in \{0,1\}^m \ \longmapsto\ \overline{x_1} w \in \{0,1\}^m,$$

$$\gamma_{\text{fork}} : x_1 w \in \{0,1\}^m \ \longmapsto\ x_1 x_1 w \in \{0,1\}^{m+1}.$$

For each $g \in \{\text{and}, \text{or}, \text{not}, \text{fork}\}$, γ_g transforms only the first one or two boolean variables, and leaves the other boolean variables unchanged. We also need to simulate the effect of a gate g on any variable x_i or pair of variables $x_i x_{i+1}$, i.e., we need to construct the map

$$u x_i x_{i+1} v \in \{0,1\}^m \ \longmapsto\ u g(x_i, x_{i+1}) v \in \{0,1\}^{m-1}$$

(and similarly in case where g is not or fork). For this, we apply wire-transpositions to move $x_i x_{i+1}$ to the wire-positions 1 and 2, then we apply γ_g, then we apply more wire-transpositions in order to move $g(x_1, x_2)$ back to position i. Thus the effect of any gate anywhere in the circuit can be expressed as a composition of γ_g and position transpositions in $\{\tau_{i,i+1}\colon 1 \leqslant i\}$. $\square$

Proposition 2.3 *(Change of generators of $lep\, M_{k,1}$). Let $\{\tau_{i,i+1}\colon 1 \leqslant i\}$ be denoted by τ. If $\Gamma, \Gamma' \subset lep\, M_{k,1}$ are two finite sets such that $\Gamma \cup \tau$ and $\Gamma' \cup \tau$ generate $lep\, M_{k,1}$, then the word-length over $\Gamma \cup \tau$ is linearly related to the word-length over $\Gamma' \cup \tau$. In other words, there are constants $c' \geqslant c \geqslant 1$ such that for all $m \in lep\, M_{k,1}$: $|m|_{\Gamma \cup \tau} \leqslant c \cdot |m|_{\Gamma' \cup \tau} \leqslant c' \cdot |m|_{\Gamma \cup \tau}$.*

Proof. Since Γ is finite, the elements of Γ can be expressed by a finite set of words of bounded length ($\leqslant c$) over $\Gamma' \cup \tau$. Thus, every word of length n over $\Gamma \cup \tau$ is equivalent to a word of length $\leqslant cn$ over $\Gamma' \cup \tau$. This proves the first inequality. A similar reasoning proves the second inequality. $\square$

Proposition 2.4 *(Circuit size vs. $lep\, M_{2,1}$ word-length). Let $\Gamma_{lep\, M_{2,1}} \cup \{\tau_{i,j}\colon 1 \leqslant i < j\}$ be a generating set of $lep\, M_{2,1}$ with $\Gamma_{lep\, M_{2,1}}$ finite. Let $f\colon \{0,1\}^m \to Q$ $(\subseteq \{0,1\}^n)$ be a function defining an element of $lep\, M_{2,1}$, and let $|f|_{lep\, M_{2,1}}$ the word-length of f over the generating set $\Gamma_{lep\, M_{2,1}} \cup \{\tau_{i,j}\colon 1 \leqslant i < j\}$. Let $|C_f|$ be the circuit size of f (using any finite universal set of gates and wire-swappings). Then $|f|_{lep\, M_{2,1}}$ and $|C_f|$ are linearly related. More precisely, for some constants $c_1 \geqslant c_o \geqslant 1$:*

$$|C_f| \leqslant c_o \cdot |f|_{lep\, M_{2,1}} \leqslant c_1 \cdot |C_f|.$$

Proof. For the proof we assume that the set of gates for circuits (not counting the wire-transpositions) is $\Gamma_{lep\, M_{2,1}}$. If we make a different choice for the universal set of gates for circuits, and a different choice for the finite portion $\Gamma_{lep\, M_{2,1}}$ of the generating set of $lep\, M_{2,1}$ then the inequalities remain the same, except for the constants c_1, c_o.

The inequality $|C_f| \leqslant |f|_{lep\, M_{2,1}}$ is obvious, since a word w over $\Gamma_{lep\, M_{2,1}} \cup \{\tau_{i,j}\colon 1 \leqslant i < j\}$ is automatically a circuit of size $|w|$.

For the other inequality, we want to simulate each gate of the circuit C_f by a word over $\Gamma_{lep\, M_{2,1}} \cup \{\tau_{i,j}\colon 1 \leqslant i < j\}$. The reasoning is the same for every gate, so let us just focus on an or gate. The essential difference between circuit gates and elements of $lep\, M_{2,1}$ is that in a circuit, a gate (with 2 input wires, for example) can be applied to any two wires in the circuit; on the other hand, the functions in $lep\, M_{2,1}$ are applied to the first few wires. However, the circuit gate or, applied to $(i, i+1)$

can be simulated by an element of $\Gamma_{lep\,M_{2,1}}$ and a few wire transpositions, since we have: $\mathsf{or}_{i,i+1}(.) = \gamma_{\mathsf{or}} \circ \tau_{2,i+1} \circ \tau_{1,i}(.)$.

The output wire of $\mathsf{or}_{i,i+1}(.)$ is wire number i, whereas the output wire of $\gamma_{\mathsf{or}} \circ \tau_{2,i+1} \circ \tau_{1,i}(.)$ is wire number 1. However, instead of permuting all the wires in order to place the output of $\gamma_{\mathsf{or}}\tau_{2,i+1}\tau_{1,i}(.)$ on wire i, we just leave the output of $\gamma_{\mathsf{or}}\tau_{2,i+1}\tau_{1,i}(.)$ on wire 1 for now. The simulation of the next gate will then use appropriate transpositions $\tau_{2,j} \cdot \tau_{1,k}$ for fetch the correct input wires for the next gate. Thus, each gate of C_f is simulated by one function in $\Gamma_{lep\,M_{2,1}}$ and a bounded number of wire-transpositions in $\{\tau_{i,j}\colon\ 1 \leqslant i < j\}$.

At the output end of the circuit, a permutation of the n output wires is needed in order to send the outputs to the correct wires; any permutation of n elements can be realized with $< n$ ($\leqslant |C_f|$) transpositions. (The inequality $n \leqslant |C_f|$ holds because since we count the output ports in the circuit size.) $\square$

Remark. The above proposition motivates our choice of generating set of the form $\Gamma \cup \{\tau_{i,j}\colon\ 1 \leqslant i < j\}$ (with Γ finite) for $lep\,M_{k,1}$; in particular, it motivates the inclusion of all the position transpositions $\tau_{i,j}$ in the generating set. The proposition also motivates the definition of word-length in which $\tau_{i,j}$ has word-length 1 for all $j > i \geqslant 1$.

Next we will study the **distortion** of $lep\,M_{k,1}$ in $M_{k,1}$. We first need some lemmas (that are not difficult; see [10]).

Lemma 2.5. *(See Lemma 3.3 in [6].) If P, Q, $R \subseteq A^*$ are such that $PA^* \cap QA^* = RA^*$ and R is a prefix code, then $R \subseteq P \cup Q$.*

Lemma 2.6. *Let P, $Q \subset A^*$ be finite prefix codes, and let $\theta : PA^* \to QA^*$ be a right-ideal homomorphism with domain PA^* and image QA^*. Let S be a prefix code with $S \subset QA^*$. Then $\theta^{-1}(S)$ is a prefix code and $\theta^{-1}(SA^*) = \theta^{-1}(S)A^*$.*

Notation. For a right-ideal homomorphism $\varphi : \mathsf{Dom}(\varphi) = PA^* \to \mathsf{Im}(\varphi) = QA^*$, where P, $Q \subset A^*$ are finite prefix codes, we define

$$\ell(\varphi) = \max\bigl\{|z|\colon\ z \in P \cup Q\bigr\}.$$

For any finite prefix code $C \subset A^*$ we define

$$\ell(C) = \max\bigl\{|z|\colon\ z \in C\bigr\}.$$

Lemma 2.7. *Let φ: $\mathsf{Dom}(\varphi) = PA^* \longrightarrow \mathsf{Im}(\varphi) = QA^*$ be a right-ideal homomorphism, where P and Q are finite prefix codes. Let $R \subset A^*$ be any finite prefix code. Then we have*:

(1) $\ell(\varphi^{-1}(R)) < \ell(\varphi) + \ell(R)$,
(2) $\ell(\varphi(R)) < \ell(\varphi) + \ell(R)$.

For any right-ideal homomorphisms φ_i (with $i = 1, \ldots, N$), the composite map $\varphi_N \circ \cdots \circ \varphi_1(.)$ is a right-ideal homomorphism. We say that right-ideal homomorphisms Φ_i (with $i = 1, \ldots, N$) are *directly composable* iff $\mathsf{Dom}(\Phi_{i+1}) = \mathsf{Im}(\Phi_i)$, for $i = 1, \ldots, N-1$. The next lemma shows that we can replace composition by direct composition.

Lemma 2.8. *Let $\varphi_i : \mathsf{Dom}(\varphi_i) = P_iA^* \longrightarrow \mathsf{Im}(\varphi_i) = Q_iA^*$ be a right-ideal homomorphism (for $i = 1, \ldots, N$), where P_i and Q_i are finite prefix codes. Then each φ_i has a (not necessarily essential) restriction to a right-ideal homomorphism Φ_i with the following properties*:

- $\Phi_N \circ \cdots \circ \Phi_1(.) = \varphi_N \circ \cdots \circ \varphi_1(.)$;

- $\mathsf{Dom}(\Phi_{i+1}) = \mathsf{Im}(\Phi_i)$, *for* $i = 1, \ldots, N-1$;
- $\ell(\Phi_i) \leqslant \sum_{j=1}^{N} \ell(\varphi_j)$ *for every* $i = 1, \ldots, N$.

Proof. We use induction on N. For $N = 1$ there is nothing to prove. So we let $N > 1$ and we assume that the lemma holds for $\varphi_i : P_i A^* \to Q_i A^*$ with $i = 2, \ldots, N$, i.e., we assume that each φ_i (for $i = 2, \ldots, N$) has a restriction $\varphi_i' : P_i' A^* \to Q_i' A^*$ such that $\varphi_N' \circ \cdots \circ \varphi_2' = \varphi_N \circ \cdots \circ \varphi_2$, $P_{i+1}' = Q_i'$ (for $i = 2, \ldots, N-1$), and $\ell(\varphi_i') \leqslant \sum_{j=2}^{N} \ell(\varphi_j)$ for every $i = 2, \ldots, N$. From $P_{i+1}' = Q_i'$ (for $i = 2, \ldots, N-1$) it follows that $\ell(\varphi_N' \circ \cdots \circ \varphi_2') \leqslant \max\{\ell(\varphi_i') \colon i = 2, \ldots, N\} \leqslant \sum_{j=2}^{N} \ell(\varphi_j)$.

Using the notation $\varphi_{[N,2]}'$ for $\varphi_N' \circ \cdots \circ \varphi_2'$ we have $\mathsf{Dom}(\varphi_{[N,2]}') = P_2 A^*$ and $\mathsf{Im}(\varphi_{[N,2]}') = Q_N A^*$. When we compose φ_1 and $\varphi_{[N,2]}'$ we obtain

$$\varphi_1^{-1}(Q_1 A^* \cap P_2 A^*) \xrightarrow{\Phi_1} Q_1 A^* \cap P_2 A^* \xrightarrow{\Phi_{[N,2]}'} \varphi_{[N,2]}'(Q_1 A^* \cap P_2 A^*).$$

In this diagram, Φ_1 is the restriction of φ_1 to the domain $\varphi_1^{-1}(Q_1 A^* \cap P_2 A^*)$ and image $Q_1 A^* \cap P_2 A^*$; and $\Phi_{[N,2]}'$ is the restriction of $\varphi_{[N,2]}'$ to the domain $Q_1 A^* \cap P_2 A^*$ and image $\varphi_{[N,2]}'(Q_1 A^* \cap P_2 A^*)$. Hence, $\Phi_{[N,2]}' \circ \Phi_1 = \varphi_{[N,2]}' \circ \varphi_1$, and $\mathsf{Dom}(\Phi_{[N,2]}') = \mathsf{Im}(\Phi_1)$ $(= Q_1 A^* \cap P_2 A^*)$. So Φ_1 and $\Phi_{[N,2]}'$ are directly composable.

By Lemma 2.5 there is a prefix code $S \subset A^*$ such that $SA^* = Q_1 A^* \cap P_2 A^*$ and $S \subseteq Q_1 \cup P_2$. Hence, $\ell(S) \leqslant \max\{\ell(Q_1), \ell(P_2)\} \leqslant \max\{\ell(\varphi_1), \ell(\varphi_2')\} \leqslant \max\{\ell(\varphi_1), \sum_{j=2}^{N} \ell(\varphi_j)\} \leqslant \sum_{j=1}^{N} \ell(\varphi_j)$.

It follows also that $\varphi_1^{-1}(Q_1 A^* \cap P_2 A^*) = \varphi_1^{-1}(SA^*) = \varphi_1^{-1}(S)A^*$ (the latter equality is from Lemma 2.6). Since $S \subseteq Q_1 \cup P_2$ implies $\varphi_1^{-1}(S) \subseteq \varphi_1^{-1}(Q_1) \cup \varphi_1^{-1}(P_2) = P_1 \cup \varphi_1^{-1}(P_2)$, we have $\ell(\varphi_1^{-1}(S)) \leqslant \max\{\ell(P_1), \ell(\varphi_1^{-1}(P_2))\}$. Obviously, $\ell(P_1) \leqslant \ell(\varphi_1)$. Moreover, by Lemma 2.7, $\ell(\varphi_1^{-1}(P_2)) \leqslant \ell(\varphi_1) + \ell(P_2)$. Since $\ell(P_2) \leqslant \ell(\varphi_2') \leqslant \sum_{j=2}^{N} \ell(\varphi_j)$ (the latter "$\leqslant$" by induction), we have $\ell(\varphi_1^{-1}(S)) \leqslant \ell(\varphi_1) + \sum_{j=2}^{N} \ell(\varphi_j) = \sum_{j=1}^{N} \ell(\varphi_j)$.

Since the domain code of Φ_1 is $\varphi_1^{-1}(S)$ and its image code is S, we conclude that $\ell(\Phi_1) \leqslant \sum_{j=1}^{N} \ell(\varphi_j)$.

Let us now consider any $\Phi_{[i,2]}'$, for $i = 1, \ldots, N$. By definition, $\Phi_{[i,2]}'$ is the restriction of $\varphi_i' \circ \cdots \circ \varphi_2'$ to the domain SA^*. So the domain code of $\Phi_{[i,2]}'$ is S, and we just proved that $\ell(S) \leqslant \sum_{j=1}^{N} \ell(\varphi_j)$. The image code of $\Phi_{[i,2]}'$ is $\varphi_i' \circ \cdots \circ \varphi_2'(S)$. Since $S \subseteq Q_1 \cup P_2$ we have

$$\varphi_i' \circ \cdots \circ \varphi_2'(S) \subseteq \varphi_i' \circ \cdots \circ \varphi_2'(Q_1) \cup \varphi_i' \circ \cdots \circ \varphi_2'(P_2) = \varphi_i' \circ \cdots \circ \varphi_2'(Q_1) \cup Q_i'.$$

Therefore: $\ell(\varphi_i' \circ \cdots \circ \varphi_2'(S)) \leqslant \max\{\ell(\varphi_i' \circ \cdots \circ \varphi_2'(Q_1)), \ell(Q_i')\}$.

We have $\ell(Q_i') \leqslant \ell(\varphi_i') \leqslant \sum_{j=2}^{N} \ell(\varphi_j)$ (the last "$\leqslant$" by induction).

By Lemma 2.7, $\ell(\varphi_i' \circ \cdots \circ \varphi_2'(Q_1)) \leqslant \ell(\varphi_i' \circ \cdots \circ \varphi_2') + \ell(Q_1) \leqslant \ell(\varphi_i' \circ \cdots \circ \varphi_2') + \ell(\varphi_1)$. And $\ell(\varphi_i' \circ \cdots \circ \varphi_2') \leqslant \max\{\ell(\varphi_j') \colon j = 2, \ldots, i\}$, because $\mathsf{Dom}(\varphi_{r+1}') = \mathsf{Im}(\varphi_r')$ for all $r = 2, \ldots, N-1$. And by induction, $\ell(\varphi_j') \leqslant \sum_{j=2}^{N} \ell(\varphi_j)$. Hence, $\ell(\varphi_i' \circ \cdots \circ \varphi_2'(Q_1)) \leqslant \sum_{j=1}^{N} \ell(\varphi_j)$.

Thus, $\ell(\Phi_{[i,2]}') \leqslant \sum_{j=1}^{N} \ell(\varphi_j)$ for every $i = 2, \ldots, N$.

Finally, we factor $\Phi_{[N,2]}'$ as $\Phi_{[N,2]}' = \Phi_N \circ \cdots \circ \Phi_2$, where Φ_i (for $i = 2, \ldots, N$) is defined to be the restriction of φ_i' to the domain $\varphi_{i-1}' \circ \cdots \circ \varphi_2'(SA^*)$ $(= \Phi_{[i-1,2]}'(SA^*))$. Since $\mathsf{Dom}(\varphi_{r+1}') = \mathsf{Im}(\varphi_r')$ (for all $r = 2, \ldots, N-1$), the domain of φ_i' is equal to the image of $\varphi_{i-1}' \circ \cdots \circ \varphi_2'$. So, the domain code of Φ_i is $\varphi_{i-1}' \circ \cdots \circ \varphi_2'(S)$, and its image code is $\varphi_i' \circ \varphi_{i-1}' \circ \cdots \circ \varphi_2'(S)$. Since we already proved that $\ell(\varphi_i' \circ \cdots \circ \varphi_2'(S)) \leqslant \sum_{j=1}^{N} \ell(\varphi_j)$ (for all i), it follows that $\ell(\Phi_i) \leqslant \sum_{j=1}^{N} \ell(\varphi_j)$. $\square$

In the next theorem we show that the distortion of *lep* $M_{k,1}$ in $M_{k,1}$ is at most quadratic (over the generators considered so far, which include the bit position transpositions). Combined with Proposition 2.4, this means the following:

Assume circuits are built with gates that are not constrained to have fixed-length inputs and outputs, but assume the input–output function has fixed-length inputs and outputs. Then the resulting circuits are not much more compact than conventional circuits, built from gates that have fixed-length inputs and outputs (*we gain at most a square-root in size*).

Theorem 2.9 *(Distortion of lep $M_{k,1}$ in $M_{k,1}$). The word-length* (*or Cayley graph*) *distortion of lep $M_{k,1}$ in $M_{k,1}$ has a quadratic upper bound*; *in other words, for all $x \in$ lep $M_{k,1}$:*

$$|x|_{lep\, M_{k,1}} \leqslant c \cdot \left(|x|_{M_{k,1}}\right)^2$$

where $c \geqslant 1$ is a constant. Here the generating sets used are $\Gamma_{M_{k,1}} \cup \{\tau_{i,j}\colon 1 \leqslant i < j\}$ for $M_{k,1}$, and $\Gamma_{lep\, M_{k,1}} \cup \{\tau_{i,j}\colon 1 \leqslant i < j\}$ for lep $M_{k,1}$, where $\Gamma_{M_{k,1}}$ and $\Gamma_{lep\, M_{k,1}}$ are finite. By $|x|_{M_{k,1}}$ and $|x|_{lep\, M_{k,1}}$ we denote the word-length of x over $\Gamma_{M_{k,1}} \cup \{\tau_{i,j}\colon 1 \leqslant i < j\}$, respectively $\Gamma_{lep\, M_{k,1}} \cup \{\tau_{i,j}\colon 1 \leqslant i < j\}$.

Proof. We only prove the result for $k = 2$; a similar proof applies for any k. We abbreviate the set $\{\tau_{i,j}\colon 1 \leqslant i < j\}$ by τ. The choice of the finite sets $\Gamma_{M_{k,1}}$ and $\Gamma_{lep\, M_{k,1}}$ does not matter (it only affects the constant c in the theorem). By Corollary 3.6 in [9] we can choose $\Gamma_{M_{k,1}}$ so that each $\gamma \in \Gamma_{M_{k,1}}$ satisfies the following (recall that $\ell(S)$ denotes the length of the longest words in a set S):

$$\ell\big(\mathsf{domC}(\gamma) \cup \mathsf{imC}(\gamma)\big) \leqslant 2, \quad \text{and}$$

$$\big|\left|\gamma(x)\right| - |x|\big| \leqslant 1 \quad \text{for all } x \in \mathsf{Dom}(\gamma).$$

Let $\varphi \in$ *lep* $M_{k,1}$, and let $w = \alpha_N \dots \alpha_1$ be a shortest word over the generating set $\Gamma_{M_{k,1}} \cup \tau$ of $M_{k,1}$, representing φ. So $N = |\varphi|_{M_{k,1}}$. We restrict each partial function α_i to a partial function α_i' such that $\mathsf{imC}(\alpha_i') = \mathsf{domC}(\alpha_{i+1}')$ for $i = 1, \dots, N-1$, according to Lemma 2.8. Hence, $\alpha_N \circ \dots \circ \alpha_1(.) = \alpha_N' \circ \dots \circ \alpha_1'(.)$, and $\ell(\alpha_i') \leqslant \sum_{j=1}^N \ell(\alpha_j)$ for every $i = 1, \dots, N$. Then $\alpha_N \circ \dots \circ \alpha_1(.)$ is a function $\{0,1\}^m\{0,1\}^* \to Q\,\{0,1\}^*$, representing φ, and we will identify $\alpha_N \circ \dots \circ \alpha_1(.)$ with φ. It follows that $\mathsf{domC}(\alpha_1') = \mathsf{domC}(\varphi) = \{0,1\}^m$, and $\mathsf{imC}(\alpha_N') = \mathsf{imC}(\varphi) = Q \subseteq \{0,1\}^n$. More generally, it follows that $\mathsf{imC}(\alpha_i' \circ \dots \circ \alpha_1') = \mathsf{imC}(\alpha_i')$, and $\mathsf{domC}(\alpha_N' \circ \dots \circ \alpha_i') = \mathsf{domC}(\alpha_i')$.

Since $\ell(\alpha_i') \leqslant \sum_{j=1}^N \ell(\alpha_j)$, and $\ell(\alpha_j) \leqslant 2$ for all j, we have for every $i = 1, \dots, N$: $\ell(\alpha_i') \leqslant 2N$.

From here on we will simply denote $\ell(\alpha_i')$ by ℓ_i. Now, we will replace each $\alpha_i' \in M_{k,1}$ by $\beta_i \in$ *lep* $M_{k,1}$, such that $\mathsf{domC}(\beta_i) = \{0,1\}^{\ell_i}$, and $\mathsf{imC}(\beta_i) \subseteq \{0,1\}^{\ell_{i+1}}$; so β_i is length-equality preserving. This will be done by artificially lengthening those words in $\mathsf{domC}(\alpha_i')$ that have length $< \ell_i$ and those words in $\mathsf{imC}(\alpha_i')$ that have length $< \ell_{i+1}$. Moreover, we make β_i defined on *all* of $\{0,1\}^{\ell_i}$. In detail, β_i is defined as follows:

- If $\ell_i \leqslant \ell_{i+1}$:

$$\beta_i(uz) = vz0^{\ell_{i+1} - \ell_i - |v| + |u|} \quad \text{for all } u \in \mathsf{domC}\big(\alpha_i'\big), \text{ and } z \in \{0,1\}^{\ell_i - |u|}; \text{ here } v = \alpha_i'(u);$$

$$\beta_i(x) = x0^{\ell_{i+1} - \ell_i} \quad \text{for all } x \notin \mathsf{Dom}\big(\alpha_i'\big), \ |x| = \ell_i.$$

- If $\ell_i > \ell_{i+1}$:

$$\beta_i(uz_1z_2) = vz_1 \quad \text{for all } u \in \mathsf{domC}\big(\alpha_i'\big) \text{ and all } z_1, z_2 \in \{0,1\}^* \text{ with}$$

$$|z_1| = \ell_{i+1} - |v|, \ |z_2| = \ell_i - \ell_{i+1} + |v| - |u|; \text{ here, } v = \alpha_i'(u);$$

$$\beta_i(x_1x_2) = x_1 \quad \text{for all } x_1, x_2 \in \{0,1\}^* \text{ such that } x_1x_2 \notin \mathsf{Dom}\big(\alpha_i'\big), \text{ with}$$

$$|x_1| = \ell_{i+1}, \ |x_2| = \ell_i - \ell_{i+1}.$$

Claim. $\beta_N \circ \cdots \circ \beta_1(.) = \varphi$.

Proof. We observe first that $\mathsf{domC}(\beta_1) = \mathsf{domC}(\alpha'_1)$ ($= \mathsf{domC}(\varphi) = \{0,1\}^m$). Next, assume by induction that for every $x \in \{0,1\}^m$: $\alpha'_{i-1} \circ \cdots \circ \alpha'_1(x) = u$ is a prefix of $\beta_{i-1} \circ \cdots \circ \beta_1(x) = uz$. Then $\beta_i(uz) = vz0^{\ell_{i+1}-\ell_i-|v|+|u|}$ (if $\ell_i \leqslant \ell_{i+1}$); or $\beta_i(uz) = vz_1$ (if $\ell_i \geqslant \ell_{i+1}$, with $|z_1| = \ell_{i+1} - |v|$ and $z = z_1z_2$). In either case we find that $\alpha'_i(\alpha'_{i-1} \circ \cdots \circ \alpha'_1(x)) = v$ is a prefix of $\beta_i(\beta_{i-1} \circ \cdots \circ \beta_1(x)) = \beta_i(uz)$.

Hence, when $i = N$ we obtain for any $x \in \{0,1\}^m$: $\beta_N \circ \cdots \circ \beta_1(x) = ys$ is a prefix of $\alpha'_N \circ \cdots \circ \alpha'_1(x) = \varphi(x) = y$ for some y and s with $|ys| = \ell_N = n$. Since $y \in \mathsf{imC}(\varphi) \subseteq \{0,1\}^n$ we conclude that s is empty, hence $\beta_N \circ \cdots \circ \beta_1(x) = \alpha'_N \circ \cdots \circ \alpha'_1(x)$. $\square$

At this point we have expressed φ as a product of N elements $\beta_i \in lep\,M_{k,1}$, where $N = |\varphi|_{M_{k,1}}$. We now want to find the word-length of each β_i over $\Gamma_{lep\,M_{k,1}} \cup \tau$, in order to find an upper bound on the total word-length of φ over $\Gamma_{lep\,M_{k,1}} \cup \tau$. As we saw above, $\ell_i \leqslant 2N$ for every $i = 1, \ldots, N$.

We examine each generator in $\Gamma_{M_{k,1}} \cup \tau$.

If $\alpha_i \in \tau$ then $\beta_i \in \tau$, so in this case $|\beta_i|_{lep\,M_{k,1}} = 1$.

Suppose now that $\alpha_i \in \Gamma_{M_{k,1}}$. By Proposition 2.4 it is sufficient to construct a circuit that computes β_i; the circuit can then be immediately translated into a word over $\Gamma_{lep\,M_{k,1}} \cup \tau$ with linear increase in length.

Since $\mathsf{domC}(\alpha_i) \subseteq \{0,1\}^{\leqslant 2}$, we can restrict α_i so that its domain code becomes a subset of $\{0,1\}^2$; next, we extend α_i to a map α''_i that acts as the identity map on $\{0,1\}^2$ where α_i was undefined. The image code of α''_i is a subset of $\{0,1\}^{\leqslant 3}$. In order to compute β_i we first introduce a circuit $C(\alpha''_i)$ that computes α''_i. A difficulty here is that α''_i does not produce fixed-length outputs in general, whereas $C(\alpha''_i)$ has to work with fixed-length inputs and outputs; so the output of $C(\alpha''_i)$ represents the output of α''_i indirectly, as follows:

The circuit $C(\alpha''_i)$ has two input bits $u = u_1u_2 \in \{0,1\}^2$, and 5 output bits: First there are 3 output bits $0^{3-|v|}v \in \{0,1\}^3$, where $v = \alpha''_i(u)$; second, there are two more output bits, $c_1c_2 \in \{0,1\}^2$, defined by $c_1c_2 = \mathsf{bin}(3 - |v|)$ (the binary representation of the non-negative integer $3 - |v|$). Hence, $c_1c_2 = 00$ if $|v| = 3$, $c_1c_2 = 01$ if $|v| = 2$, $c_1c_2 = 10$ if $|v| = 1$; since $|v| > 0$, the value $c_1c_2 = 11$ will not occur. Thus $c_1c_20^{3-|v|}v$ contains the same information as v, but has the advantage of having a fixed length (always 5). The circuit $C(\alpha''_i)$ can be built with a small constant number of and, or, not, fork gates, and we will not need to know the details.

We now build a circuit for β_i.

- Circuit for β_i if $\ell_i \leqslant \ell_{i+1}$:

On input $uz \in \{0,1\}^{\ell_i}$ (with $u \in \{0,1\}^2$), we want to produce the output $vz0^{\ell_{i+1}-\ell_i-|v|+|u|}$, where $v = \alpha''_i(u)$.

We first apply the circuit $C(\alpha''_i)$, thus obtaining $c_1c_20^{3-|v|}vz$. Then we apply two fork operations (always to the last bit in z) to produce $c_1c_20^{3-|v|}vzbb$, where b is the last bit of z. Applying a negation to the first b and an and operation, we obtain $c_1c_20^{3-|v|}vz0$. Applying $\ell_{i+1} - \ell_i - 1$ more fork operations to the last 0 yields $c_1c_20^{3-|v|}vz0^{\ell_{i+1}-\ell_i-1}$.

Next, we want to move $0^{3-|v|}$ to the right of the output, in order to obtain $c_1c_2vz0^{3-|v|+\ell_{i+1}-\ell_i-1}$. For this effect we introduce a *controlled cycle*. Let $\kappa : x_1x_2x_3 \in \{0,1\}^3 \longmapsto x_3x_1x_2$ be the usual cyclic permutations of 3 bit positions. The controlled cycle acts as the identity map when $c_1c_2 = 00$ or 11, $\tau_{1,2}$ when $c_1c_2 = 01$, and κ when $c_1c_2 = 10$. More precisely,

$$\kappa_c : c_1c_2x_1x_2x_3 \in \{0,1\}^5 \longmapsto \begin{cases} c_1c_2x_1x_2x_3 & \text{if } c_1c_2 = 00 \text{ or } 11, \\ c_1c_2x_2x_1x_3 & \text{if } c_1c_2 = 01, \\ c_1c_2x_3x_1x_2 & \text{if } c_1c_2 = 10. \end{cases}$$

We apply ℓ_i copies of $\kappa_c(c_1, c_2, ., ., .)$ (all controlled by the same value of c_1c_2) to $0^{3-|v|}vz$. The first $\kappa_c(c_1, c_2, ., ., .)$ is applied to the 3 bits $0^{3-|v|}v$, producing 3 bits $y_1y_2y_3$; the second $\kappa_c(c_1, c_2, ., ., .)$ is applied to y_2y_3 and the first bit of z, producing 3 bits $y'_1y'_2y'_3$; the third $\kappa_c(c_1, c_2, ., ., .)$ is applied to

$y_2'y_3'$ and the second bit of z, etc. So, each one of the ℓ_i copies of κ_c acts one bit further down than the previous copy of κ_c. This will yield $c_1c_2vz0^{3-|v|+\ell_{i+1}-\ell_i-1}$. Finally, to make c_1c_2 disappear, we apply two fork operations to c_1, then a negation and an and, to make a 0 appear. We combine this 0 with c_1 and c_2 by and gates, thus transforming $0c_1c_2$ into 0. Finally, an or operation between this 0 and the first bit of v makes this 0 disappear.

The number of gates used to compute β_i is $O(\ell_{i+1}+\ell_i)$, which is $\leqslant O(N)$.

- Circuit for β_i if $\ell_i > \ell_{i+1}$:

On input $uz \in \{0,1\}^{\ell_i}$ (with $u \in \{0,1\}^2$), we want to produce the output vz_1, where $v = \alpha_i''(u)$.

We first apply the circuit $C(\alpha_i'')$, which yields the output $c_1c_20^{3-|v|}vz$. Now we want to erase the $\ell_i - \ell_{i+1} + 1$ last bits of z. For this we apply two fork operations to the last bit of z (let us call it b), then a negation and an and, to make a 0 appear. We combine this 0 with the last $\ell_i - \ell_{i+1}$ bits of z, using that many and gates, turning all these bits into a single 0; finally, an or operation between this 0 and the bit of the remainder of z makes this 0 disappear. At this point, the output is $c_1c_20^{3-|v|}vZ_1$, where Z_1 is the prefix of length $\ell_{i+1}-1$ of z.

Next, we apply $O(\ell_{i+1})$ position transpositions to Z_1 in order move the two last bits of Z_1 to the front of Z_1. Let b_1b_2 be the last two bits of Z_1; so, $Z_1 = z_0b_1b_2$ (where z_0 is the prefix of length $\ell_{i+1}-3$ of z); at this point, the output of the circuit is $c_1c_20^{3-|v|}vb_1b_2z_0$.

We now introduce a fixed small circuit with 7 input bits and 5 output bits, defined by the following input–output map:

$$\omega_c : c_1c_2x_1x_2x_3b_1b_2 \in \{0,1\}^7 \longmapsto \begin{cases} c_1c_2x_1x_2x_3 & \text{if } c_1c_2 = 00 \text{ or } 11, \\ c_1c_2x_1x_2b_1 & \text{if } c_1c_2 = 01, \\ c_1c_2x_3b_1b_2 & \text{if } c_1c_2 = 10. \end{cases}$$

When this map is applied to $c_1c_20^{3-|v|}vb_1b_2$ the output is therefore given by

$$\omega_c : c_1c_20^{3-|v|}vb_1b_2 \in \{0,1\}^7 \longmapsto \begin{cases} c_1c_2v & \text{if } |v| = 3, \\ c_1c_2vb_1 & \text{if } |v| = 2, \\ c_1c_2vb_1b_2 & \text{if } |v| = 1. \end{cases}$$

A circuit for ω_c can be built with a small fixed number of and, or, not, fork gates, and we will not need to know the details.

After applying ω_c to $c_1c_20^{3-|v|}vb_1b_2z_0$ the output has length $\ell_{i+1}+2$; the "+2" comes from c_1c_2. The output is $c_1c_2vz_0$, or $c_1c_2vb_1z_0$, or $c_1c_2vb_1b_2z_0$, depending on whether $|v| = 3, 2$, or 1.

We need to move b_1b_2 or b_1 (or nothing) back to the right-most positions of z_0. We do this by applying ℓ_{i+1} copies of the *controlled cycle* $\kappa_c(c_1, c_2, .,.,.)$ (all copies controlled by the same value of c_1c_2). We proceed in the same way as when we used κ_c in the previous case, and we obtain the output $c_1c_2vz_0$ (if $|v| = 3$), or $c_1c_2vz_0b_1$ (if $|v| = 2$), or $c_1c_2vz_0b_1b_2$ (if $|v| = 1$).

Finally, we erase c_1c_2 in the same way as in the previous case, thus obtaining the final output. The number of gates used to compute β_i is $O(\ell_{i+1}+\ell_i) \leqslant O(N)$.

This completes the construction of a circuit for β_i. Through this circuit, $\beta_i : \{0,1\}^{\ell_i} \to \{0,1\}^{\ell_{i+1}}$ is expressed as a word over the generating set $\Gamma_{lep\,M_{k,1}} \cup \tau$, of length $\leqslant O(\ell_{i+1}+\ell_i) \leqslant O(N)$.

Since we have described φ as a product of $N = |\varphi|_{M_{k,1}}$ elements $\beta_i \in lep\,M_{k,1}$, each of word-length $O(N)$, we conclude that φ has word-length $\leqslant O(N^2)$ over the generating set $\Gamma_{lep\,M_{k,1}} \cup \tau$ of $lep\,M_{k,1}$. □

Question. Does the distortion of $lep\,M_{k,1}$ in $M_{k,1}$ (over the generators of Theorem 2.9) have an upper bound that is less than quadratic?

3. Word-length asymmetry vs. computational asymmetry

Proposition 3.1. *The word-length asymmetry function λ of the Thompson group $lp\,G_{2,1}$ within the Thompson monoid $lep\,M_{2,1}$ is linearly equivalent to the computational asymmetry function α:*

$$\alpha \simeq_{\mathrm{lin}} \lambda.$$

Here the generating set used for $lep\,M_{2,1}$ is $\Gamma_{lep\,M_{2,1}} \cup \{\tau_{i,j}\colon 0 \leqslant i < j\}$, where $\Gamma_{lep\,M_{2,1}}$ is finite. The gates used for circuits are any finite universal set of gates, together with the wire-swapping operations $\{\tau_{i,j}\colon 0 \leqslant i < j\}$.

We can choose $\Gamma_{lep\,M_{2,1}}$ to consist exactly of the gates used in the circuits; then $\alpha = \lambda$.

Proof. For any $g \in lp\,G_{2,1}$ we have

$$C\big(g^{-1}\big) \leqslant c_0 \cdot \big|g^{-1}\big|_{lep\,M_{2,1}} \leqslant c_0 \cdot \lambda\big(|g|_{lep\,M_{2,1}}\big) \leqslant c_0 \cdot \lambda\big(c_1 \cdot C(g)\big).$$

The first and last "$\leqslant$" come from Proposition 2.4 (since $lp\,G_{2,1} \subset lep\,M_{2,1}$), and the middle "$\leqslant$" comes from the definition of λ; c_0 and c_1 are positive constants. Hence,

$$\alpha(n) \leqslant c_0 \cdot \lambda(c_1 n) \quad \text{for all } n.$$

In a very similar way we prove that $\lambda(n) \leqslant c'_0 \cdot \alpha(c'_1 n)$ for some positive constants c'_0, c'_1. $\square$

Proposition 3.2. *The word-length asymmetry function $\lambda_{M_{2,1}}$ of the Thompson group $lp\,G_{2,1}$ within the Thompson monoid $M_{2,1}$ is polynomially equivalent to the word-length asymmetry function $\lambda_{lep\,M_{2,1}}$ of $lp\,G_{2,1}$ within the Thompson monoid $lep\,M_{2,1}$. More precisely we have for all n:*

$$\lambda_{M_{2,1}}(n) \leqslant c_0 \cdot \lambda_{lep\,M_{2,1}}\big(c_1 n^2\big),$$
$$\lambda_{lep\,M_{2,1}}(n) \leqslant c'_0 \cdot \big(\lambda_{M_{2,1}}\big(c'_1 n\big)\big)^2,$$

where c_0, c_1, c'_0, c'_1 are positive constants. Here the generating set used for $lep\,M_{2,1}$ is $\Gamma_{lep\,M_{2,1}} \cup \{\tau_{i,j}\colon 0 \leqslant i < j\}$, where $\Gamma_{lep\,M_{2,1}}$ is finite. The generating set used for $M_{2,1}$ is $\Gamma_{M_{2,1}} \cup \{\tau_{i,j}\colon 0 \leqslant i < j\}$, where $\Gamma_{M_{2,1}}$ is a finite generating set of $M_{2,1}$.

Proof. For any $g \in lp\,G_{2,1}$ we have

$$\big|g^{-1}\big|_{M_{2,1}} \leqslant c_0 \cdot \big|g^{-1}\big|_{lep\,M_{2,1}} \leqslant c_0 \cdot \lambda_{lep\,M_{2,1}}\big(|g|_{lep\,M_{2,1}}\big) \leqslant c_0 \cdot \lambda_{lep\,M_{2,1}}\big(c_1 \cdot |g|^2_{M_{2,1}}\big).$$

The first "$\leqslant$" holds because $lp\,G_{2,1} \subset lep\,M_{2,1} \subset M_{2,1}$ and because of the choice of the generating sets. The second "$\leqslant$" holds by the definition of $\lambda_{lep\,M_{2,1}}$. The third "$\leqslant$" comes from the quadratic distortion of $lep\,M_{2,1}$ in $M_{2,1}$ (Theorem 2.9). For the same reasons we also have the following:

$$\big|g^{-1}\big|_{lep\,M_{2,1}} \leqslant c'_0 \cdot \big|g^{-1}\big|^2_{M_{2,1}} \leqslant c'_0 \cdot \big(\lambda_{M_{2,1}}\big(|g|_{M_{2,1}}\big)\big)^2 \leqslant c'_0 \cdot \big(\lambda_{M_{2,1}}\big(c_1 \cdot |g|_{lep\,M_{2,1}}\big)\big)^2$$

where c'_0, c'_1 are positive constants. $\square$

4. Reversible representation over the Thompson groups

Theorems 4.1 and 4.2 below introduce a representation of elements of the Thompson monoid $lep\,M_{2,1}$ by elements of the Thompson group $G_{2,1}$, in analogy with the Toffoli representation (Theorem 1.5 above), and the Fredkin representation (Theorem 1.6 above). Our representation preserves complexity, up to a polynomial change, and uses only *one* constant-0 input. Note that although the functions and circuits considered here use fixed-length inputs and outputs, the representation is over the Thompson group $G_{2,1}$, which includes functions with variable-length inputs and outputs.

In the theorem below, $\Gamma_{G_{2,1}}$ is any finite generating set of $G_{2,1}$. We denote the length of a word w by $|w|$, and we denote the size of a circuit C by $|C|$. The gates and, or, not will also be denoted respectively by $\wedge$, $\vee$, $\neg$. We distinguish between a word W_f (over a generating set of $G_{2,1}$) and the element w_f of $G_{2,1}$ represented by W_f.

Theorem 4.1 *(Representation of boolean functions by the Thompson group). Let $f : \{0,1\}^m \to \{0,1\}^n$ be any total function and let C_f be a minimum-size circuit (made of $\wedge$, $\vee$, $\neg$, fork-gates and wire-swappings $\tau_{i,j}$) that computes f. Then there exists a word W_f over the generating set $\Gamma_{G_{2,1}} \cup \{\tau_{i,i+1}\colon 1 \leqslant i\}$ of $G_{2,1}$ such that:*

- *For all $x \in \{0,1\}^m$: $w_f(0x) = 0f(x)x$, where w_f is the element of $G_{2,1}$ represented by W_f.*
- *The length of the word W_f is bounded by $|W_f| \leqslant O(|C_f|^4)$.*
- *The largest subscript of any transposition $\tau_{i,i+1}$ occurring in W_f has an upper bound $\leqslant |C_f|^2 + 2$.*

Proof. Wire-swappings in circuits are represented by the position transpositions $\tau_{i,i+1} \in G_{2,1}$. The gates not, or, and and of circuits are represented by the following elements of $G_{2,1}$:

$$\varphi_\neg = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad \varphi_\vee = \begin{bmatrix} 0x_1x_2 & 1x_1x_2 \\ (x_1 \vee x_2)x_1x_2 & (\overline{x_1 \vee x_2})x_1x_2 \end{bmatrix}, \qquad \varphi_\wedge = \begin{bmatrix} 0x_1x_2 & 1x_1x_2 \\ (x_1 \wedge x_2)x_1x_2 & (\overline{x_1 \wedge x_2})x_1x_2 \end{bmatrix},$$

where x_1, x_2 range over $\{0,1\}$. Hence the domain and image codes of $\varphi_\vee$ and $\varphi_\wedge$ are all equal to $\{0,1\}^3$.

To represent fork we use the following element, in which we recognize $\sigma \in F_{2,1}$, one of the commonly used generators of the Thompson group $F_{2,1}$:

$$\sigma = \begin{bmatrix} 0 & 10 & 11 \\ 00 & 01 & 1 \end{bmatrix} = \begin{bmatrix} 00 & 01 & 10 & 11 \\ 000 & 001 & 01 & 1 \end{bmatrix}.$$

Note that σ agrees with fork only on input 0, but that is all we will need. By its very essence, the forking operation cannot be represented by a length-equality preserving element of $G_{2,1}$, because $G_{2,1} \cap lep\,M_{2,1} = lp\,G_{2,1}$ (the group of length-preserving elements of $G_{2,1}$). A small remark: In [6–8], what we call "σ" here, was called "σ^{-1}."

We will occasionally use the wire-swapping $\tau_{i,j}$ $(1 \leqslant i < j)$; note that $\tau_{i,j}$ can be expressed in terms of transpositions of neighboring wires as follows:

$$\tau_{i,j}(.) = \tau_{i,i+1}\tau_{i+1,i+2}\ldots\tau_{j-2,j-1}\tau_{j-1,j}\tau_{j-2,j-1}\ldots\tau_{i+1,i+2}\tau_{i,i+1}(.)$$

so the word-length of $\tau_{i,j}$ over $\{\tau_{\ell,\ell+1}\colon 1 \leqslant \ell\}$ is $\leqslant 2(j-i)-1$.

For $x = x_1 \ldots x_m \in \{0,1\}^m$ and $f(x) = y = y_1 \ldots y_n \in \{0,1\}^n$, we will construct a word W_f over the generators $\Gamma_{G_{2,1}} \cup \{\tau_{i,i+1}\colon 1 \leqslant i\}$ of $G_{2,1}$, such that W_f defines the map $w_f(.) : 0x \mapsto 0f(x)x$.

The circuit C_f is partitioned into *slices* c_ℓ $(\ell = 1, \ldots, L)$. Two gates g_1 and g_2 are in the same slice iff the length of the *longest* path from g_1 to any input port is the same as the length of the longest path from g_2 to any input port. We assume that C_f is *strictly layered*, i.e., each gate in slice c_ℓ only has in-wires coming from slice $c_{\ell-1}$, and out-wires going toward slice $c_{\ell+1}$, for all ℓ. To make a circuit C

strictly layered we need to add at most $|C|^2$ identity-gates (see p. 52 in [7]). The input–output map of slice c_ℓ has the form

$$c_\ell(.) : y^{(\ell-1)} = y_1^{(\ell-1)} \ldots y_{n_{\ell-1}}^{(\ell-1)} \in \{0,1\}^{n_{\ell-1}} \longmapsto y^{(\ell)} = y_1^{(\ell)} \ldots y_{n_\ell}^{(\ell)} \in \{0,1\}^{n_\ell}.$$

Then $y^{(0)} = x$ and $y^{(L)} = y$, where $x \in \{0,1\}^m$ is the input and $y \in \{0,1\}^n$ is the output of C_f. Each slice is a circuit of depth 1.

Before studying in more detail how C_f is built from slices, let us see how a slice is built from gates (inductively, one gate at a time).

Let C be a depth-1 circuit with $k+1$ gates, obtained by adding one gate to a depth-1 circuit K with k gates. Let $K(.) : x_1 \ldots x_m \longmapsto y_1 \ldots y_n$ be the input–output map of the circuit K. Assume by induction that K is represented by a word W_K over the generating set $\Gamma_{G_{2,1}} \cup \{\tau_{i,i+1}\colon 1 \leqslant i\}$ of $G_{2,1}$. The input–output map of W_K is, by induction hypothesis,

$$w_K(.) : 0x_1 \ldots x_m \longmapsto 0y_1 \ldots y_n x_1 \ldots x_m.$$

The word W_C that represents C over $G_{2,1}$ is obtained as follows from W_K; there are several cases, depending on the gate that is added to K to obtain C.

Case 1. An identity-gate (or a not-gate) is added to K to form C, i.e.,

$$C(.) : x_1 \ldots x_m x_{m+1} \longmapsto y_1 \ldots y_n x_{m+1}$$
$$\big(\text{or, } C(.) : x_1 \ldots x_m x_{m+1} \longmapsto y_1 \ldots y_n \bar{x}_{m+1}\big).$$

Then W_C is given by

$$\begin{aligned} w_C : 0x_1x_2 \ldots x_m x_{m+1} &\overset{\sigma}{\longmapsto} 00x_1x_2 \ldots x_m x_{m+1} \overset{\tau_{3,m+3}}{\longmapsto} 00x_{m+1}x_2 \ldots x_m x_1 \\ &\overset{\varphi_\vee}{\longmapsto} x_{m+1}0x_{m+1}x_2 \ldots x_m x_1 \overset{\tau_{3,m+3}}{\longmapsto} x_{m+1}0x_1x_2 \ldots x_m x_{m+1} \overset{\pi}{\longmapsto} 0x_1x_2 \ldots x_m x_{m+1}x_{m+1} \\ &\overset{w_K}{\longmapsto} 0y_1 \ldots y_n x_1 \ldots x_m x_{m+1} x_{m+1} \overset{\pi'}{\longmapsto} 0y_1 \ldots y_n x_{m+1} x_1 \ldots x_m x_{m+1}, \end{aligned}$$

where $\pi(.) = \tau_{m+1,m+2} \ldots \tau_{2,3}\tau_{1,2}(.)$ shifts x_{m+1} from position 1 to position $m+2$, while shifting $0x_1 \ldots x_m$ one position to the left; and $\pi'(.) = \tau_{m+2,m+3} \ldots \tau_{n+m+1,n+m+2}\tau_{n+m+2,n+m+3}(.)$ shifts x_{m+1} from position $n+m+3$ to position $n+2$, while shifting $x_1 \ldots x_m$ one position to the right.

So, $W_C = \pi' W_K \pi \tau_{3,m+3} \varphi_\vee \tau_{3,m+3} \sigma$, noting that functions act on the left. Thus, $|W_C| = |W_K| + m + n + 5$ if we use all of $\{\tau_{i,j}\colon 1 \leqslant i < j\}$ in the generating set; over $\{\tau_{i,i+1}\colon 1 \leqslant i\}$, $\tau_{3,m+3}$ has length $\leqslant 2m-1$, hence $|W_C| \leqslant 3m+n+4$. If we denote the maximum index in the transpositions occurring in W_C by J_C then we have $J_C = \max\{J_K, n+m+3\}$.

In case a not-gate is added (instead of an identity-gate), $\varphi_\vee$ is replaced by $\varphi_\neg\varphi_\vee$ in W_C, and the result is similar.

Case 2. An and-gate (or an or-gate) is added to K to form C, i.e.,

$$C(.) : x_1 \ldots x_m x_{m+1} x_{m+2} \longmapsto y_1 \ldots y_n (x_{m+1} \wedge x_{m+2})$$
$$\big(\text{or, } C(.) : x_1 \ldots x_m x_{m+1} x_{m+2} \longmapsto y_1 \ldots y_n (x_{m+1} \vee x_{m+2})\big).$$

Then W_C is given by

$$w_C : 0x_1x_2\ldots x_mx_{m+1}x_{m+2} \stackrel{\sigma}{\longmapsto} 00x_1x_2\ldots x_mx_{m+1}x_{m+2}$$

$$\stackrel{\tau_{2,m+3}}{\longmapsto}\stackrel{\tau_{3,m+4}}{\longmapsto} 0x_{m+1}x_{m+2}x_2\ldots x_m0x_1 \stackrel{\varphi_\wedge}{\longmapsto} (x_{m+1}\wedge x_{m+2})x_{m+1}x_{m+2}x_2\ldots x_m0x_1$$

$$\stackrel{\tau_{2,m+3}}{\longmapsto}\stackrel{\tau_{3,m+4}}{\longmapsto} (x_{m+1}\wedge x_{m+2})0x_1x_2\ldots x_mx_{m+1}x_{m+2} \stackrel{\pi}{\longmapsto} 0x_1x_2\ldots x_m(x_{m+1}\wedge x_{m+2})x_{m+1}x_{m+2}$$

$$\stackrel{w_K}{\longmapsto} 0y_1\ldots y_nx_1x_2\ldots x_m(x_{m+1}\wedge x_{m+2})x_{m+1}x_{m+2}$$

$$\stackrel{\pi'}{\longmapsto} 0y_1\ldots y_n(x_{m+1}\wedge x_{m+2})x_1x_2\ldots x_mx_{m+1}x_{m+2},$$

where $\pi = \tau_{m+1,m+2}\ldots\tau_{2,3}\tau_{1,2}$ shifts $(x_{m+1}\wedge x_{m+2})$ from position 1 to position $m+2$, while shifting $0x_1x_2\ldots x_m$ one position to the left; and $\pi' = \tau_{m+2,m+3}\ldots\tau_{m+n+1,m+n+2}$ shifts $(x_{m+1}\wedge x_{m+2})$ from position $n+m+2$ to position $m+2$, while shifting $x_1\ldots x_m$ one position to the right.

So, $W_C = \pi' W_K \pi \tau_{3,m+4}\tau_{2,m+3}\varphi_\wedge\tau_{3,m+4}\tau_{2,m+3}\sigma$, hence $|W_C| = |W_K|+n+m+7$ if all of $\{\tau_{i,j}\colon 1\leqslant i<j\}$ is used in the generating set; over $\{\tau_{i,i+1}\colon 1\leqslant i\}$, $\tau_{3,m+4}$ and $\tau_{2,m+3}$ have length $\leqslant 2(m+1)-1$, so $|W_C|\leqslant |W_K|+5m+n+9$. Moreover, $J_C = \max\{J_K, m+n+2\}$.

Case 3. A fork-gate is added to K to form C, i.e.,

$$C(.) : x_1\ldots x_mx_{m+1} \longmapsto y_1\ldots y_nx_{m+1}x_{m+1}.$$

Then W_C is given by

$$w_C : 0x_1x_2\ldots x_mx_{m+1} \stackrel{\sigma^2}{\longmapsto} 000x_1x_2\ldots x_mx_{m+1} \stackrel{\tau_{3,m+4}}{\longmapsto} 00x_{m+1}x_1x_2\ldots x_m0$$

$$\stackrel{\varphi_\vee}{\longmapsto} x_{m+1}0x_{m+1}x_1x_2\ldots x_m0 \stackrel{\tau_{1,m+4}}{\longmapsto} 00x_{m+1}x_1x_2\ldots x_mx_{m+1} \stackrel{\varphi_\vee}{\longmapsto} x_{m+1}0x_{m+1}x_1x_2\ldots x_mx_{m+1}$$

$$\stackrel{\pi}{\longmapsto} 0x_1x_2\ldots x_mx_{m+1}x_{m+1}x_{m+1} \stackrel{w_K}{\longmapsto} 0y_1\ldots y_nx_1x_2\ldots x_mx_{m+1}x_{m+1}x_{m+1}$$

$$\stackrel{\pi'}{\longmapsto} 0y_1\ldots y_nx_{m+1}x_{m+1}x_1x_2\ldots x_mx_{m+1},$$

where $\pi = \tau_{m+3,m+4}\ldots\tau_{1,2}\tau_{m+3,m+4}\ldots\tau_{3,4}$ shifts the two copies of x_{m+1} at the left end from positions 1 and 3 to positions $m+3$ and $m+4$, while shifting 0 to position 1 and shifting $x_1\ldots x_m$ two positions to the left; and $\pi' = \tau_{m+3,m+4}\ldots\tau_{m+n+2,m+n+3}\tau_{m+2,m+3}\ldots\tau_{m+n+1,m+n+2}$ shifts $x_{m+1}x_{m+1}$ from positions $m+n+2$ and $m+n+3$ to positions $m+2$ and $m+3$, while shifting $x_1\ldots x_m$ two positions to the right.

So, $W_C = \pi' W_K \pi\varphi_\vee\tau_{1,m+4}\varphi_\vee\tau_{3,m+4}\sigma^2$, hence $|W_C| = |W_K|+2m+n+10$, if all of $\{\tau_{i,j}\colon 1\leqslant i<j\}$ is used in the generating set; over $\{\tau_{i,i+1}\colon 1\leqslant i\}$, $\tau_{1,m+4}$ has length $\leqslant 2(m+3)-1$ and $\tau_{3,m+4}$ has length $\leqslant 2m-1$. Hence, $|W_C|\leqslant |W_K|+6m+n+14$. Moreover, $J_C = \max\{J_K, m+n+3\}$.

In all cases, $|W_C|\leqslant |W_K|+c\cdot(m+n+1)$ (for some constant $c>1$), and $J_C\leqslant \max\{J_K, n+m+3\}$. Thus, each slice c_ℓ, with input–output map $c_\ell(.) : y^{(\ell-1)} \longmapsto y^{(\ell)}$, is represented by a word W_{c_ℓ} with map $w_{c_\ell}(.) : 0y^{(\ell-1)} \longmapsto 0y^{(\ell)}y^{(\ell-1)}$, such that $|W_{c_\ell}|\leqslant c\cdot(n_{\ell-1}^2+n_\ell^2)$ (for some constant $c>1$), and $J_{c_\ell}\leqslant n_{\ell-1}+n_\ell+c$.

Regarding wire-crossings, we do not include them into other slices; we put the wire-crossings into pure wire-crossing slices. So we consider two kinds of slices: Slices entirely made of wire-crossings and identities, slices without any wire-crossings. Wire-crossings in circuits are identical to the group elements $\tau_{i,i+1}$.

We now construct the word W_f from the words W_{c_ℓ} ($\ell = 1,\ldots,L$). First observe that since the map $w_{c_\ell}(.)$ is a right-ideal isomorphism (being an element of $G_{2,1}$), we not only have

$$w_{c_\ell}(.) : 0y^{(\ell-1)} \longmapsto 0y^{(\ell)}y^{(\ell-1)}$$

but also

$$w_{c_\ell}(.) : 0y^{(\ell-1)}y^{(\ell-2)}\dots y^{(1)}y^{(0)} \longmapsto 0y^{(\ell)}y^{(\ell-1)}y^{(\ell-2)}\dots y^{(1)}y^{(0)}.$$

Then, by concatenating all W_{c_ℓ} (and by recalling that $y = y^{(L)}$ and $x = y^{(0)}$) we obtain

$$w_{c_L}w_{c_{L-1}}\dots w_{c_2}w_{c_1}(.) : 0x \longmapsto 0yy^{(L-1)}\dots y^{(2)}y^{(1)}x.$$

Let π_{C_f} be the position permutation that shifts y right to the positions just right of x:

$$\pi_{C_f} : 0yy^{(L-1)}\dots y^{(2)}y^{(1)}x \longmapsto 0y^{(L-1)}\dots y^{(2)}y^{(1)}xy.$$

Observe that for $(W_{c_{L-1}}\dots W_{c_2}W_{c_1})^{-1}$ we have

$$(w_{c_{L-1}}\dots w_{c_2}w_{c_1})^{-1}(.) : 0y^{(L-1)}\dots y^{(2)}y^{(1)}xy \longmapsto 0xy.$$

Then we have:

$$w_{c_L}w_{c_{L-1}}\dots w_{c_2}w_{c_1}\pi_{C_f}(w_{c_{L-1}}\dots w_{c_2}w_{c_1})^{-1}(.) : 0x \longmapsto 0xy.$$

By using the position permutation $\pi_{m,n} : 0xy \longmapsto 0yx$, we now see how to define W_f:

$$W_f = \pi_{m,n}W_{c_L}W_{c_{L-1}}\dots W_{c_2}W_{c_1}\pi_{C_f}(W_{c_{L-1}}\dots W_{c_2}W_{c_1})^{-1}.$$

Then we have:

$$w_f(.) : 0x \longmapsto 0yx,$$

where $y = f(x)$.

Finally, we need to examine the length of the word W_f in terms of the size of the circuit C_f that computes $f : \{0,1\}^m \to \{0,1\}^n$.

The position permutation $\pi_{m,n}$ shifts the $n = |y|$ letters of y to the left over the $m = |x|$ positions of x. So, $\pi_{m,n}$ can be written as the product of nm transpositions in $\{\tau_{i,i+1}\colon 1 \leqslant i\}$, with maximum subscript $J_{\pi_{m,n}} \leqslant m+n+1$.

The position permutation π_{C_f} shifts y to the right from positions in the interval $[2, n+1]$ within the string $0yy^{(L-1)}\dots y^{(2)}y^{(1)}x$ to positions in the interval $[2+\sum_{i=0}^{L-1} n_i, 2+\sum_{i=0}^{L} n_i]$ within the string $0y^{(L-1)}\dots y^{(2)}y^{(1)}xy$. Note that $\sum_{i=0}^{L} n_i = |C_f|$ (the size of the circuit C_f), and $n_L = |y| = n$, $n_0 = |x| = m$. We shift y starting with the right-most letters of y. This takes $n\sum_{i=0}^{L-1} n_i = n(|C_f| - n)$ transpositions in $\{\tau_{i,i+1}\colon 1 \leqslant i\}$, with maximum subscript $J_{\pi_{C_f}} = |C_f| + 2$.

We saw already that $|W_{c_\ell}| \leqslant c(n_{\ell-1}^2 + n_\ell^2)$, and $J_{c_\ell} \leqslant n_{\ell-1} + n_\ell + c$, for some constant $c > 1$. Note that $\sum_{i=0}^{L} n_i^2 \leqslant (\sum_{i=0}^{L} n_i)^2 = |C_f|^2$. Hence we have: $|W_f| \leqslant c_o|C_f|^2$, for some constant $c_o > 1$. Moreover, the largest subscript in any transposition occurring in W_f is $J_{W_f} \leqslant |C_f| + 2$.

Recall that we assumed that our circuit C_f was *strictly layered*, and that the circuit size has to be squared (at most) in order to make the circuit strictly layered. Thus, if C_f was originally not strictly layered, our bounds become $|W_f| \leqslant c_o|C_f|^4$, and $J_{W_f} \leqslant |C_f|^2 + 2$. □

The next theorem gives a representation of a boolean permutation by an element of the Thompson group $G_{2,1}$; the main point of the theorem is the polynomial bound on the *word-length* in terms of *circuit size*.

Theorem 4.2 *(Representation of permutations by the Thompson group). Let $g : \{0,1\}^m \to \{0,1\}^m$ be any permutation and let C_g and $C_{g^{-1}}$ be minimum-size circuits that compute g, respectively g^{-1}. Then there exists a word $W_{(g,g^{-1})}$ over the generating set $\Gamma_{G_{2,1}} \cup \{\tau_{i,i+1}\colon 1 \leqslant i\}$ of $G_{2,1}$, representing an element $w_{(g,g^{-1})} \in G_{2,1}$ such that:*

- *For all $x \in \mathsf{Dom}(g)$ and all $y \in \mathsf{Im}(g)$:*

$$w_{(g,g^{-1})}(0x) = 0g(x), \quad \text{and} \quad (w_{(g,g^{-1})})^{-1}(0y) = 0g^{-1}(y),$$

  where $(w_{(g,g^{-1})})^{-1} \in G_{2,1}$ is represented by the free-group inverse $(W_{(g,g^{-1})})^{-1}$ of the word $W_{(g,g^{-1})}$.
- *$w_{(g,g^{-1})}(.)$ and $(w_{(g,g^{-1})})^{-1}$ stabilize both $0\{0,1\}^*$ and $1\{0,1\}^*$.*
- *We have a length upper bound $|W_{(g,g^{-1})}| = |(W_{(g,g^{-1})})^{-1}| \leqslant O(|C_g|^4 + |C_{g^{-1}}|^4)$.*
- *The largest subscript of transpositions $\tau_{i,i+1}$ occurring in $W_{(g,g^{-1})}$ is $\leqslant \max\{|C_g|^2, |C_{g^{-1}}|^2\} + 2$.*

Note that we distinguish between the word $W_{(g,g^{-1})}$ (over a generating set of $G_{2,1}$) and the element $w_{(g,g^{-1})}$ of $G_{2,1}$ represented by $W_{(g,g^{-1})}$. Also, note that although g is length-preserving ($g \in lp\,G_{2,1}$), $w_{(g,g^{-1})} \in G_{2,1}$ is not length-preserving.

Proof. Consider the position permutation $\pi : 0yx \longmapsto 0xy$, for all $x, y \in \{0,1\}^m$; we express π as a composition of $\leqslant m^2$ position transpositions of the form $\tau_{i,i+1}$. Let W_g be the word constructed in Theorem 4.1 for g, and let $W_{g^{-1}}$ be the word constructed for g^{-1}. We define $W_{(g,g^{-1})}$ by

$$W_{(g,g^{-1})} = (W_{g^{-1}})^{-1} \pi\, W_g.$$

Then for all $x \in \mathsf{Dom}(g)$ we have: $w_{(g,g^{-1})} : 0x \longmapsto 0y$, where $y = g(x)$. More precisely, for all $x \in \mathsf{domC}(g)$,

$$0x \xrightarrow{w_g} 0g(x)x = 0yx \xrightarrow{\pi} 0xy = 0g^{-1}(y) \xrightarrow{(w_{g^{-1}})^{-1}} 0y = 0g(x).$$

Since $\mathsf{domC}(g)$ is a maximal prefix code, $w_{(g,g^{-1})}$ maps $0\{0,1\}^*$ into $0\{0,1\}^*$ (where defined).

Similarly, for all $y \in \mathsf{Im}(g) = \mathsf{Dom}(g^{-1})$ we have: $(w_{(g,g^{-1})})^{-1} : 0y \longmapsto 0x$, where $x = g^{-1}(y)$, $y = g(x)$. Since $\mathsf{domC}(g^{-1})$ is a maximal prefix code, $(w_{(g,g^{-1})})^{-1}$ maps $0\{0,1\}^*$ into $0\{0,1\}^*$ (where defined). Hence, elements of $0\{0,1\}^*$ are never images of $1\{0,1\}^*$. Thus, $1\{0,1\}^*$ is also stabilized by $w_{(g,g^{-1})}$ and by $(w_{(g,g^{-1})})^{-1}$.

The length of the word $W_{(g,g^{-1})}$ is bounded as follows: We have $|W_g| \leqslant c_o |C_g|^4$, and $|(W_{g^{-1}})^{-1}| = |W_{g^{-1}}| \leqslant c_o |C_{g^{-1}}|^4$, by Theorem 4.1. Moreover, π can be expressed as the composition of $\leqslant m^2$ $(< |C_g|^2)$ transpositions in $\{\tau_{i,i+1}\colon 1 \leqslant i\}$.

The bound on the subscripts also follows from Theorem 4.1. $\square$

5. Distortion vs. computational asymmetry

We show in this section that the computational asymmetry function $\alpha(.)$ is polynomially related to a certain distortion of the group $lp\,G_{2,1}$.

By Theorem 4.2, for every element $g \in lp\,G_{2,1}$ there is an element $w_{(g,g^{-1})} \in G_{2,1}$ which agrees with g on $0\{0,1\}^*$, and which stabilizes $0\{0,1\}^*$ and $1\{0,1\}^*$. The main property of $W_{(g,g^{-1})}$ is that its length is polynomially bounded by the circuit sizes of g and g^{-1}; that fact will be crucial later. First we want to study how $w_{(g,g^{-1})}$ is related to g. Recall that we distinguish between the word $W_{(g,g^{-1})}$ (over a generating set of $G_{2,1}$) and the element $w_{(g,g^{-1})}$ of $G_{2,1}$ represented by $W_{(g,g^{-1})}$.

Theorem 4.2 inspires the following concepts.

Definition 5.1. Let G be a subgroup of $G_{2,1}$. For any prefix codes $P_1, \ldots, P_k \subset \{0,1\}^*$, the *joint stabilizer* (in G) of the right ideals $P_1\{0,1\}^*, \ldots, P_k\{0,1\}^*$ is defined by

$$\mathsf{Stab}_G(P_1, \ldots, P_k) = \{g \in G\colon\ g(P_i\{0,1\}^*) \subseteq P_i\{0,1\}^* \text{ for every } i = 1, \ldots, k\}.$$

The *fixator* (in G) of $P_1\{0,1\}^*$ is defined by

$$\mathsf{Fix}_G(P_1) = \{g \in G\colon\ g(x) = x \text{ for all } x \in P_1\{0,1\}^*\}.$$

The fixator is also called "point-wise stabilizer."

The following is an easy consequence of the definition: $\mathsf{Fix}_G(P_i)$ is a subgroup of G ($\subseteq G_{2,1}$), for $i = 1, \ldots, k$. If the prefix codes $P_1, \ldots, P_k$ are such that the right ideals $P_1\{0,1\}^*, \ldots, P_k\{0,1\}^*$ are two-by-two disjoint, and such that $P_1 \cup \cdots \cup P_k$ is a maximal prefix code, then $\mathsf{Stab}_G(P_1, \ldots, P_k)$ is closed under inverse. Hence in this case $\mathsf{Stab}_G(P_1, \ldots, P_k)$ is a subgroup of G.

In particular, we will consider the following groups:

- The joint stabilizer of $0\{0,1\}^*$ and $1\{0,1\}^*$,

$$\mathsf{Stab}_G(0,1) = \{g \in G\colon\ g(0\{0,1\}^*) \subseteq 0\{0,1\}^* \text{ and } g(1\{0,1\}^*) \subseteq 1\{0,1\}^*\}.$$

- The fixator of $0\{0,1\}^*$,

$$\mathsf{Fix}_G(0) = \{g \in G\colon\ g(x) = x \text{ for all } x \in 0\{0,1\}^*\}.$$

- The fixator of $1\{0,1\}^*$,

$$\mathsf{Fix}_G(1) = \{g \in G\colon\ g(x) = x \text{ for all } x \in 1\{0,1\}^*\}.$$

Clearly, $\mathsf{Fix}_G(0)$ and $\mathsf{Fix}_G(1)$ are subgroups of $\mathsf{Stab}_G(0,1)$.

Lemma 5.2 *(Self-embeddings of $G_{2,1}$). Let G be a subgroup of $G_{2,1}$. Then G is isomorphic to* $\mathsf{Fix}_G(1)$ *and to* $\mathsf{Fix}_G(0)$ *by the following isomorphisms*:

$$\Lambda_0 : g \in G \longmapsto (g)_0 \in \mathsf{Fix}_G(1),$$
$$\Lambda_1 : g \in G \longmapsto (g)_1 \in \mathsf{Fix}_G(0),$$

where $(g)_0$ and $(g)_1$ defined as follows for any $g \in G_{2,1}$:

$$(g)_0\colon \begin{cases} 0x \in 0\{0,1\}^* \longmapsto 0g(x), \\ 1x \in 1\{0,1\}^* \longmapsto 1x, \end{cases} \qquad (g)_1\colon \begin{cases} 1x \in 1\{0,1\}^* \longmapsto 1g(x), \\ 0x \in 0\{0,1\}^* \longmapsto 0x. \end{cases}$$

Proof. It is straightforward to verify that Λ_0 and Λ_1 are injective homomorphisms. That Λ_0 is onto $\mathsf{Fix}_G(1)$ can be seen from the fact that every element of $\mathsf{Fix}_G(1)$ has a table of the form

$$\begin{bmatrix} 0x_1 & \cdots & 0x_n & 1 \\ 0y_1 & \cdots & 0y_n & 1 \end{bmatrix}$$

where $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_n\}$ are two maximal prefix codes, and $\left[\begin{smallmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{smallmatrix}\right]$ is an arbitrary element of G. $\Box$

Lemma 5.3. *Let G be a subgroup of $G_{2,1}$. Then the direct product $G \times G$ is isomorphic to* $\mathsf{Stab}_G(0,1)$ *by the isomorphism*

$$\Lambda : (f, g) \in G \times G \longmapsto \big(0x \mapsto 0f(x),\ 1x \mapsto 1g(x)\big) \in \mathsf{Stab}_G(0,1).$$

Proof. It is straightforward to verify that Λ is a homomorphism. That Λ is onto $\mathsf{Stab}_G(0,1)$ and injective follows from the fact that every element of $\mathsf{Stab}_G(0,1)$ has a table of the form

$$\begin{bmatrix} 0x_1 & \cdots & 0x_m & 1x'_1 & \cdots & 1x'_n \\ 0y_1 & \cdots & 0y_m & 1y'_1 & \cdots & 1y'_n \end{bmatrix}$$

where $\{x_1,\dots,x_m\}$, $\{y_1,\dots,y_m\}$, $\{x'_1,\dots,x'_n\}$, and $\{y'_1,\dots,y'_n\}$, are maximal prefix codes, and $\left[\begin{smallmatrix} x_1 & \cdots & x_m \\ y_1 & \cdots & y_m \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} x'_1 & \cdots & x'_n \\ y'_1 & \cdots & y'_n \end{smallmatrix}\right]$ are arbitrary elements of G ($\subseteq G_{2,1}$). □

Lemmas 5.2 and 5.3 reveal certain *self-similarity* properties of the Thompson group $G_{2,1}$. (Self-similarity of groups with total action on an infinite tree is an important subject, see [28]. However, the action of $G_{2,1}$ is partial, so much of the known theory does not apply directly.)

The stabilizer and the fixators above have some interesting properties.

Lemma 5.4.

(1) *For all $f, g \in G$: $(f)_0(g)_1 = (g)_1(f)_0$*
(*i.e., the commutator of* $\mathsf{Fix}_G(0)$ *and* $\mathsf{Fix}_G(1)$ *is the identity*).
(2) $\mathsf{Fix}_G(0) \cdot \mathsf{Fix}_G(1) = \mathsf{Stab}_G(0,1)$ *and* $\mathsf{Fix}_G(0) \cap \mathsf{Fix}_G(1) = \mathbf{1}$;
(3) $\mathsf{Stab}_G(0,1)$ *is the internal direct product of* $\mathsf{Fix}_G(0)$ *and* $\mathsf{Fix}_G(1)$.
(*This is equivalent to the combination of* (1) *and* (2).)
(4) *For all $f, g \in G$: $\Lambda(f,g) = \Lambda_0(f) \cdot \Lambda_1(g)$, $\Lambda_0(f) = \Lambda(f, \mathbf{1})$, and $\Lambda_1(g) = \Lambda(\mathbf{1}, g)$.*
Moreover, $\mathsf{Fix}_G(0) = \Lambda_1(G)$, $\mathsf{Fix}_G(1) = \Lambda_0(G)$, *and* $\mathsf{Stab}_G(0,1) = \Lambda(G \times G)$.

Proof. The proof is a straightforward verification. □

Lemma 5.5. *For every position transposition $\tau_{i,j}$, with $1 \leqslant i < j$, we have*

$$(\tau_{i,j})_0 = \tau_{2,i+1} \circ \tau_{3,j+1} \circ (\tau_{1,2})_0 \circ \tau_{3,j+1} \circ \tau_{2,i+1}.$$

Hence, assuming $(\tau_{1,2})_0 \in \Gamma_{G_{2,1}}$, and abbreviating $\{\tau_{i,j}\colon 0 < i < j\}$ by τ, we have:

$$\left|(\tau_{i,j})_0\right|_{\Gamma_{G_{2,1}} \cup \tau} \leqslant 5.$$

Proof. Recall that for $(\tau_{1,2})_0$ we have, by definition, $(\tau_{1,2})_0(1w) = 1w$, and $(\tau_{1,2})_0(0x_2x_3w) = 0x_3x_2w$, for all $w \in \{0,1\}^*$ and $x_2, x_3 \in \{0,1\}$. The proof of the lemma is a straightforward verification. □

Now we arrive at the relation between $w_{(g,g^{-1})}$ and g.

Lemma 5.6. *For all $g \in lp\,G_{2,1}$ the following relation holds between g and $w_{(g,g^{-1})}$*:

$$w_{(g,g^{-1})} \cdot (g)_0^{-1},\ (g)_0^{-1} \cdot w_{(g,g^{-1})} \in \mathsf{Fix}_{lp\,G_{2,1}}(0).$$

Equivalently,

$$(g)_0 \cdot \mathsf{Fix}_{lp\,G_{2,1}}(0) = w_{(g,g^{-1})} \cdot \mathsf{Fix}_{lp\,G_{2,1}}(0), \quad \textit{and}$$

$$\mathsf{Fix}_{lp\,G_{2,1}}(0) \cdot (g)_0 = \mathsf{Fix}_{lp\,G_{2,1}}(0) \cdot w_{(g,g^{-1})}.$$

Proof. By Theorem 4.2 we have $w_{(g,g^{-1})}(0x) = 0g(x)$ for all $x \in \mathsf{Dom}(g)$. So, $w_{(g,g^{-1})}$ and $(g)_0$ act in the same way on $0\{0,1\}^*$. Also, both $w_{(g,g^{-1})}$ and $(g)_0$ map $0\{0,1\}^*$ into $0\{0,1\}^*$, and both map $1\{0,1\}^*$ into $1\{0,1\}^*$. The lemma follows from this. $\square$

We abbreviate $\{\tau_{i,j}\colon\ 0 < i < j\}$ by τ. The element $w_{(g,g^{-1})}$ of $G_{2,1}$, represented by the word $W_{(g,g^{-1})}$, belongs to $\mathsf{Stab}_{lp\,G_{2,1}}(0,1)$ as we saw in Theorem 4.2. However, the word $W_{(g,g^{-1})}$ itself is a sequence over the generating set $\Gamma_{G_{2,1}} \cup \tau$ of $G_{2,1}$. Therefore, in order to follow the action of $W_{(g,g^{-1})}$ and of its prefixes we need to take $\mathsf{Fix}(0)$ as a subgroup of $G_{2,1}$. This leads us to the **Schreier left coset graph** of $\mathsf{Fix}_{G_{2,1}}(0)$ within $G_{2,1}$, over the generating set $\Gamma_{G_{2,1}} \cup \tau$. By definition this Schreier graph has vertex set $G_{2,1}/\mathsf{Fix}_{G_{2,1}}(0)$, i.e., the left cosets, of the form $g \cdot \mathsf{Fix}_{G_{2,1}}(0)$ with $g \in G_{2,1}$. And it has directed edges of the form $g \cdot \mathsf{Fix}_{G_{2,1}}(0) \xrightarrow{\gamma} \gamma g \cdot \mathsf{Fix}_{G_{2,1}}(0)$ for $g \in G_{2,1}$, $\gamma \in \Gamma_{G_{2,1}} \cup \tau$. Lemma 5.6 implies that for all $g \in lp\,G_{2,1}$,

$$(g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0) = w_{(g,g^{-1})} \cdot \mathsf{Fix}_{G_{2,1}}(0).$$

We assume that $\Gamma_{G_{2,1}} = \Gamma_{G_{2,1}}^{-1}$, so the Schreier graph is symmetric, and hence it has a distance function based on path length; we denote this distance by

$$d_{G/F}(.,.) : G_{2,1}/\mathsf{Fix}_{G_{2,1}}(0) \times G_{2,1}/\mathsf{Fix}_{G_{2,1}}(0) \longrightarrow \mathbb{N}.$$

Lemma 5.7. *There are injective morphisms*

$$g \in lp\,G_{2,1} \hookrightarrow g \in G_{2,1} \overset{\simeq}{\Rightarrow} (g)_0 \in \mathsf{Fix}_{G_{2,1}}(1) \overset{\simeq}{\Rightarrow} (g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0) \in \mathsf{Stab}_{G_{2,1}}(0,1)/\mathsf{Fix}_{G_{2,1}}(0),$$

and an inclusion map

$$(g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0) \in \mathsf{Stab}_{G_{2,1}}(0,1)/\mathsf{Fix}_{G_{2,1}}(0) \hookrightarrow (g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0) \in G_{2,1}/\mathsf{Fix}_{G_{2,1}}(0).$$

In particular,

$$g \in G_{2,1} \longmapsto (g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0) \in G_{2,1}/\mathsf{Fix}_{G_{2,1}}(0)$$

is an embedding of $G_{2,1}$, as a set, into the vertex set $G_{2,1}/\mathsf{Fix}_{G_{2,1}}(0)$ of the Schreier graph.

Proof. Recall that the map $\Lambda_0 : g \in G_{2,1} \longmapsto (g)_0 \in \mathsf{Fix}_{G_{2,1}}(1)$ is a bijective morphism (Lemma 5.2). Also, the map $u \in \mathsf{Fix}_{G_{2,1}}(1) \longmapsto u \cdot \mathsf{Fix}_{G_{2,1}}(0) \in G_{2,1}/\mathsf{Fix}_{G_{2,1}}(0)$ is injective; indeed, if $u \cdot \mathsf{Fix}_{G_{2,1}}(0) = v \cdot \mathsf{Fix}_{G_{2,1}}(0)$ with $u, v \in \mathsf{Fix}_{G_{2,1}}(1)$ then $v^{-1}u \in \mathsf{Fix}_{G_{2,1}}(0) \cap \mathsf{Fix}_{G_{2,1}}(1) = \{\mathbf{1}\}$.

The map $g \in G_{2,1} \longmapsto (g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0) \in \mathsf{Stab}_{G_{2,1}}(0,1)/\mathsf{Fix}_{G_{2,1}}(0)$ is a surjective group homomorphism since $\mathsf{Fix}_{G_{2,1}}(0)$ is a normal subgroup of $\mathsf{Stab}_{G_{2,1}}(0,1)$. Since $\mathsf{Fix}_{G_{2,1}}(0) \cap \mathsf{Fix}_G(1) = \{\mathbf{1}\}$, this homomorphism is injective from $\mathsf{Fix}_{G_{2,1}}(1)$ onto $\mathsf{Stab}_{G_{2,1}}(0,1)/\mathsf{Fix}_{G_{2,1}}(0)$.

The combination of these maps provides an isomorphism from $G_{2,1}$ onto $\mathsf{Stab}_{G_{2,1}}(0,1)/\mathsf{Fix}_{G_{2,1}}(0)$. Hence we also have an embedding of $G_{2,1}$, as a set, into the vertex set $G_{2,1}/\mathsf{Fix}_{G_{2,1}}(0)$ of the Schreier graph. $\square$

Since by Lemma 5.7 we can consider $G_{2,1}$ as a subset of the vertex set $G_{2,1}/\mathsf{Fix}_{G_{2,1}}(0)$ of the Schreier graph, the path-distance $d_{G/F}(.,.)$ on $G_{2,1}/\mathsf{Fix}_{G_{2,1}}(0)$ leads to a distance on $G_{2,1}$, inherited from $d_{G/F}(.,.)$:

Definition 5.8. For all $g, g' \in G_{2,1}$ the *Schreier graph distance inherited by* $G_{2,1}$ is

$$D(g, g') = d_{G/F}\big((g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0), (g')_0 \cdot \mathsf{Fix}_{G_{2,1}}(0)\big).$$

The comparison of the Schreier graph distance $D(.,.)$ on $lp\,G_{2,1}$ with the word-length that $lp\,G_{2,1}$ inherits from its embedding into $lep\,M_{2,1}$ leads to the following *distortion of* $lp\,G_{2,1}$:

Definition 5.9. In $lp\,G_{2,1}$ we consider the *distortion*

$$\Delta(n) = \max\big\{D(\mathbf{1}, g)\colon\ |g|_{lep\,M_{2,1}} \leqslant n,\ \ g \in lp\,G_{2,1}\big\}.$$

We now state and prove the main theorem relating $\Delta(.)$ and α. Recall that $\alpha(.)$ is the computational asymmetry function of boolean permutations, defined in terms of circuit size.

Theorem 5.10 *(Computational asymmetry vs. distortion). The computational asymmetry function $\alpha(.)$ and the distortion $\Delta(.)$ of $lp\,G_{2,1}$ are polynomially related. More precisely, for all $n \in \mathbb{N}$:*

$$\big(\alpha(n)\big)^{1/2} \leqslant c' \cdot \Delta(n) \leqslant cn^4 + c \cdot \big(\alpha(cn)\big)^4$$

where $c \geqslant c' \geqslant 1$ are constants.

Proof. The theorem follows immediately from Lemmas 5.11 and 5.12. $\square$

Lemma 5.11. *There is a constant $c \geqslant 1$ such that for all $n \in \mathbb{N}$: $\Delta(n) \leqslant cn^4 + c \cdot (\alpha(cn))^4$.*

Proof. By Lemma 5.6, $(g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0) = w_{(g,g^{-1})} \cdot \mathsf{Fix}_{G_{2,1}}(0)$, hence

$$d\big(\mathsf{Fix}_{G_{2,1}}(0), (g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0)\big) = d\big(\mathsf{Fix}_{G_{2,1}}(0), w_{(g,g^{-1})} \cdot \mathsf{Fix}_{G_{2,1}}(0)\big).$$

Since the word $W_{(g,g^{-1})}$ and the Schreier graph use the same generating set, namely $\Gamma_{G_{2,1}} \cup \tau$, we have

$$d\big(\mathsf{Fix}_{G_{2,1}}(0), w_{(g,g^{-1})} \cdot \mathsf{Fix}_{G_{2,1}}(0)\big) \leqslant |W_{(g,g^{-1})}|.$$

By Theorem 4.2, $|W_{(g,g^{-1})}| \leqslant O(|C_g|^4 + |C_{g^{-1}}|^4)$. And by the definition of the computational asymmetry function, $|C_{g^{-1}}| \leqslant \alpha(|C_g|)$. Hence

$$d\big(\mathsf{Fix}_{G_{2,1}}(0), (g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0)\big) \leqslant O\big(|C_g|^4 + |C_{g^{-1}}|^4\big) \leqslant O\big(|C_g|^4 + \alpha\big(|C_g|\big)^4\big).$$

By Proposition 2.4, $|C_g| = O(|g|_{lep\,M_{2,1}})$. Hence, for some constants $c,'' c' \geqslant 1$,

$$d\big(\mathsf{Fix}_{G_{2,1}}(0), (g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0)\big) \leqslant c' \cdot |g|^4_{lep\,M_{2,1}} + c' \cdot \alpha\big(c'' \cdot |g|_{lep\,M_{2,1}}\big)^4.$$

Thus,

$$\max\big\{d\big(\mathsf{Fix}_{G_{2,1}}(0), (g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0)\big)\colon\ |g|_{lep\,M_{2,1}} \leqslant n,\ \ g \in lp\,G_{2,1}\big\} \leqslant c'n^4 + c'\alpha(c''n)^4.$$

By Definition 5.9 of the distortion function Δ we have therefore

$$\Delta(n) \leqslant c'n^4 + c'\alpha(c''n)^4.$$

This proves the lemma. $\square$

Lemma 5.12. *There is a constant $c \geqslant 1$ such that for all $n \in \mathbb{N}$: $\alpha(n) \leqslant c \cdot \Delta(cn)^2$.*

Proof. We first prove the following.

Claim. *For every $g \in lp\, G_{2,1}$, the inverse permutation g^{-1} can be computed by a circuit $C_{g^{-1}}$ of size $|C_{g^{-1}}| \leqslant c \cdot d(\mathsf{Fix}_{G_{2,1}}(0), (g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0))^2$, for some constant $c \geqslant 1$.*

Proof. There is a word W' of length $|W'| = d(\mathsf{Fix}_{G_{2,1}}(0), (g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0))$ over $\Gamma_{G_{2,1}} \cup \tau$ that labels a shortest path from $\mathsf{Fix}_{G_{2,1}}(0)$ to $(g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0)$ in the Schreier graph of $G_{2,1}/\mathsf{Fix}_{G_{2,1}}(0)$. Let $W = (W')^{-1}$ (the free-group inverse of W'), so $|W| = |W'|$. Let w be the element of $G_{2,1}$ represented by W. Then W labels a shortest path from $\mathsf{Fix}_{G_{2,1}}(0)$ to $(g^{-1})_0 \cdot \mathsf{Fix}_{G_{2,1}}(0)$ in the Schreier graph of $G_{2,1}/\mathsf{Fix}_{G_{2,1}}(0)$; this path has length $|W| = |W'| = d(\mathsf{Fix}_{G_{2,1}}(0), (g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0)) = d(\mathsf{Fix}_{G_{2,1}}(0), (g^{-1})_0 \cdot \mathsf{Fix}_{G_{2,1}}(0))$.

We have $w \cdot \mathsf{Fix}_{G_{2,1}}(0) = (g^{-1})_0 \cdot \mathsf{Fix}_{G_{2,1}}(0)$, thus for all $x \in \{0,1\}^*$: $w(0x) = 0g^{-1}(x)$. We now take the word VWU over the generating set $\Gamma_{M_{2,1}} \cup \tau$ of the monoid $M_{2,1}$, where we choose the words U and V to be $U = (\mathsf{and}, \mathsf{not}, \mathsf{fork}, \mathsf{fork})$, and $V = (\mathsf{or})$. The functions and, not, fork, or were defined in Subsection 1.1. Then for all $x = x_1 \ldots x_n \in \{0,1\}^*$, with $x_1, \ldots, x_n \in \{0,1\}$, we have

$$\begin{aligned} x_1 \ldots x_n &\xrightarrow{\mathsf{fork}} x_1 x_1 \ldots x_n \xrightarrow{\mathsf{fork}} \xrightarrow{\mathsf{not}} \overline{x_1} x_1 x_1 \ldots x_n \xrightarrow{\mathsf{and}} 0 x_1 \ldots x_n \\ &= 0x \xrightarrow{W} 0g^{-1}(x) \xrightarrow{\mathsf{or}} g^{-1}(x). \end{aligned}$$

The last or combines 0 and the first bit of $g^{-1}(x)$, and this makes 0 disappear. Thus overall, $VWU(x) = g^{-1}(x)$. The length is $|VWU| = |W| + 5$.

Since $g^{-1} \in lp\, G_{2,1} \subset lep\, M_{2,1}$, Theorem 2.9 implies that there exists a word Z over the generators $\Gamma_{lep\, M_{2,1}} \cup \tau$ of $lep\, M_{2,1}$ such that

(1) $|Z| \leqslant c_1 \cdot |VWU|^2$, for some constant $c_1 \geqslant 1$, and
(2) Z represents the same element of $lep\, M_{2,1}$ as VWU, namely g^{-1}.

Moreover, by Proposition 2.4, the word Z can be transformed into a circuit of size $\leqslant c_2 \cdot |Z|$ (for some constant $c_2 \geqslant 1$). This proves that there is a circuit $C_{g^{-1}}$ for g^{-1} of size $|C_{g^{-1}}| \leqslant c \cdot |W|^2$ (for some constant $c \geqslant 1$). Since we saw that $|W| = d_{G/F}(\mathsf{Fix}_{G_{2,1}}(0), (g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0))$, the claim follows. □

By definition, $D(\mathbf{1}, g) = d_{G/F}(\mathsf{Fix}_{G_{2,1}}(0), (g)_0 \cdot \mathsf{Fix}_{G_{2,1}}(0))$. Hence, by the claim above:

$$|C_{g^{-1}}| \leqslant c \cdot \big(D(\mathbf{1}, g)\big)^2.$$

By Proposition 2.4 the word-length in $lep\, M_{2,1}$ and the circuit size are linearly related; hence $|g|_{lep\, M_{2,1}} \leqslant c_0 |C_g|$, for some constant $c_0 \geqslant 1$. Therefore,

$$\begin{aligned} \alpha(n) &= \max\big\{|C_{g^{-1}}|:\ |C_g| \leqslant n,\ g \in lp\, G_{2,1}\big\} \\ &\leqslant \max\big\{|C_{g^{-1}}|:\ |g|_{lep\, M_{2,1}} \leqslant c_0 n,\ g \in lp\, G_{2,1}\big\} \\ &\leqslant \max\big\{c \cdot \big(D(\mathbf{1}, g)\big)^2:\ |g|_{lep\, M_{2,1}} \leqslant c_0 n,\ g \in lp\, G_{2,1}\big\} \\ &\leqslant c \cdot \big(\Delta(c_0 n)\big)^2. \end{aligned}$$

This proves the lemma. □

6. Other distortion bounds

6.1. Other distortions in the Thompson groups and monoids

The next proposition gives more upper bounds on the computational asymmetry function α.

Proposition 6.1. *Assume $\Gamma_{lep\,G_{2,1}} \subset \Gamma_{lep\,M_{2,1}} \subset \Gamma_{M_{2,1}}$. Let $\delta_{lp\,G,lep\,M} = \delta[|.|_{\Gamma_{lp\,G_{2,1}} \cup \tau}, |.|_{\Gamma_{lep\,M_{2,1}} \cup \tau}]$ denote the distortion function of $lp\,G_{2,1}$ in the Thompson monoid $lep\,M_{2,1}$, based on word-length. Similarly, let $\delta_{lp\,G,M} = \delta[|.|_{\Gamma_{lp\,G_{2,1}} \cup \tau}, |.|_{\Gamma_{M_{2,1}} \cup \tau}]$ denote the distortion function of $lp\,G_{2,1}$ in the Thompson monoid $M_{2,1}$. Then for some constant $c \geqslant 1$ and for all $n \in \mathbb{N}$,*

$$\alpha(n) \leqslant c \cdot \delta_{lp\,G,lep\,M}(cn) \leqslant c \cdot \delta_{lp\,G,M}(cn).$$

Proof. We first prove that $\delta_{lp\,G,lep\,M}(n) \leqslant \delta_{lp\,G,M}(n)$. Recall that by definition, $\delta_{lp\,G,lep\,M}(n) = \max\{|g|_{lp\,G_{2,1}}\colon\ g \in lp\,G_{2,1},\ |g|_{lep\,M_{2,1}} \leqslant n\}$, and similarly for $\delta_{lp\,G,M}(n)$. Since $\Gamma_{lep\,M_{2,1}} \subset \Gamma_{M_{2,1}}$ we have $|x|_{lep\,M_{2,1}} \leqslant |x|_{M_{2,1}}$. Hence, $\{|g|_{lp\,G_{2,1}}\colon\ g \in lp\,G_{2,1},\ |g|_{lep\,M_{2,1}} \leqslant n\} \subseteq \{|g|_{lp\,G_{2,1}}\colon\ g \in lp\,G_{2,1},\ |g|_{M_{2,1}} \leqslant n\}$. By taking max over each of these two sets it follows that $\delta_{lp\,G,lep\,M}(n) \leqslant \delta_{lp\,G,M}(n)$.

Next we prove that $\alpha(n) \leqslant c \cdot \delta_{lp\,G,lep\,M}(cn)$. For any $g \in lp\,G_{2,1}$ we have $C(g^{-1}) \leqslant O(|g^{-1}|_{lep\,M_{2,1}})$, by Proposition 3.2. Moreover, $|g^{-1}|_{lep\,M_{2,1}} \leqslant |g^{-1}|_{lp\,G_{2,1}}$ since $lp\,G_{2,1}$ is a subgroup of $lep\,M_{2,1}$, and since the generating set used for $lp\,G_{2,1}$ (including all $\tau_{i,j}$) is a subset of the generating set used for $lep\,M_{2,1}$. For any group with generating set closed under inverse we have $|g^{-1}|_G = |g|_G$. And by the definition of the distortion $\delta_{lp\,G,lep\,M}$ we have $|g|_{lp\,G_{2,1}} \leqslant \delta_{lp\,G,lep\,M}(|g|_{lep\,M_{2,1}})$. And again, by Proposition 3.2, $|g|_{lep\,M_{2,1}} \leqslant O(C(g))$. Putting all this together we have

$$\begin{aligned} C\big(g^{-1}\big) &\leqslant c_1 \cdot \big|g^{-1}\big|_{lep\,M_{2,1}} \leqslant c_1 \cdot \big|g^{-1}\big|_{lp\,G_{2,1}} = c_1 \cdot |g|_{lp\,G_{2,1}} \\ &\leqslant c_1 \cdot \delta_{lp\,G,lep\,M}\big(|g|_{lep\,M_{2,1}}\big) \leqslant c_1 \cdot \delta_{lp\,G,lep\,M}\big(c_2 C(g)\big). \end{aligned}$$

Thus, $c_1 \cdot \delta_{lp\,G,lep\,M}(c_2 C(g))$ is an upper bound on $C(g^{-1})$. Since, by definition, $\alpha(C(g))$ is the smallest upper bound on $C(g^{-1})$, it follows that $\alpha(C(g)) \leqslant c_1 \cdot \delta_{lp\,G,lep\,M}(c_2 C(g))$. $\square$

Recall that in Definition 5.9 of the distortion Δ we compared $D(.,.)$ with the word-length in $lep\,M_{2,1}$. If, instead, we compare $D(.,.)$ with the word-length in $M_{2,1}$ we obtain the following distortion of $lp\,G_{2,1}$:

$$\delta(n) = \max\big\{D(\mathbf{1}, g)\colon\ |g|_{M_{2,1}} \leqslant n,\ g \in lp\,G_{2,1}\big\}.$$

Proposition 6.2. *The distortion functions $\Delta(.)$ and $\delta(.)$ are polynomially related. More precisely, there are constants $c', c_1, c_2 \geqslant 1$ such that for all $n \in \mathbb{N}$: $\Delta(n) \leqslant c_1\delta(n) \leqslant c_2\Delta(c'n^2)$.*

Proof. Let us assume first that $\Gamma_{lep\,M_{2,1}} \subseteq \Gamma_{M_{2,1}}$, from which it follows that $|g|_{M_{2,1}} \leqslant |g|_{M_{2,1}}$. Therefore, $\{D(\mathbf{1}, g)\colon\ |g|_{lep\,M_{2,1}} \leqslant n\} \subseteq \{D(\mathbf{1}, g)\colon\ |g|_{M_{2,1}} \leqslant n\}$. Hence, $\Delta(n) \leqslant \delta(n)$.

By Theorem 2.9, $|g|_{lep\,M_{2,1}} \leqslant c \cdot |g|^2_{M_{2,1}}$. So, $\{D(\mathbf{1}, g)\colon\ |g|_{M_{2,1}} \leqslant n\} \subseteq \{D(\mathbf{1}, g)\colon\ |g|_{lep\,M_{2,1}} \leqslant cn^2\}$. Hence, $\delta(n) \leqslant \Delta(cn^2)$.

When we do not have $\Gamma_{lep\,M_{2,1}} \subseteq \Gamma_{M_{2,1}}$, the constants in the theorem change, but the statement remains the same. $\square$

6.2. Monotone boolean functions and distortion

On $\{0,1\}^*$ we can define the *product order*, also called "bit-wise order." It is a partial order (and in fact, a lattice order), denoted by "$\preccurlyeq$," and defined as follows. First, $0 \prec 1$; next, for any $u, v \in \{0,1\}^*$

we have $u \preccurlyeq v$ iff $|u| = |v|$ and $u_i \preccurlyeq v_i$ for all $i = 1, \ldots, |u|$, where u_i (or v_i) denotes the ith bit of u (respectively v).

By definition, a partial function $f : \{0,1\}^* \to \{0,1\}^*$ is *monotone* (also called "product-order preserving") iff for all $u, v \in \mathrm{Dom}(f)$: $u \preccurlyeq v$ implies $f(u) \preccurlyeq f(v)$.

The following fact is well known (see e.g., [45, Section 4.5]): A function $f : \{0,1\}^m \to \{0,1\}^n$ is monotone iff f can be computed by a combinational circuit that only uses gates of type and, or, fork, and wire-swappings; i.e., not is absent. A circuit of this restricted type is called a *monotone circuit*.

Razborov [32] proved super-polynomial lower bounds for the size of monotone circuits that solve the clique problem, and in [33] he proved super-polynomial lower bounds for the size of monotone circuits that solve the perfect matching problem for bipartite graphs; the latter problem is in P. Tardos [39], based on work by Alon and Boppana [1], gave an exponential lower bound for the size of monotone circuits that solve a problem in P; see also [44] (Chapter 14 by Boppana and Sipser). Thus, there exist problems that can be solved by polynomial-size circuits but for which monotone circuits must have exponential size. In particular (for some constants $b > 1, c > 0$), there are infinitely many monotone functions $f_n : \{0,1\}^n \to \{0,1\}^n$ such that f_n has a combinational circuit of size $\leqslant n^c$, but f_n has no monotone circuit of size $\leqslant b^n$.

Based on an alphabet $A = \{a_1, \ldots, a_k\}$ with $a_1 \prec a_2 \prec \cdots \prec a_k$ we define a partial function $f : A^* \to A^*$ to be *monotone* iff f preserves the product order of A^*. The monotone functions enable us to define the following submonoid of the Thompson–Higman monoid $lep\, M_{k,1}$:

$$mon\, M_{k,1} = \{\varphi \in lep\, M_{k,1}\colon\ \varphi \text{ can be represented by a monotone function } P \to Q,$$
$$\text{where } P \text{ and } Q \text{ are prefix codes, with } P \text{ maximal}\}.$$

An essential extension or restriction of an element of $mon\, M_{k,1}$ is again in $mon\, M_{k,1}$, so this set is well defined as a subset of $lep\, M_{k,1}$. It is easily seen to be closed under composition, so $mon\, M_{k,1}$ is a submonoid of $lep\, M_{k,1}$.

We saw that all monotone finite functions have circuits made from gates of type and, or, fork. Hence $mon\, M_{2,1}$ has the following generating set:

$$\{\mathsf{and}, \mathsf{or}, \mathsf{fork}\} \cup \{\tau_{i,j}\colon\ j > i \geqslant 1\}.$$

The results about monotone circuit size imply the following distortion result. Here, "exponential" refers to a function with a lower bound of the form $n \in \mathbb{N} \longmapsto \exp(\sqrt[c]{c'n})$, for some constants $c' > 0$ and $c \geqslant 1$.

Proposition 6.3. *Consider the monoid $mon\, M_{2,1}$ over the generating set $\{\mathsf{and}, \mathsf{or}, \mathsf{fork}\} \cup \{\tau_{i,j}\colon\ j > i \geqslant 1\}$, and the monoid $lep\, M_{2,1}$ over the generating set $\Gamma_{lep\, M_{2,1}} \cup \{\tau_{i,j}\colon\ j > i \geqslant 1\}$, where $\Gamma_{lep\, M_{2,1}}$ is finite. Then $mon\, M_{2,1}$ has exponential word-length distortion in $lep\, M_{2,1}$.*

Proof. Let $\Gamma_{mon} = \{\mathsf{and}, \mathsf{or}, \mathsf{fork}\}$. By Proposition 2.4 we have $|f|_{\Gamma_{lep\, M_{2,1}} \cup \tau} = |C_f|$, where $|C_f|$ denotes the ordinary circuit size of f. By a similar argument we obtain: $|f|_{\Gamma_{mon} \cup \tau} = |mon\, C_f|$, where $|mon\, C_f|$ denotes the monotone circuit size of f. We saw that as a consequence of the work of Razborov, Alon, Boppana, and Tardos, there exists an infinite set of monotone functions that have polynomial-size circuits but whose monotone circuit size is exponential. The exponential distortion follows. □

Since $lep\, M_{2,1}$ has quadratic distortion in $M_{2,1}$, $mon\, M_{2,1}$ also has exponential word-length distortion in $M_{2,1}$.

6.3. Conclusion and open problems

The main theme of this paper is the distortion of groups within monoids, and the relation between this distortion and one-way permutations. The Thompson–Higman groups and monoids, as well as some of their subgroups and submonoids, can serve as models of computation. Moreover, we can define functions on those algebraic objects that, in addition to their own interest, are polynomially related to computational complexity functions.

As usual, the most interesting questions remain open:

1. What is the distortion of the Thompson group $G_{2,1}$ within the monoid $M_{2,1}$? This is especially interesting when we use finite generating sets $\Gamma_{G_{2,1}}$ and $\Gamma_{M_{2,1}}$, or when we use generating sets of the form $\Gamma_{G_{2,1}} \cup \tau$ and $\Gamma_{M_{2,1}} \cup \tau$, where τ is the set of transpositions of letter positions.
   The same question arises for the distortion of $lp\,G_{2,1}$ within $lep\,M_{2,1}$.
2. Let $\delta_{lp\,G,lep\,M}$ be the distortion of $lp\,G_{2,1}$ within $lep\,M_{2,1}$ over the generating sets $\Gamma_{lp\,G_{2,1}} \cup \tau$, respectively $\Gamma_{lep\,M_{2,1}} \cup \tau$. We saw in Proposition 6.1 that $\delta_{lp\,G,lep\,M}$ is an upper bound on the computational asymmetry function $\alpha(.)$. Does $\alpha(.)$ also have a *lower bound* of the form $(\delta_{lp\,G,lep\,M}(.))^c$ for some constant $c > 0$?
   Similarly, let $\delta_{G,M}(.)$ be the distortion of $G_{2,1}$ within $M_{2,1}$ over generating sets $\Gamma_{G_{2,1}} \cup \tau$, respectively $\Gamma_{M_{2,1}} \cup \tau$. Does $\alpha(.)$ also have a lower bound (and an upper bound) of the form $(\delta_{G,M}(.))^c$ for a constant $c > 0$?

These questions (in combination) can be expected to be very difficult (especially with the generating set $\Gamma_{G_{2,1}} \cup \tau$), as they are closely related to the major open question whether one-way permutations or one-way functions exist.

References

[1] N. Alon, R. Boppana, The monotone circuit complexity of boolean functions, Combinatorica 7 (1987) 1–23.

[2] G. Arzhantseva, C. Druţu, M. Sapir, Compression functions of uniform embeddings of groups into Hilbert and Banach spaces, J. Reine Angew. Math. (Crelle's Journal) (2008), in press, preprint: Mathematics arXiv: http://arxiv.org/abs/math.GR/0612378, 2006.

[3] G. Arzhantseva, V. Guba, M. Sapir, Metrics on diagram groups and uniform embeddings in a Hilbert space, Comment. Math. Helv. 81 (4) (2006) 911–929.

[4] C.H. Bennett, Logical reversibility of computation, IBM J. Res. Dev. 17 (November 1973) 525–532.

[5] C.H. Bennett, Time/space tradeoffs for reversible computation, SIAM J. Comput. 18 (1989) 766–776.

[6] J.C. Birget, The groups of Richard Thompson and complexity, Internat. J. Algebra Comput. 14 (5, 6) (December 2004) 569–626, preprint: Mathematics arXiv: math.GR/0204292, April 2002.

[7] J.C. Birget, Circuits, coNP-completeness, and the groups of Richard Thompson, Internat. J. Algebra Comput. 16 (1) (February 2006) 35–90, preprint: Mathematics arXiv: http://arXiv.org/abs/math.GR/0310335, October 2003.

[8] J.C. Birget, Factorizations of the Thompson–Higman groups, and circuit complexity, Internat. J. Algebra Comput. 18.2 (March 2008) 285–320, preprint: Mathematics arXiv: math.GR/0607349, July 2006.

[9] J.C. Birget, Monoid generalizations of the Richard Thompson groups, J. Pure Appl. Algebra, in press, preprint: Mathematics arXiv: math.GR/0704.0189, April 2007.

[10] J.C. Birget, One-way permutations, computational asymmetry and distortion, Mathematics arXiv: http://arxiv.org/abs/0704.1569, April 2007, preprint of the present paper, with a few more proof details.

[11] J. Bourgain, On Lipschitz embedding of finite metric spaces in Hilbert space, Israel J. Math. 52 (1985) 46–52.

[12] R.B. Boppana, J.C. Lagarias, One-way functions and circuit complexity, Inform. and Comput. 74 (1987) 226–240.

[13] J.W. Cannon, W.J. Floyd, W.R. Parry, Introductory notes on Richard Thompson's groups, Enseign. Math. 42 (1996) 215–256.

[14] W. Diffie, M.E. Hellman, New directions in cryptography, IEEE Trans. Inform. Theory 22 (1976) 644–655.

[15] B. Farb, The extrinsic geometry of subgroups and the generalized word problem, Proc. London Math. Soc. (3) 68 (1994) 577–593.

[16] E. Fredkin, T. Toffoli, Conservative logic, Internat. J. Theoret. Phys. 21 (1982) 219–253.

[17] J. Grollman, A. Selman, Complexity measure for public-key cryptosystems, SIAM J. Comput. 17 (1988) 309–335.

[18] M. Gromov, Asymptotic invariants of infinite groups, in: G. Niblo, M. Roller (Eds.), Geometric Group Theory, in: London Math. Soc. Lecture Note Ser., vol. 182, Cambridge Univ. Press, 1993.

[19] G. Higman, Finitely Presented Infinite Simple Groups, Notes Pure Math., vol. 8, The Australian National University, Canberra, 1974.

[20] A.P.L. Hiltgen, Cryptographically relevant contributions to combinatorial complexity, Dissertation, ETH-Zürich, Hartung-Gorre Verlag, Konstanz, 1994.

[21] A.P.L. Hiltgen, Construction of feebly-one-way families of permutations, in: Advances in Cryptology—AUSCRYPT'92, in: Lecture Notes in Comput. Sci., vol. 718, 1993, pp. 422–434.

[22] P. Indyk, J. Matousek, Low-distortion embeddings of finite metric spaces, Chapter 8 in: J.E. Goodman, J. O'Rourke (Eds.), Handbook of Discrete and Computational Geometry, second ed., CRC Press, LLC, Boca Raton, FL, 2004.

[23] Y. Lecerf, Machines de Turing réversibles…, C. R. Acad. Sci. Paris 257 (18) (October 1963) 2597–2600.

[24] O.B. Lupanov, A method of circuit synthesis, Izv. VUZ Radiofizika 1 (1958) 120–140.

[25] S. Margolis, J. Meakin, Z. Šunić, Distortion functions and the membership problem for submonoids of groups and monoids, in: Contemp. Math., vol. 372, Amer. Math. Soc., 2005, pp. 109–129.
[26] J.L. Massey, The difficulty with difficulty, in: IACR Distinguished Lecture delivered at EUROCRYPT '96, July 17, 1996, Saragossa, Spain, http://www.iacr.org/publications/dl/.
[27] R. McKenzie, R.J. Thompson, An elementary construction of unsolvable word problems in group theory, in: W.W. Boone, F.B. Cannonito, R.C. Lyndon (Eds.), Word Problems, North-Holland, 1973, pp. 457–478.
[28] V. Nekrashevych, Self-Similar Groups, Math. Surveys Monogr., vol. 117, Amer. Math. Soc., 2005.
[29] A.Y. Ol'shanskii, On subgroup distortion in finitely presented groups, Mat. Sb. 188 (1997) 51–98.
[30] A.Y. Ol'shanskii, M.V. Sapir, Length and area functions on groups and quasi-metric Higman embedding, Internat. J. Algebra Comput. 11 (2001) 137–170.
[31] G.B. Purdy, A high security log-in procedure, Comm. ACM 17 (1974) 442–445.
[32] A.A. Razborov, Lower bounds for the monotone complexity of some boolean functions, Dokl. Akad. Nauk SSSR 281 (4) (1985) 798–801, English transl.: Soviet Math. Dokl. 31 (1985) 354–357.
[33] A.A. Razborov, Lower bounds of monotone complexity of the logical permanent function, Mat. Zametki 37 (6) (1985) 887–900, English transl.: Math. Notes Acad. Sci. USSR 37 (1985) 485–493.
[34] J.E. Savage, Models of Computation, Addison–Wesley, 1998.
[35] E.A. Scott, A construction which can be used to produce finitely presented infinite simple groups, J. Algebra 90 (1984) 294–322.
[36] A. Selman, A survey of one-way functions in complexity theory, Math. Systems Theory 25 (1992) 203–221.
[37] C.E. Shannon, The synthesis of two-terminal switching circuits, Bell System Technical J. 28 (1949) 59–98.
[38] V. Shende, A. Prasad, I. Markov, J. Hayes, Synthesis of reversible logic circuits, IEEE Trans. Computer-Aided Design Integrated Circuits Systems 22 (6) (2003) 710–722.
[39] E. Tardos, The gap between monotone and non-monotone circuit complexity is exponential, Combinatorica 7 (4) (1987) 141–142.
[40] Richard J. Thompson, Manuscript, 1960s.
[41] R.J. Thompson, Embeddings into finitely generated simple groups which preserve the word problem, in: S. Adian, W. Boone, G. Higman (Eds.), Word Problems II, North-Holland, 1980, pp. 401–441.
[42] T. Toffoli, Reversible computing, MIT Laboratory for Computer Science, Technical Memo MIT/LCS/TM-151, 1980.
[43] T. Toffoli, Reversible computing, in: Automata, Languages and Programming (7th Colloquium), in: Lecture Notes in Comput. Sci., vol. 85, July 1980, pp. 623–644, abridged version of [42].
[44] J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, vol. A, MIT Press, Elsevier, 1990.
[45] I. Wegener, The Complexity of Boolean Functions, Wiley/Teubner, 1987.