View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Loughborough University Institutional Repository

Green's Relations in Deterministic Finite Automata*

Lukas Fleischer

FMI, University of Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany
fleischer@fmi.uni-stuttgart.de

Manfred Kufleitner

FMI, University of Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany
and
Department of Computer Science, Loughborough University
Loughborough LE11 3TU, United Kingdom
m.kufleitner@lboro.ac.uk

Abstract. Green's relations are a fundamental tool in the structure theory of semigroups. They can be defined by reachability in the (right/left/two-sided) Cayley graph. The equivalence classes of Green's relations then correspond to the strongly connected components. We study the complexity of Green's relations in semigroups generated by transformations on a finite set. We show that, in the worst case, the number of equivalence classes is in the same order of magnitude as the number of elements. Another important parameter is the maximal length of a chain of strongly connected components. Our main contribution is an exponential lower bound for this parameter. There is a simple construction for an arbitrary set of generators. However, the proof for a constant size alphabet is rather involved. We also investigate the special cases of unary and binary alphabets. All these results are extended to deterministic finite automata and their syntactic semigroups.

*This work was supported by the DFG grant DI 435/5-2.

1 Introduction

Extremal semigroup theory considers parameters of semigroups and investigates the maxima and minima of them. This line of research is driven by the question *"What is the worst that could happen?"* Typical parameters include the size of a finite semigroup, the minimal number of generators, or the number of states in transformation semigroups. The relation between these three is well understood; see below for further details. Another set of parameters is given by Green's relations. These relations are probably the most prominent tool for studying the structure of semigroups. Green's relations are naturally induced by pre-orders. This leads to two types of parameters: the number of equivalence classes and the maximal length of a chain. For both types we are interested in upper and lower bounds depending on the number of states. Before going into more detail, we introduce some basic notation.

Let Q be a finite set with n elements. There are n^n mappings from Q to Q. Such mappings are called *transformations* and the elements of Q are called *states*. The composition of transformations defines an associative operation. If Σ is some arbitrary subset of transformations, we can consider the *transformation semigroup* S generated by Σ; this is the closure of Σ under composition.[1] The set of all transformations on Q is called the *full transformation semigroup* on Q. One can view (Q, Σ) as a description of S. Since every element s of a semigroup S defines a transformation $x \mapsto x \cdot s$ on $S^1 = S \cup \{1\}$, every semigroup S admits such a description (S^1, S); here, 1 either denotes the neutral element of S or, if S does not have a neutral element, we add 1 as a new neutral element. Essentially, the description (S^1, S) is nothing but the multiplication table for S. On the other hand, there are cases where a description as a transformation semigroup is much more succinct than the multiplication table. For instance, the full transformation semigroup on Q can be generated by a set Σ with three elements; see e.g. [9]. In addition to the size of S, it would be interesting to know which other properties could be derived from the number of states.

Green's relations are an important tool for analyzing the structure of a semigroup S. They are defined as follows:

$$s \leqslant_{\mathcal{R}} t \text{ if } sS^1 \subseteq tS^1, \quad s \leqslant_{\mathcal{L}} t \text{ if } S^1 s \subseteq S^1 t, \quad s \leqslant_{\mathcal{J}} t \text{ if } S^1 s S^1 \subseteq S^1 t S^1.$$

We write $s \mathcal{R} t$ if both $s \leqslant_{\mathcal{R}} t$ and $t \leqslant_{\mathcal{R}} s$; and we set $s <_{\mathcal{R}} t$ if $s \leqslant_{\mathcal{R}} t$ but not $s \mathcal{R} t$. The relations $\mathcal{L}$, $<_{\mathcal{L}}$, $\mathcal{J}$ and $<_{\mathcal{J}}$ are defined analogously. The relations $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{J}$ form equivalence relations. The equivalence classes corresponding to these relations are called $\mathcal{R}$-*classes* (resp. $\mathcal{L}$-*classes*, $\mathcal{J}$-*classes*). Instead of ideals, one could alternatively also use reachability in the right (resp. left, two-sided) Cayley graph of S for defining $\leqslant_{\mathcal{R}}$ (resp. $\leqslant_{\mathcal{L}}$, $\leqslant_{\mathcal{J}}$). We note that $s <_{\mathcal{R}} t$ implies $s <_{\mathcal{J}} t$ and, symmetrically, $s <_{\mathcal{L}} t$ implies $s <_{\mathcal{J}} t$. The complexity of deciding Green's relations

[1] When introducing transformation semigroups in terms of actions, this is the framework of *faithful* actions.

for transformation semigroups was recently shown to be PSPACE-complete [3]. When considering a transformation semigroup on n states, one of our first results shows that the maximal number of $\mathcal{J}$-classes is in $n^{\Theta(n)}$. In particular, the maximal number of equivalence classes is in the same order of magnitude as the size of the transformation semigroup. Since every $\mathcal{J}$-class contains at least one $\mathcal{R}$- and one $\mathcal{L}$-class, the same bound holds for $\mathcal{R}$ and $\mathcal{L}$.

Another important parameter is the maximal length ℓ such that there are elements $s_1, \ldots, s_\ell$ with $s_1 >_\mathcal{R} \cdots >_\mathcal{R} s_\ell$, called the *$\mathcal{R}$-height*. Similarly, we are interested in the $\mathcal{L}$- and $\mathcal{J}$-height. Many semigroup constructions such as the Rhodes expansion and variants thereof rely on this parameter; see e.g. [4, 5, 8]. We show that the maximal $\mathcal{R}$-height is in $2^{\Theta(n)}$; for the maximal $\mathcal{L}$-height and $\mathcal{J}$-height we only have $2^{\Omega(n)}$ as a lower bound. Proving the lower bounds for a fixed number of generators is much more involved than for arbitrarily many generators. The exponential lower bounds are quite unexpected in the following sense: if the transformation semigroup is small, then the number of equivalence classes (and hence, the lengths of chains) cannot be big. On the other hand, the transformation semigroup is maximal if it is full. And an equivalence class in the full transformation semigroup only depends on the number of states in the image; this is because we can apply arbitrary permutations. In particular, the number of equivalence classes in these two extreme cases is small.

There is a tight connection between deterministic automata and transformation semigroups. Roughly speaking, a transformation semigroup is an automaton without initial and final states. The main difference is that for automata, one usually is interested in the syntactic semigroup rather than the transformation semigroup; the syntactic semigroup is the transformation semigroup of the minimal automaton. We show that the above bounds on the number of equivalence classes and heights also apply to syntactic semigroups.

Theorem 1. *For each $n \in \mathbb{N}$, there exists a minimal automaton $\mathcal{A}_n$ with n states over an alphabet of size 5 such that the number of $\mathcal{J}$-classes (resp. $\mathcal{R}$-classes, $\mathcal{L}$-classes) of the transformation semigroup $T(\mathcal{A}_n)$ is at least $(n-4)^{n-4}$.*

Theorem 2. *There exists a sequence of minimal automata $(\mathcal{A}_n)_{n\in\mathbb{N}}$ over a fixed alphabet such that $\mathcal{A}_n$ has n states and the $\mathcal{R}$-height (resp. $\mathcal{L}$-height, $\mathcal{J}$-height) of the transformation semigroup $T(\mathcal{A}_n)$ is in $\Omega(2^n/n^{9.5})$.*

This is the full and extended version of a conference publication at the *12th International Computer Science Symposium in Russia* (CSR 2017) [7].

2 Preliminaries

A *semigroup* is a set S equipped with an associative operation $\cdot\colon S \times S \to S$. We usually write $s_1 s_2$ instead of $s_1 \cdot s_2$. A *subsemigroup* of S is a subset T such that $s_1 s_2 \in T$ for all $s_1, s_2 \in T$. It is called *completely isolated* if the converse

implication holds, i.e., $s_1 s_2 \in T$ implies $s_1 \in T$ and $s_2 \in T$ for all $s_1, s_2 \in S$. The *opposite* semigroup of S is obtained by replacing the operation with its left-right dual $\circ\colon S \times S \to S$ defined by $x \circ y = y \cdot x$. A semigroup S is *generated* by a set $\Sigma \subseteq S$ if every element of S can be written as a product of elements from Σ. It is *m-generated* if there exists a set of cardinality at most m generating S. A 1-generated semigroup is called *cyclic*.

In general, Green's relations in a subsemigroup T of S do not coincide with the corresponding relations in S. However, if T is a completely isolated subsemigroup, the following property holds:

Proposition 3. *Let S be a semigroup and let T be a completely isolated subsemigroup of S. Let $\mathcal{K}$ be one of the relations $\leqslant_{\mathcal{R}}$, $\leqslant_{\mathcal{L}}$, $\leqslant_{\mathcal{J}}$, $<_{\mathcal{R}}$, $<_{\mathcal{L}}$, $<_{\mathcal{J}}$, $\mathcal{R}$, $\mathcal{L}$ or $\mathcal{J}$. Then, for all $x, y \in T$, we have $x \mathcal{K} y$ in S if and only if $x \mathcal{K} y$ in T.*

Proof. We only prove the statement for the preorder $\leqslant_{\mathcal{R}}$. For $\mathcal{R}$ and $<_{\mathcal{R}}$, the statement follows immediately and for the remaining relations, one can use symmetric arguments.

For the implication from right to left, we have $xS^1 \subseteq xT^1S^1 \subseteq yT^1S^1 \subseteq yS^1$. For the converse implication, suppose that $xS^1 \subseteq yS^1$, i.e., there exists some $z \in S^1$ such that $yz = x$. Since T is completely isolated, we have $z \in T^1$, which yields $zT^1 \subseteq T^1$ and thus, $xT^1 = yzT^1 \subseteq yT^1$. □

An *$\mathcal{R}$-chain* is a sequence $(s_1, \ldots, s_\ell)$ of elements of S such that $s_{i+1} <_{\mathcal{R}} s_i$ for all $i \in \{1, \ldots, \ell - 1\}$; ℓ is called the *length* of the $\mathcal{R}$-chain. The maximal length of an $\mathcal{R}$-chain of S is called the *$\mathcal{R}$-height* of S. The notions $\mathcal{L}$-chains, $\mathcal{J}$-chains, $\mathcal{L}$-height and $\mathcal{J}$-height are defined analogously.

A *partial transformation* on a set Q is a partial function $f\colon Q \to Q$. If the domain of f is all of Q, i.e., if f is a total function, f is called a *transformation*. A partial transformation $f\colon Q \to Q$ is called *injective* if $f(p) \neq f(q)$ whenever $p \neq q$ and both $f(p)$ and $f(q)$ are defined. The elements of Q are often called *states*. In the following, we use the notation $q \cdot f$ instead of $f(q)$ to denote the image of an element $q \in Q$ under f. For $R \subseteq Q$ let $R \cdot f = \{q \cdot f \mid q \in R\}$. Note that for all subsets $R \subseteq Q$ and all partial transformations $f\colon Q \to Q$, the inequality $|R \cdot f| \leqslant |R|$ holds; we will implicitly use this property throughout the paper. The *composition* fg of two transformations $f\colon Q \to Q$ and $g\colon Q \to Q$ is defined by $q \cdot fg = (q \cdot f) \cdot g$. The composition is associative.

The set of all partial transformations (resp. transformations) on a fixed set Q forms a semigroup with composition as the binary operation. It is called the *full partial transformation semigroup* (resp. *full transformation semigroup*) on Q. Subsemigroups of full (partial) transformation semigroups are called *(partial) transformation semigroups* and are often specified in terms of generators. Partial transformation semigroups and transformation semigroups are strongly related. On one side, every

transformation semigroup also is a partial transformation semigroup. In the other direction a slightly weaker statement holds:

Proposition 4. *Let P be an m-generated partial transformation semigroup on n states. Then there exists an m-generated transformation semigroup on $n+1$ states which is isomorphic to P.*

Proof. Let P be a partial transformation semigroup on a finite set Q, generated by a set Σ. Let p be a new element not in Q. We extend the elements of Σ to transformations on $Q \cup \{p\}$ by letting $q \cdot a = p$ whenever $q \cdot a$ is undefined in P. In particular, we let $p \cdot a = p$ for all $a \in \Sigma$. By construction, the transformation semigroup generated by this extended set of generators is isomorphic to P. □

A partial transformation semigroup is called *injective* if it is generated by a set of injective partial transformations. An important property of injective partial transformation semigroups is that they have a left-right dual:

Proposition 5. *The opposite semigroup of an injective partial transformation semigroup is a partial transformation semigroup.*

Proof. Let P be a partial transformation semigroup on a set of states Q, generated by a set of injective transformations Σ. Since the composition of injective partial transformations is injective, every element of P is injective. For each $f \in P$ we let $\overline{f}\colon Q \to Q$ be the partial transformation that is undefined on $Q \setminus (Q \cdot f)$ and defined by $(q \cdot f) \cdot \overline{f} = q$ for all $q \in Q$. It is well-defined because f is injective. Moreover, it is easy to check that $\overline{fg} = \overline{g}\,\overline{f}$ for all $f, g \in P$.

Let $\overline{P}$ be the partial transformation semigroup generated by $\overline{\Sigma} = \{\overline{a} \mid a \in \Sigma\}$. Then, for all $f, g, h \in P$, we have $fg = h$ in P if and only if $\overline{g}\,\overline{f} = \overline{h}$ in $\overline{P}$, which shows that $\overline{P}$ is isomorphic to the opposite semigroup of P. □

Transformation semigroups naturally arise when considering deterministic finite automata. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton. Then, each letter $a \in \Sigma$ can be interpreted as a transformation $a\colon Q \to Q$ where $q \cdot a = \delta(q, a)$. The transformation semigroup on Q generated by all letters in Σ is denoted by $T(\mathcal{A})$ and it is called the *transition semigroup* of $\mathcal{A}$. Conversely, given a transformation semigroup T on a finite set Q and a finite set of generators Σ, for each $q_0 \in Q$ and $F \subseteq Q$, one can define a deterministic finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where $\delta\colon Q \times \Sigma \to Q$ is defined as $\delta(q, a) = q \cdot a$.

A well-known approach for translating bounds on the size of a transformation semigroup to syntactic monoids is to *make* an automaton minimal. This can be done by introducing a new generator c with $q_i \cdot c = q_{i+1}$ for $Q = \{q_1, \ldots, q_n\}$ and $q_{n+1} = q_1$; moreover, one chooses some arbitrary state to be both initial and final. We adapt this construction to also work with Green's relations.

Proposition 6. *Let T be an m-generated transformation semigroup on n states. Then there exists a minimal $(n+1)$-state deterministic finite automaton $\mathcal{A}$ over an alphabet of size $m+1$ such that T is a completely isolated subsemigroup of $T(\mathcal{A})$.*

Proof. Let T be a transformation semigroup on a set of states $Q = \{q_1, \ldots, q_n\}$, generated by Σ. Let $\mathcal{A} = (Q \cup \{q_0\}, \Sigma \cup \{c\}, \delta, q_0, \{q_n\})$ be the automaton defined by $\delta(q_0, a) = q_0$ and $\delta(q_i, a) = q_i \cdot a$ for $i \geqslant 1$ and all $a \in \Sigma$. The transitions for the letter c are defined by $\delta(q_i, c) = q_{i+1}$ for $i < n$ and $\delta(q_n, c) = q_1$. This automaton is minimal: for two different states $q_i, q_j \in Q \cup \{q_0\}$ with $i > j$, we have $\delta(q_i, c^{n-i}) = q_n$ but $\delta(q_j, c^{n-i}) \neq q_n$.

By construction, T is a subsemigroup of $T(\mathcal{A})$. To see that T is completely isolated within $T(\mathcal{A})$, note that we have $\delta(q_0, u) = q_0$ if and only if $u \in \Sigma^*$. □

3 Bounds for the Number of Classes

Let $\mathcal{K}$ be any of the relations $\mathcal{R}$, $\mathcal{L}$ or $\mathcal{J}$. The naïve upper bound for the number of $\mathcal{K}$-classes of a transformation semigroup T on n states is given by the size of T itself. Since there are n^n different functions from Q to Q, the semigroup T contains at most n^n elements. It is well known that this bound is tight even for a constant number of generators, since for each $n \geqslant 1$ there exists a transformation semigroup of size n^n generated by a set Σ with three elements; see e.g. [9].

As each $\mathcal{R}$-class (resp. $\mathcal{L}$-class, $\mathcal{J}$-class) consists of at least one element, the number of such classes is also bounded by n^n. We now show that this upper bound is tight up to a constant factor.

Proposition 7. *Let T be an m-generated transformation semigroup on n states. Then there exists an $(m+1)$-generated transformation semigroup on $n+3$ states which has at least $|T|$ different $\mathcal{J}$-classes.*

Proof. Let T be a semigroup of transformations on a set of states Q, generated by a finite set Σ, and let q_0 be an arbitrary element from Q. Let q_1, q_2, q_3 be new states not in Q and let c be a new generator not in Σ. Let U be the transformation semigroup on $Q \cup \{q_1, q_2, q_3\}$ obtained by extending the transformations of T as follows: for each $a \in \Sigma$ and $q \in Q$, let $q \cdot c = q$, $q_1 \cdot a = q_3 \cdot a = q_3 \cdot c = q_0$, $q_1 \cdot c = q_2 \cdot a = q_2$, and $q_2 \cdot c = q_3$.

Let $u, v \in \Sigma^*$ be different elements of T. Then cuc and cvc are different in U. We claim that $cuc \not\leqslant_{\mathcal{J}} cvc$ in U. For the sake of contradiction, suppose that there exist $x, y \in (\Sigma \cup \{c\})^*$ such that $cuc = xcvcy$ in U. Clearly, $q_1 \cdot cuc = q_3 \notin Q$. Moreover, at least one of the words x or y must be non-empty and therefore $q_1 \cdot xcucy \in Q$. This shows that $cuc \neq xcvcy$, as desired. □

Combining the result with statements from the previous section, we obtain a lower bound for the number of $\mathcal{J}$-classes of the transition semigroup of an automaton.

Proof of Theorem 1. As we mentioned before, it is well known that there exists a 3-generator transformation semigroup on n states of size n^n. If we first apply Proposition 7 and then Proposition 6 to T, we obtain the claim by Proposition 3. The statement extends to $\mathcal{R}$-classes (resp. $\mathcal{L}$-classes) because each $\mathcal{J}$-class contains at least one $\mathcal{R}$-class (resp. $\mathcal{L}$-class). □

4 Bounds for the Length of Chains

Let $\mathcal{K}$ be any of the relations $\mathcal{R}$, $\mathcal{L}$ or $\mathcal{J}$. As for the number of $\mathcal{K}$-classes, the naïve upper bound for the length of $\mathcal{K}$-chains is given by the maximal size n^n of the transformation semigroup on n states. In this section, we improve this upper bound for $\mathcal{R}$-chains and later give a lower bound that matches up to a polynomial gap. For $\mathcal{L}$-chains, slightly weaker results are obtained.

Lemma 8. *Let P be a partial transformation semigroup on a finite set Q of cardinality n. Let $x, y \in P$ such that $Q \cdot x = Q \cdot xy$. Then $x \mathcal{R}\, xy$.*

Proof. Let $\omega = n!$ and let $z = y^{\omega-1}$. It suffices to show that $xyz = x$ in P, i.e., for all $q \in Q$, we have $q \cdot x = q \cdot xyz$. By assumption, the restriction of y to the set $Q \cdot x$ is bijective. Thus, the mapping y^ω acts as identity on $Q \cdot x$. This yields $q \cdot xyz = q \cdot xy^\omega = (q \cdot x) \cdot y^\omega = q \cdot x$. □

Proposition 9. *Let P be a partial transformation semigroup on n states. Then the $\mathcal{R}$-height of P is at most 2^n.*

Proof. Let P be a partial transformation semigroup on a set of states Q with $|Q| = n$. Let $(u_1, u_2, \ldots, u_\ell)$ be an $\mathcal{R}$-chain of P. We show that all sets $Q \cdot u_i$ must be pairwise distinct which yields the desired bound. Suppose that $Q \cdot u_i = Q \cdot u_j$ for $1 \leqslant i < j \leqslant \ell$. Since $u_j <_{\mathcal{R}} u_i$, there exists $v \in P$ with $u_i v = u_j$. Lemma 8 yields $u_j \mathcal{R}\, u_i$ which is a contradiction. □

A similar technique can be applied for $\mathcal{L}$-chains. However, it does not yield an exponential upper bound. As before, let P be a partial transformation semigroup on a finite set Q. For a partial transformation $x \in P$ and $p, q \in Q$, we write $p \sim_x q$ if $p \cdot x = q \cdot x$. It is easy to see that $\sim_x$ is an equivalence relation on Q.

Lemma 10. *Let $x, y \in P$. If $\sim_x$ and $\sim_{yx}$ coincide, then all the relations $\sim_{y^i x}$ with $i \geqslant 0$ coincide.*

Proof. We will show that each of the relations $\sim_{y^i x}$ coincides with $\sim_x$. For $i = 0$, the statement is trivial. Let $i \geqslant 1$ and let $p, q \in Q$. Note that by definition, $p \sim_{y^i x} q$ if and only if $p \cdot y \sim_{y^{i-1} x} q \cdot y$. By induction, the latter holds if and only if $p \cdot y \sim_x q \cdot y$ which is equivalent to $p \sim_{yx} q$ by the definition of $\sim_{yx}$ and also equivalent to $p \sim_x q$ by the premise that $\sim_{yx}$ and $\sim_x$ coincide. □

Lemma 11. *Let P be a partial transformation semigroup on a finite set Q of cardinality n. Let $x, y \in P$ such that $\sim_x$ and $\sim_{yx}$ coincide. Then $x \mathcal{L}\, yx$.*

Proof. Let $\omega = n!$ and let $z = y^{\omega-1}$. It suffices to show that $zyx = x$ in P, i.e., for all $q \in Q$, we have $q \sim_x q \cdot zy$. Let $q \in Q$ and let $p = q \cdot y^\omega$. By the definition of p and by idempotence of y^ω, we have $p \cdot y^\omega x = (q \cdot y^\omega) \cdot y^\omega x = q \cdot y^\omega x$, i.e., $p \sim_{y^\omega x} q$. By Lemma 10, we obtain $q \sim_x p = q \cdot y^\omega = q \cdot zy$, thereby proving the claim. □

We denote by B_n the number of different equivalence relations on a n-element set. In the literature, this sequence of numbers $(B_n)_{n \geqslant 1}$ is usually called *Bell numbers*; see e.g. [1, 2].

Proposition 12. *Let P be a partial transformation semigroup on n states. Then the $\mathcal{L}$-height of P is at most $B_n \leqslant \left(\frac{0.792n}{\ln(n+1)}\right)^n$.*

Proof. Suppose that P is a partial transformation semigroup on a set of states Q with $|Q| = n$ and let $(u_1, u_2, \ldots, u_\ell)$ be an $\mathcal{L}$-chain of P. By Lemma 11, the equivalence relations $\sim_{u_i}$ must be pairwise different. If any two of these equivalence relations are identical we obtain a contradiction using the arguments given in the proof of Proposition 9. The upper bound for B_n can be found in [2]. □

4.1 Token Computations in Transformation Semigroups

In this subsection, we introduce the building blocks for the lower bound on the $\mathcal{R}$-height (resp. $\mathcal{L}$-height, $\mathcal{J}$-height). A *token machine* is a pair (C, I) where C is a finite set and I is a set of partial transformations on C. The elements of the set C are called *cells*, subsets of C are called *configurations* and the generators I are called *instructions*. The token machine is called *injective* if all partial transformations in I are injective.

A *program* is a finite word over the alphabet I and a *computation* is a sequence

$$R_0 \xrightarrow{\iota_1} R_1 \xrightarrow{\iota_2} R_2 \cdots \xrightarrow{\iota_\ell} R_\ell$$

where all $R_i \subseteq C$ have the same cardinality and $R_{i-1} \cdot \iota_i = R_i$. The configuration R_0 is called *initial configuration* and R_ℓ is called the *final configuration* of the computation. The program $\iota_1 \iota_2 \cdots \iota_\ell$ is the *label* of the computation and ℓ is its *length*. It is *progressing* if all configurations appearing in the computation are pairwise distinct and for each $i \in \{1, \ldots, \ell\}$ and each $\iota \in I \setminus \{\iota_i\}$, we have $|R_{i-1} \cdot \iota| < |R_i|$. It is *maximal* if $|R_\ell \cdot \iota| < |R_\ell|$ for all $\iota \in I$.

A language over programs $L \subseteq I^*$ is called *deterministic* on a configuration $R \subseteq C$ if $|R \cdot u_1| = |R| = |R \cdot u_2|$ implies $u_1 = u_2$ for all $u_1, u_2 \in L$. This means that there exists at most one computation which starts at the initial configuration R and is labeled by a program from L.

The focal idea of token machines is captured in the following proposition which states that computations in token machines naturally yield lower bounds for the length of $\mathcal{R}$-chains.

Proposition 13. *Let (C, I) be a token machine and let P be the partial transformation semigroup on C generated by I. If there exists a maximal progressing computation of length ℓ, then the $\mathcal{R}$-height of P is at least ℓ.*

Proof. Let $R_0 \xrightarrow{\iota_1} R_1 \xrightarrow{\iota_2} R_2 \cdots \xrightarrow{\iota_\ell} R_\ell$ be a maximal progressing computation. For each $i \in \{1, \ldots, \ell\}$, we let $u_i = \iota_1 \iota_2 \cdots \iota_i$. It remains to show that $(u_1, \ldots, u_\ell)$ is an $\mathcal{R}$-chain. By definition, we immediately obtain $u_{i+1} \leqslant_{\mathcal{R}} u_i$. Assume, for the sake of contradiction, that $u_i \leqslant_{\mathcal{R}} u_{i+1}$ for some $i \in \{1, \ldots, \ell - 1\}$, i.e., there exists $v \in I^*$ with $u_i = u_{i+1} v$. Without loss of generality, we may assume that i is maximal with this property. If $|v| = 0$, then $R_i = R_0 \cdot u_i = R_0 \cdot u_{i+1} = R_{i+1}$, contradicting the premise of progression. Thus, $|v| \geqslant 1$ and since the computation is progressing and maximal, we have $i < \ell - 1$ and $v = \iota_{i+2} w$ for some $w \in I^*$. This yields $u_{i+2} w \iota_{i+1} = u_{i+1} \iota_{i+2} w \iota_{i+1} = u_{i+1} v \iota_{i+1} = u_i \iota_{i+1} = u_{i+1}$, contradicting the maximality of i. □

4.2 Lower Bounds over a Growing Instruction Set

Before describing the technical ingredients required in our main result, we prove a slightly weaker statement. In contrast to the result presented later, it relies on an alphabet that grows exponentially with the number of elements.

Theorem 14. *For all even $n \in \mathbb{N}$, there exists a token machine with n cells which admits a maximal progressing computation of length at least $\binom{n}{n/2} - 1$.*

Proof. Let $C = \{1, 2, \ldots, n\}$. Let $\ell = \binom{n}{n/2} - 1$ and let $\{R_0, R_1, \ldots, R_\ell\}$ be the set of $n/2$-element subsets of C. For each $i \in \{1, \ldots, \ell\}$, let $\iota_i \colon R_{i-1} \to R_i$ be a bijection. Note that in the context of the present proof, it does not matter which of the $(n/2)!$ bijections is chosen; for example, one can always choose the unique bijection ι_i such that $\iota_i(j) < \iota_i(k)$ if and only if $j < k$. Each ι_i can be viewed as a partial transformation on C which is undefined for all $c \in C \setminus R_{i-1}$. We now show that in the token machine (C, I) with $I = \{\iota_i \mid 1 \leqslant i \leqslant \ell\}$, the sequence

$$R_0 \xrightarrow{\iota_1} R_1 \xrightarrow{\iota_2} R_2 \cdots \xrightarrow{\iota_\ell} R_\ell$$

is a maximal progressing computation. It is a valid computation by the definition of the instructions ι_i. Consider $i \in \{0, \ldots, \ell\}$ and $j \in \{1, 2, \ldots, \ell\} \setminus \{i + 1\}$. Since $R_{j-1} \neq R_i$, the instruction ι_j is undefined on at least one element of R_i and thus, $|R_i \cdot \iota_j| < |R_i|$. This shows that the computation is both progressing and maximal. □

The theorem has a series of interesting consequences which will be outlined in Section 4.4, after proving a stronger variant of the theorem with fixed alphabet.

4.3 Tapes and Binary Counters

A *sub-machine* of a token machine (C, I) is a subset $S \subseteq C$ such that for each configuration R and for each instruction $\iota \in I$ with $|R \cdot \iota| = |R|$, we also have $|(R \cap S) \cdot \iota| = |R \cap S|$. In other words, each computation stays a computation when restricted to S. The *union* of two token machines (C, I) and (C', I') with $C \cap C' = \emptyset$ is the token machine $(C \cup C', I \cup I')$ where the instructions in $I \setminus I'$ are extended to act as identity on C' and the instructions in $I' \setminus I$ are extended to act as identity on C. The cells C and C' of the original machines are sub-machines of the union.

An *n-bit tape* T is a token machine (C, I) with n cells and an arbitrary (but fixed) order $(c_0, c_1, \ldots, c_{n-1})$. One can interpret configurations $R \subseteq C$ as bit strings $b_{n-1}b_{n-2}\cdots b_0$ where $b_i = 1$ if and only if $c_i \in R$ and $b_i = 0$ otherwise, and think of T as a ring buffer with a read/write head at position 0. An instruction ι^T_{rotl} can be used to move the tape head to the right (or, actually, retain the head position but left-rotate the buffer). For each $i \in \{0, \ldots, n-2\}$, we let $c_i \cdot \iota^T_{\mathsf{rotl}} = c_{i+1}$ and $c_{n-1} \cdot \iota^T_{\mathsf{rotl}} = c_0$. The instruction ι^T_{rotr} is defined analogously and moves in the opposite direction. An instruction $\iota^T_{=\mathsf{0}}$ can be used to check whether the head is scanning a zero and halt the program otherwise. It is undefined on c_0 and defined as the identity on $\{c_1, \ldots, c_{n-1}\}$. Conversely, the ι^T_{sync} instruction is defined as the identity on c_0 and undefined on every other cell. An instruction ι^T_{mvl} maps c_0 to c_1, acts as the identity on $\{c_2, c_3, \ldots, c_{n-1}\}$ and is undefined on c_1. Analogously, ι^T_{mvr} maps c_0 to c_{n-1}, acts as the identity on $\{c_1, c_2, \ldots, c_{n-2}\}$ and is undefined on c_{n-1}. The *value* of T under a configuration R is $\sum_{c_i \in R} 2^i$.

An *n-bit binary counter* N is constructed as follows. Three new n-bit tapes S, T and $\overline{T}$ are introduced. Their cells are $(d_0, d_1, \ldots, d_{n-1})$, $(c_0, c_1, \ldots, c_{n-1})$ and $(\overline{c}_0, \overline{c}_1, \ldots, \overline{c}_{n-1})$, respectively. Then, the union of S, T and $\overline{T}$ is constructed and the following instructions are added:

- $\iota^N_{\mathsf{rotl}} = \iota^T_{\mathsf{rotl}}\iota^{\overline{T}}_{\mathsf{rotl}}\iota^S_{\mathsf{rotl}}$,
- $\iota^N_{=\mathsf{0}} = \iota^T_{=\mathsf{0}}$,
- $\iota^N_{\mathsf{sync}} = \iota^S_{\mathsf{sync}}$,
- $\iota^N_{\mathsf{rotr}} = \iota^T_{\mathsf{rotr}}\iota^{\overline{T}}_{\mathsf{rotr}}\iota^S_{\mathsf{rotr}}$,
- $\iota^N_{=\mathsf{1}} = \iota^{\overline{T}}_{=\mathsf{0}}$,
- $\iota^N_{\mathsf{off}} = \iota^S_{=\mathsf{0}}$,
- ι^N_{inc} with $\overline{c}_0 \cdot \iota^N_{\mathsf{inc}} = c_0$ and $c_0 \cdot \iota^N_{\mathsf{inc}}$ undefined and $c \cdot \iota^N_{\mathsf{inc}} = c$ for all $c \notin \{c_0, \overline{c}_0\}$,
- ι^N_{dec} with $c_0 \cdot \iota^N_{\mathsf{dec}} = \overline{c}_0$ and $\overline{c}_0 \cdot \iota^N_{\mathsf{dec}}$ undefined and $c \cdot \iota^N_{\mathsf{dec}} = c$ for all $c \notin \{c_0, \overline{c}_0\}$.

Following this, the original instructions of S, T and $\overline{T}$ are removed from I. Thus, a binary counter provides exactly eight instructions. A configuration R of N is *valid* if $|R \cap S| = 1$ and for each $i \in \{0, \ldots, n-1\}$, we have $c_i \in R$ if and only if $\overline{c}_i \notin R$.

Lemma 15. *Let R be a valid configuration of a binary counter N and let $u \in I^*$. Then $R \cdot u$ is a valid configuration of N if and only if $|R \cdot u| = |R|$.*

Proof. If N is an n-bit binary counter, the cardinality of every valid configuration of N is $n + 1$. This shows the implication from left to right. For the converse direction, it suffices to prove that the action of instructions on R preserves validity. The statement then follows by induction on the length of u.

The instructions ι^N_{rotl} and ι^N_{rotr} cyclically rotate the tapes T, $\overline{T}$ and S. Thus, if R is valid, then $R \cdot \iota^N_{\mathsf{rotl}}$ and $R \cdot \iota^N_{\mathsf{rotr}}$ are valid as well.

For each $\iota \in \left\{\iota^N_{=0}, \iota^N_{=1}, \iota^N_{\mathsf{sync}}, \iota^N_{\mathsf{off}}\right\}$, we have either $|R \cdot \iota| < |R|$ or $R \cdot \iota = R$.

If $\left|R \cdot \iota^N_{\mathsf{inc}}\right| = |R|$, then R does not contain c_0. If, moreover, R is a valid configuration, it contains $\overline{c}_0$. But then, $R \cdot \iota^N_{\mathsf{inc}}$ contains c_0 and does not contain $\overline{c}_0$. It coincides with R on all other cells. Thus, $R \cdot \iota^N_{\mathsf{inc}}$ is valid as well. By a symmetric argument, the instruction ι^N_{dec} preserves validity. □

We now define three regular languages

$$\begin{aligned} L^N_{\mathsf{reset}} &= \iota^N_{\mathsf{sync}}((\iota^N_{=0} \mid \iota^N_{\mathsf{dec}})\iota^N_{\mathsf{rotr}}\iota^N_{\mathsf{off}})^*(\iota^N_{=0} \mid \iota^N_{\mathsf{dec}})\iota^N_{\mathsf{rotr}}\iota^N_{\mathsf{sync}}, \\ L^N_{\mathsf{inc}} &= \iota^N_{\mathsf{sync}}(\iota^N_{\mathsf{dec}}\iota^N_{\mathsf{rotr}}\iota^N_{\mathsf{off}})^*\iota^N_{\mathsf{inc}}(\iota^N_{\mathsf{off}}\iota^N_{\mathsf{rotr}})^*\iota^N_{\mathsf{sync}} \text{ and} \\ L^N_{\mathsf{dec}} &= \iota^N_{\mathsf{sync}}(\iota^N_{\mathsf{inc}}\iota^N_{\mathsf{rotr}}\iota^N_{\mathsf{off}})^*\iota^N_{\mathsf{dec}}(\iota^N_{\mathsf{off}}\iota^N_{\mathsf{rotr}})^*\iota^N_{\mathsf{sync}}. \end{aligned}$$

Before giving the intuition behind these languages and their actions on valid configurations of the binary counter, we show that for each valid configuration, there exists at most one word in each of the three languages that preserves validity.

Lemma 16. *The languages L^N_{reset}, L^N_{inc} and L^N_{dec} are deterministic on all valid configurations.*

Proof. Suppose there are two different words $u_1, u_2 \in L^N_{\mathsf{reset}}$ and a valid configuration R such that $|R \cdot u_1| = |R|$. Since L^N_{reset} is prefix-free, there exist a unique program $p \in I^*$ and different instructions $\iota_1, \iota_2 \in I$ such that $u_1 \in p\iota_1 I^*$ and $u_2 \in p\iota_2 I^*$. A careful analysis of the structure of the regular expression for L^N_{reset} shows that either $\{\iota_1, \iota_2\} = \left\{\iota^N_{=0}, \iota^N_{\mathsf{dec}}\right\}$ or $\{\iota_1, \iota_2\} = \left\{\iota^N_{\mathsf{off}}, \iota^N_{\mathsf{sync}}\right\}$.

In the first case, we may assume without loss of generality that $\iota_1 = \iota^N_{=0}$ and $\iota_2 = \iota^N_{\mathsf{dec}}$. From $|R \cdot p\iota_1| = |R \cdot p|$, we deduce $c_0 \notin R \cdot p$ because $\iota^N_{=0}$ is undefined on c_0. This implies $\overline{c}_0 \in R \cdot p$ since $R \cdot p$ is a valid configuration by Lemma 15. Since ι^N_{dec} is undefined on $\overline{c}_0$, it follows that $|R \cdot u_2| \leqslant |R \cdot p\iota_2| < |R \cdot p| \leqslant |R|$.

In the second case, we may assume that $\iota_1 = \iota^N_{\mathsf{off}}$ and $\iota_2 = \iota^N_{\mathsf{sync}}$. Since $|R \cdot p\iota_1| = |R \cdot p|$ and since ι^N_{off} is undefined on d_0, we have $d_0 \notin R \cdot p$. This implies $d_i \in R \cdot p$ for some $i \in \{1, \ldots, n-1\}$ because $R \cdot p$ is valid by Lemma 15. The instruction ι^N_{sync} is undefined on $\{d_1, d_2, \ldots, d_{n-1}\}$ which yields $|R \cdot u_2| < |R|$, as above.

The proofs for L^N_{inc} and L^N_{dec} follow by a similar reasoning. □

Let R be a configuration of N. We say that the counter is *synchronized* under R if $d_0 \in R$. The *value* of N under R is the value of T under $R \cap \{c_0, c_1, \ldots, c_{n-1}\}$.

The intuition behind the three languages defined above is as follows. Suppose that R is a valid configuration such that N is synchronized under R. Then there exists a unique program from L^N_{reset} which, again, brings the counter into a valid and synchronized state while changing the counter value to 0. Equivalently, there

is a program from L^N_{inc} that increases the counter value by 1 and a program from L^N_{dec} that decreases the counter value by 1. This idea is formalized in the following lemmas.

Lemma 17. *Let R be a valid configuration and let $u \in L^N_{\mathsf{reset}}$ such that $|R \cdot u| = |R|$. Then, under $R \cdot u$, the counter is synchronized and its value is zero.*

Proof. It is easy to see that each word $u \in L^N_{\mathsf{reset}}$ with $|R \cdot u| = |R|$ cyclically rotates the three tapes of N exactly n times and after each cyclic rotation, either $\iota^N_{=0}$ or ι^N_{dec} is applied. The codomains of both $\iota^N_{=0}$ and ι^N_{dec} do not contain c_0 and thus, we have $R \cdot u \cap \{c_0, c_1, \ldots, c_{n-1}\} = \emptyset$ which is equivalent to saying that the value under $R \cdot u$ is zero. To see that the counter is synchronized, note that applying ι^N_{sync} to a valid configuration preserves the number of elements if and only if the configuration is synchronized. □

Lemma 18. *Let R be a valid configuration and let $u \in L^N_{\mathsf{inc}}$ such that $|R \cdot u| = |R|$. If v is the value of the counter under R and v' is its value under $R \cdot u$, we have $v' = v + 1 \leqslant 2^n - 1$.*

Proof. Let us first assume that $v < 2^n - 1$. Let $i \in \{0, \ldots, n-1\}$ be minimal such that $c_i \notin R$ and let

$$w = \iota^N_{\mathsf{sync}}(\iota^N_{\mathsf{dec}}\iota^N_{\mathsf{rotr}}\iota^N_{\mathsf{off}})^i \iota^N_{\mathsf{inc}}(\iota^N_{\mathsf{off}}\iota^N_{\mathsf{rotr}})^{n-i}\iota^N_{\mathsf{sync}}.$$

We claim that $u = w$. By Lemma 16, it suffices to show that $|R \cdot w| = |R|$. Let us first investigate the instructions operating on S. The word starts with an ι^N_{sync} instruction, each ι^N_{off} instruction is applied after R has been rotated cyclically 1 to $n-1$ times and the second ι^N_{sync} instruction is applied after exactly n cyclic rotations. We deduce $\left|R \cdot \iota^N_{\mathsf{sync}}\right| = |R|$ from $|R \cdot u| = |R|$, and thus, the counter is synchronized on both R and on the configuration reached before the last ι^N_{sync} instruction. Moreover, whenever a ι^N_{off} instruction is applied to a configuration R', we have $d_i \in R'$ for some $i \in \{1, \ldots, n-1\}$. Note that the case $v = 2^n - 1$ can be excluded since in order for the ι^N_{inc} instruction to preserve the number of elements in the configuration, it would have to be preceded by at least n $\iota^N_{\mathsf{rotr}}\iota^N_{\mathsf{off}}$-factors and one of those factors would reduce the number of elements.

The instruction ι^N_{dec} is applied exactly once before each of the first i cyclic rotations. Since $\{c_0, c_1, \ldots, c_{i-1}\} \subseteq R$, we have $c_0 \in R \cdot \iota^N_{\mathsf{sync}}(\iota^N_{\mathsf{dec}}\iota^N_{\mathsf{rotr}}\iota^N_{\mathsf{off}})^j$ for all $j \in \{0, \ldots, i-1\}$. Moreover, since $c_i \notin R$, we have $c_0 \notin R \cdot \iota^N_{\mathsf{sync}}(\iota^N_{\mathsf{dec}}\iota^N_{\mathsf{rotr}}\iota^N_{\mathsf{off}})^i$ which implies $\overline{c}_0 \in R \cdot \iota^N_{\mathsf{sync}}(\iota^N_{\mathsf{dec}}\iota^N_{\mathsf{rotr}}\iota^N_{\mathsf{off}})^i$ by Lemma 15. Consequently, the occurrences of ι^N_{dec} and ι^N_{inc} in w do not reduce the number of elements in the configuration. The above observations also show that

$$R \cdot u = R \cdot w = \{c_i\} \cup (R \cap \{c_{i+1}, c_{i+2}, \ldots, c_{n-1}\})$$

which is equivalent to the claim $v' = v + 1$. □

For the ι^N_{dec} instruction, a symmetric version of the lemma holds.

Lemma 19. *Let R be a valid configuration and let $u \in L^N_{\mathsf{dec}}$ such that $|R \cdot u| = |R|$. If v is the value of the counter under R and v' is its value under $R \cdot u$, we have $v' = v - 1 \geqslant 0$.*

Before moving on to the main construction, let us describe how a counter can be extended such that for any fixed constant $k \in \{0, \ldots, 2^n - 1\}$ one can test whether the counter value equals k. To this end, we complement the eight basic counter instructions by a new instruction $\iota^N_{\mathsf{val}=\mathsf{k}}$ as follows. For each $i \in \{0, \ldots, n-1\}$ with $k \bmod 2^{i+1} \geqslant 2^i$, we let $c_i \cdot \iota^N_{\mathsf{val}=\mathsf{k}} = c_i$ and let $\overline{c}_i \cdot \iota^N_{\mathsf{val}=\mathsf{k}}$ be undefined. Symmetrically, we let $\overline{c}_i \cdot \iota^N_{\mathsf{val}=\mathsf{k}} = \overline{c}_i$ and $c_i \cdot \iota^N_{\mathsf{val}=\mathsf{k}}$ undefined if $k \bmod 2^{i+1} < 2^i$. By construction, applying this new instruction to a valid configuration R preserves the cardinality of the configuration if and only if the value of N under R is k.

4.4 Main Result

Let $n \in \mathbb{N}$ be an even number. Let T be an n-bit tape with cells $(t_0, t_1, \ldots, t_{n-1})$. The union of T with three $\lceil \log_2 n \rceil$-bit counters P, Q and Z forms a token machine, henceforth referred to as U. A configuration of U is *valid* if it is valid when restricted to each of the three counters.

Informally, the idea of our construction is the following: as in the proof of Theorem 14, we enumerate all $n/2$-element subsets of an n-element set on the tape T. In order to do so with a constant number of generators, this enumeration needs to be done in a very specific way. We say that a word $Y \in \{0,1\}^*$ is a *successor* of $X \in \{0,1\}^*$ if there exist $p \in \{0,1\}^*$, $i \geqslant 1$ and $j \geqslant 0$ such that $X = p01^i0^j$ and $Y = p10^{j+1}1^{i-1}$. For each $m \in \{0, 1, \ldots, n\}$ one can define a sequence of bit strings $(X_0, X_1, \ldots, X_\ell)$ as stated in the following lemma:

Lemma 20. *For all $n \in \mathbb{N}$ and $m \in \{0, 1, \ldots, n\}$, there exists a unique sequence $(X_0, X_1, \ldots, X_\ell)$ such that*

- *$X_0 = 0^{n-m}1^m$,*
- *for each $k \in \{1, \ldots, \ell\}$, X_k is a successor of X_{k-1} and*
- *X_ℓ does not have a successor.*

The terms of this sequence are pairwise distinct, each term contains exactly m occurrences of the letter 1, and we have $\ell = \binom{n}{m}$ as well as $X_\ell = 1^m0^{n-m}$.

Proof. First observe that if a word $X \in \{0,1\}^*$ can be factorized as $X = p01^i0^j$ with $p \in \{0,1\}^*$ and $i \geqslant 1$ and $j \geqslant 0$, then this factorization is unique. As a consequence, the sequence defined above is unique and its terms are pairwise distinct. It is also easy to see that if Y is a successor of X, then X and Y contain the same number of 1's. The remaining two properties $\ell = \binom{n}{m}$ and $X_\ell = 1^m0^{m-n}$ clearly hold if $n = 0$ or $m \in \{0, n\}$.

We now assume $n \geqslant 1$, as well as $m \in \{1, \ldots, n-1\}$, and proceed by induction on n. Let $s \in \{0, \ldots, n\}$ such that $X_0, X_1, \ldots, X_s \in 0\{0,1\}^{n-1}$ and $X_{s+1}, X_{s+2}, \ldots, X_\ell \in 1\{0,1\}^{n-1}$. Applying the induction hypothesis to the suffixes of length $n-1$ of X_0, X_1, …, X_s, we know that $s = \binom{n-1}{m}$ and $X_s = 01^m0^{(n-1)-m}$. This yields $X_{s+1} = 10^{n-m}1^{m-1}$ and by applying induction again to the suffixes of X_{s+1}, X_{s+2}, …, X_ℓ, we obtain $\ell - s = \binom{n-1}{m-1}$ as well as $X_\ell = 11^{m-1}0^{(n-1)-(m-1)} = 1^m0^{n-m}$. Note that by Pascal's rule, $\ell = \ell - s + s = \binom{n-1}{m-1} + \binom{n-1}{m} = \binom{n}{m}$ which concludes the proof. □

Note that the sequence corresponds to binary counting and deleting all counter values not having m bits equal 1. Since we are interested in enumerating $n/2$-element subsets, we only consider the case $m = n/2$. Interpreting the bit strings X_k as $n/2$-element subsets of an n-element set, the sequence $(X_0, X_1, \ldots, X_\ell)$ describes our enumeration order. Thus, all configurations appearing in the computation always contain $n/2$ elements when restricted to T.

The counter P keeps track of the position of the head on T. It is needed for moving a block of 1-bits as far to the right as possible when transitioning from X_{k-1} to X_k. The volatile counters Q and Z are only used by the following macro that checks whether the bit below the tape head of T is 1.

$$L_{=1} = \iota^T_{\mathsf{rotr}}((\varepsilon \mid \iota^T_{=0}L^Z_{\mathsf{inc}})\iota^T_{\mathsf{rotr}}L^Q_{\mathsf{inc}})^*\iota^Q_{\mathsf{val=n-1}}\iota^Z_{\mathsf{val=n/2}}L^Q_{\mathsf{reset}}L^Z_{\mathsf{reset}}.$$

Roughly speaking, a program from $L_{=1}$, which preserves the cardinality of the configuration, rotates the tape T cyclically n times. The counter Q is used to ensure that neither more nor less rotations are performed. After each rotation, except for the last one, the counter Z is increased non-deterministically if the bit under the tape head is 0. Then, the value of Z is checked to be exactly $n/2$. Since we know that the number of 0-bits on T is $n/2$ and since the bit under the tape head cannot contribute to the value of Z, this is only possible if the bit under the tape head is set. More precisely, the following lemma holds.

Lemma 21. *Let R be a valid configuration such that $|R \cap T| = n/2$, the counters P and Q are synchronized and the values of P and Q are zero. Then there exists a program $u \in L_{=1}$ with $|R \cdot u| = |R|$ if and only if $t_0 \in R$. Moreover, if such a program u exists, it is unique and we have $R \cdot u = R$.*

Proof. For $i \in \{0, 1, \ldots, n-1\}$, let $m_i = 1$ if $t_i \notin R$ and let $m_i = 0$ otherwise.

By Lemma 18, the $\iota^Q_{\mathsf{val=n-1}}$ instruction in a program $w \in L_{=1}$ preserves the number of elements in a valid configuration if and only if w contains exactly $n-1$ occurrences of L^Q_{inc}. Therefore, each word that preserves the number of elements when applied to R contains the instruction ι^N_{rotr} exactly n times. Since each occurrence of L^Z_{inc} is paired with a $\iota^T_{=0}$ instruction, L^Z_{inc} is applied at most m_i times after the i-th rotation,

i.e., every program that does not reduce the number of elements when applied to R has the form

$$\iota^T_{\mathsf{rotr}} \prod_{i=1}^{n-1} ((\iota^T_{=0} L^Z_{\mathsf{inc}})^{k_i} \iota^T_{\mathsf{rotr}} L^Q_{\mathsf{inc}}) \iota^Q_{\mathsf{val=n-1}} \iota^Z_{\mathsf{val=n/2}} L^Q_{\mathsf{reset}} L^Z_{\mathsf{reset}}$$

for some $k_i \in \{0, 1\}$ with $k_i \leqslant m_i$. Moreover, the $\iota^Z_{\mathsf{val=n/2}}$ instruction preserves the cardinality of the configuration if and only if the sum of all k_i with $1 \leqslant i \leqslant n-1$ equals $n/2$. Therefore, any choice of values k_i must also satisfy

$$n/2 = \sum_{i=1}^{n-1} k_i \leqslant \sum_{i=1}^{n-1} m_i = n/2 - m_0$$

where the last equality follows from the assumption that $|R \cap T| = n/2$. This is only possible if $m_0 = 0$, i.e., $t_0 \in R$, and $k_i = m_i$ for all $i \in \{1, 2, \ldots, n-1\}$. By letting $k_i = m_i$ in the program above, we obtain the unique program u such that $|R \cdot u| = |R|$. To see that $R \cdot u = R$, note that after n cyclic rotations, the tape T returns to its original state. Moreover, by Lemma 17, both Q and Z are synchronized and have value zero. □

We also define two auxiliary languages $L_{\mathsf{rotl}} = L^P_{\mathsf{dec}} \iota^T_{\mathsf{rotl}}$ and $L_{\mathsf{rotr}} = L^P_{\mathsf{inc}} \iota^T_{\mathsf{rotr}}$ to perform rotations while keeping the counter P up-to-date.

The language L, which is used to control the main program, is now defined as $L = L^P_{\mathsf{reset}} L^Q_{\mathsf{reset}} L^Z_{\mathsf{reset}} (L_{=1} L_{\mathsf{rotr}})^* \iota^T_{=0} L_{\mathsf{rotl}} (\iota^T_{\mathsf{mvl}} (L_1 \mid L_2 \mid L_3))^* \iota^P_{\mathsf{val=n-1}}$ with

$$\begin{aligned}
L_1 &= (\iota^P_{\mathsf{val=0}} \mid L_{\mathsf{rotl}} \iota^T_{=0} L_{\mathsf{rotr}}) L_{\mathsf{rotr}} (L_{=1} L_{\mathsf{rotr}})^* \iota^T_{=0} L_{\mathsf{rotl}}, \\
L_2 &= (L_{\mathsf{rotl}} L_{=1})^+ \iota^P_{\mathsf{val=0}} (L_{=1} L_{\mathsf{rotr}})^+ \iota^T_{=0} L_{\mathsf{rotl}}, \\
L_3 &= (L_{\mathsf{rotl}} L_{=1})^+ L_{\mathsf{rotl}} \iota^T_{=0} L_{\mathsf{rotr}} L_{\mathsf{rotr}} (K_1 \mid K_2 K_3^* K_4), \\
K_1 &= \iota^T_{=0} L_{\mathsf{rotl}} (\iota^T_{\mathsf{mvr}} L_{\mathsf{rotl}})^* \iota^P_{\mathsf{val=0}}, \\
K_2 &= L_{=1} L_{\mathsf{rotl}} (\iota^T_{\mathsf{mvr}} L_{\mathsf{rotl}})^* \iota^P_{\mathsf{val=0}} L_{\mathsf{rotr}} (\iota^T_{=0} L_{\mathsf{rotr}})^* L_{=1} L_{\mathsf{rotr}}, \\
K_3 &= L_{=1} L_{\mathsf{rotl}} (\iota^T_{\mathsf{mvr}} L_{\mathsf{rotl}})^* L_{\mathsf{rotl}} L_{=1} L_{\mathsf{rotr}} L_{\mathsf{rotr}} (\iota^T_{=0} L_{\mathsf{rotr}})^* L_{=1} L_{\mathsf{rotr}}, \\
K_4 &= \iota^T_{=0} L_{\mathsf{rotl}} (\iota^T_{\mathsf{mvr}} L_{\mathsf{rotl}})^* L_{\mathsf{rotl}} L_{=1} L_{\mathsf{rotr}}.
\end{aligned}$$

The following lemma is the technical main ingredient for Theorem 23.

Lemma 22. *There exists a valid initial configuration R such that L is deterministic on R. Moreover, there exists a program $u \in L$ of length at least $\binom{n}{n/2}$ such that $|R \cdot u| = |R|$.*

Proof. Let us first show that L is deterministic on all configurations R which are valid and satisfy $|R \cap T| = n/2$. Since every word in L starts with a program from

$L^P_{\mathsf{reset}} L^Q_{\mathsf{reset}} L^Z_{\mathsf{reset}}$, we may also assume without loss of generality that each of the three counters is synchronized and has value zero under R.

Suppose there are two different words $u_1, u_2 \in L$ and a valid configuration R such that $|R \cdot u_1| = |R|$. We show that then, $|R \cdot u_2| < |R|$. Since L is prefix-free, there exist unique programs $p, q_1, q_2 \in I^*$ and different instructions $\iota_1, \iota_2 \in I$ such that $u_1 = p\iota_1 q_1$ and $u_2 = p\iota_2 q_2$. By Lemma 16 and Lemma 21, we already know that ι_1 and ι_2 do not correspond to a factor belonging to any of the languages L^P_{reset}, L^Q_{reset}, L^Z_{reset}, L_{rotl}, L_{rotr} or $L_{=1}$. The remaining cases are:

1. $\iota_1 = \iota^T_{=0}$ and $\iota_2 q_2 \in L_{=1} I^*$ (or vice versa),
2. $\iota_1 = \iota^T_{\mathsf{mvr}}$ and $\iota_2 q_2 \in L_{\mathsf{rotl}} L_{=1} I^*$ (or vice versa),
3. $\iota_1 = \iota^P_{\mathsf{val}=0}$ and $\iota_2 q_2 \in L_{\mathsf{rotl}} I^*$ (or vice versa),
4. $\iota_1 = \iota^P_{\mathsf{val}=0}$ and $\iota_2 q_2 \in \iota^T_{\mathsf{mvr}} L_{\mathsf{rotl}} I^*$ (or vice versa),
5. $\iota_1 = \iota^P_{\mathsf{val}=\mathsf{n}-1}$ and $\iota_2 q_2 \in \iota^T_{\mathsf{mvl}}(L_1 \mid L_2 \mid L_3) I^*$ (or vice versa).

In the first case, since $\left|R \cdot p\iota^T_{=0}\right| = |R \cdot p|$, we have $t_0 \notin R \cdot p$. As T is a sub-machine of U, we have $|R \cdot p \cap T| = |R \cap T| = n/2$. It is an invariant that P and Q are always synchronized and have value zero before and after applying a factor from $L_{=1}$. Therefore, we know by Lemma 21 that $|R \cdot u_2| < |R \cdot p|$. In the second case, observe that $\left|R \cdot p\iota^T_{\mathsf{mvr}}\right| = |R \cdot p|$ implies $t_{n-1} \notin R \cdot p$ and after applying the prefix r of $\iota_2 q_2$ corresponding to L_{rotl}, we have $t_0 \notin R \cdot pr$. This implies $|R \cdot u_2| < |R \cdot pr|$ as in the first case.

In the third case, by $\left|R \cdot p\iota^P_{\mathsf{val}=0}\right| = |R \cdot p|$, we know that the value of P under $R \cdot p$ is zero. Since $\iota_2 q_2 \in L_{\mathsf{rotl}} I^* = L^P_{\mathsf{dec}} \iota^T_{\mathsf{rotl}} I^*$, we conclude that $|R \cdot u_2| < |R \cdot p|$ by Lemma 19. The fourth case is analogous to the third case and the last case is covered later.

We now describe how to construct a program u of the given length such that $|R \cdot u| = |R|$. At any time, the value of the counter P describes the position of the tape head, i.e., the difference between the number of right and left rotations performed since the beginning of the computation. By construction, the tape head of each tape always stays in the same place and the tape content is rotated or modified. However, it is often convenient to think of the tape head moving on a stationary tape instead. This idea is captured in the following definition. We say that a configuration R *encodes* a word $b_{n-1} b_{n-2} \cdots b_0$ with $b_i \in \{0, 1\}$ if the value of P under R is v and for each $i \in \{0, 1, \ldots, n-1\}$, we have $b_{i+v \bmod n} = 1$ if and only if $t_i \in R$.

The initial configuration is the unique valid configuration encoding the word $0^{n/2} 1^{n/2}$. Then, the idea is that if some valid configuration R', which satisfies a series of invariants described below, encodes a word $X \in \{0, 1\}^*$, applying a program from $\iota^T_{\mathsf{mvl}}(L_1 \mid L_2 \mid L_3)$ to R' results in a configuration encoding the successor of X. This process can be repeated until we arrive at a configuration encoding $1^{n/2} 0^{n/2}$. Moreover, before and after applying a program from $\iota^T_{\mathsf{mvl}}(L_1 \mid L_2 \mid L_3)$, the tape head on T always points at the leftmost bit of the rightmost 1-block. Lemma 20 yields

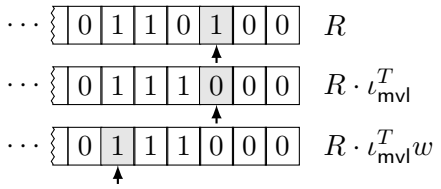

Figure 1: Action of a program $w \in L_1$

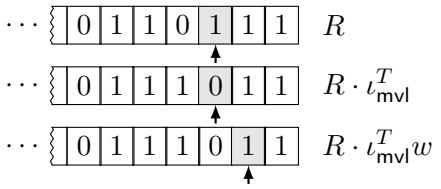

Figure 2: Action of a program $w \in L_2$

the desired lower bound for the length of the sequence of words that corresponds to the iterated process of going from one encoding to its succesor.

Let us now verify that for each configuration R' corresponding to an encoding X_{k-1} for some $k \geqslant 1$, there exists a program $\iota^T_{\mathsf{mvl}} w$ with $w \in L_1 \cup L_2 \cup L_3$ such that $\left|R' \cdot \iota^T_{\mathsf{mvl}} w\right| = |R'|$ and the configuration $R' \cdot \iota^T_{\mathsf{mvl}} w$ encodes X_k. By the invariant that the tape head of T points to the leftmost bit of the rightmost 1-block, the instruction ι^T_{mvl} moves the leftmost bit of the rightmost 1-block to the left, thereby replacing the encoding $p01^i0^j$ by $p101^{i-1}0^j$ while the tape head stays in the same position, now pointing at a 0. The program w now needs to move the remaining 1-block of length $i-1$ to the right (if applicable) and restore the invariant of the tape head pointing at the leftmost bit of the rightmost 1-block.

If $i = 1$, we apply a program from L_1. In that case, the new encoding already is $p10^{j+1}$ as desired and the instructions of w move the tape head to the left, skipping the first block of 1-bits, moving on to the first 0-bit left to the 1-block and then returning to the leftmost 1-bit of the rightmost 1-block. The action is illustrated in Figure 1 for an encoding with $i = 1$ and $j = 2$. In this case, the word preserving the cardinality of the configuration is

$$w \in L_{\mathsf{rotl}} \iota^T_{=0} L_{\mathsf{rotr}} L_{\mathsf{rotr}} L_{=1} L_{\mathsf{rotr}} L_{=1} L_{\mathsf{rotr}} L_{=1} L_{\mathsf{rotr}} \iota^T_{=0} L_{\mathsf{rotl}} \subseteq L_1.$$

Note that for each configuration, only the corresponding encoding, i.e., the restriction of the configuration to T relative to the tape head, is depicted.

If $i > 1$ and $j = 0$, we apply a program from L_2. In that case, the new encoding already is $p101^{i-1}$ as desired and w also only moves the tape head back to the right position using rotation instructions similar to those in the case $i = 1$. The action is illustrated in Figure 2 for an encoding with $i = 3$ and $j = 0$; the word is

$$w \in L_{\mathsf{rotl}} L_{=1} L_{\mathsf{rotl}} L_{=1} \iota^P_{\mathsf{val}=0} L_{=1} L_{\mathsf{rotr}} L_{=1} L_{\mathsf{rotr}} \iota^T_{=0} L_{\mathsf{rotl}} \subseteq L_2.$$

Again, only the encoding corresponding to each configuration is depicted.

The remaining case is $i > 1$ and $j \geqslant 1$ which means that the 1-block to the right of the tape head must be moved. If $j = 1$, this can be accomplished by applying a program from L_3 which ends with a program from K_1. If $j \geqslant 2$, one can choose a word that ends with a program from $K_2 K_3^{j-2} K_4$. In the latter case, the program

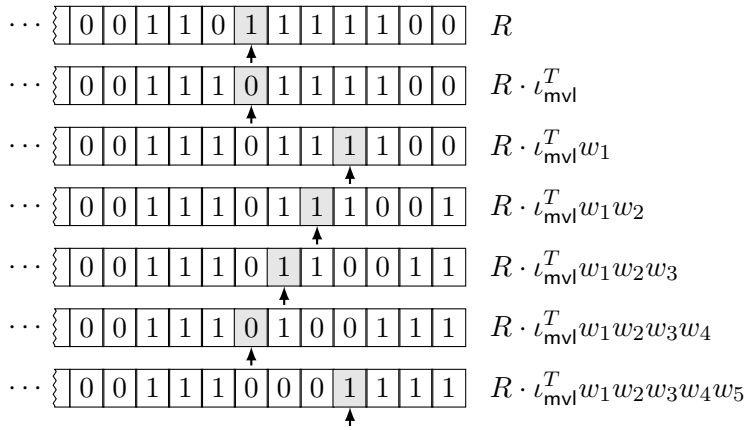

Figure 3: Action of a program $w \in L_3$

corresponding to K_2 moves the rightmost bit of the 1-block to position 0, then each program corresponding to K_3 moves one of the middle bits and the last bit is moved by a program in K_4. Each of the programs corresponding to K_2 or to K_3 verify that the bit moved to the right is not the last bit before starting the process. The program corresponding to K_4 checks that the bit moved to the right is the last bit by verifying that the left-hand cell contains a 0. Another difference between K_2/K_3 and K_4 is that K_2/K_3 move the tape head back to the left to fetch the next bit while K_4 leaves the pointer on the last moved bit which becomes the new leftmost bit of the rightmost 1-block. The operation of a program $w \in L_3$ is illustrated in Figure 3 for an encoding with $i = 5$ and $j = 2$. For better understanding, the word w is factorized as $w = w_1 w_2 w_3 w_4 w_5$ with

$$
\begin{aligned}
w_1 &\in (L_{\mathsf{rotl}} L_{=1})^4 L_{\mathsf{rotl}} \iota^T_{=0} L_{\mathsf{rotr}} L_{\mathsf{rotr}} \subseteq (L_{\mathsf{rotl}} L_{=1})^+ L_{\mathsf{rotl}} \iota^T_{=0} L_{\mathsf{rotr}} L_{\mathsf{rotr}},\\
w_2 &\in L_{=1} L_{\mathsf{rotl}} \iota^T_{\mathsf{mvr}} L_{\mathsf{rotl}} \iota^T_{\mathsf{mvr}} L_{\mathsf{rotl}} \iota^P_{\mathsf{val}=0} L_{\mathsf{rotr}} \iota^T_{=0} L_{\mathsf{rotr}} \iota^T_{=0} L_{\mathsf{rotr}} L_{=1} L_{\mathsf{rotr}} \subseteq K_2,\\
w_3, w_4 &\in L_{=1} L_{\mathsf{rotl}} \iota^T_{\mathsf{mvr}} L_{\mathsf{rotl}} \iota^T_{\mathsf{mvr}} L_{\mathsf{rotl}} L_{\mathsf{rotl}} L_{=1} L_{\mathsf{rotr}} L_{\mathsf{rotr}} \iota^T_{=0} L_{\mathsf{rotr}} \iota^T_{=0} L_{\mathsf{rotr}} L_{=1} L_{\mathsf{rotr}} \subseteq K_3,\\
w_5 &\in \iota^T_{=0} L_{\mathsf{rotl}} \iota^T_{\mathsf{mvr}} L_{\mathsf{rotl}} \iota^T_{\mathsf{mvr}} L_{\mathsf{rotl}} L_{\mathsf{rotl}} L_{=1} L_{\mathsf{rotr}} \subseteq K_4
\end{aligned}
$$

and the intermediate results after applying each of the factors are depicted.

The distinction between K_2 and K_3 is needed because we need to make sure that whenever moving a 1-block, the rightmost bit eventually reaches position 0 on the tape. This assertion ensures we do not return to a previous encoding from $1^{n/2}0^{n/2}$: after arriving at $1^{n/2}0^{n/2}$, the tape head of T is moved to position $n-1$. Assume we apply a word from $\iota^T_{\mathsf{mvl}}(L_1 \mid L_2 \mid L_3)^*$ to this configuration. Then, since T can be thought of as a ring buffer, the encoding is replaced with $01^{n/2-1}0^{n/2-1}1$ and the subsequent program from K_2 will overwrite the rightmost 1-bit, resulting in a reduction of the configuration size. More generally, this argument holds for any

configuration, which is reachable from R and under which the value of P is $n-1$, thereby concluding the analysis of the last case in the case distinction above to show that L is deterministic. □

The last missing piece is a component that imposes the language L on the labels of valid computations. To this end, let $\mathcal{A} = (Q, I, \delta, q_0, F)$ be the minimal deterministic automaton of L. We remove the sink state from Q and let all transitions leading to that state be undefined instead. Then, as long as there exists a state which has two ingoing transitions labeled by the same letter, we create a copy of the state and redirect one of the transitions to the copy. When interpreting the letters of I as actions on Q, the tuple (Q, I) then forms a token machine which we call *control unit*. By construction, all instructions are injective.

Putting the pieces together leads to the following theorem.

Theorem 23. *For each $n \in \mathbb{N}$, there exists an injective token machine with $n + 9\lceil \log n \rceil + \mathcal{O}(1)$ cells and 32 instructions which admits a maximal progressing computation of length at least $\binom{n}{\lfloor n/2 \rfloor}$.*

Proof. It suffices to prove the theorem for even numbers n. Let V be the union of U and the control unit. Any program, which is not a prefix of a word in L, empties the configuration when applied to the initial configuration $\{q_0\}$ in the control unit. Thus, by taking the union of the initial configuration from Lemma 22 and $\{q_0\}$, we obtain a maximal progressing computation of the desired length in V.

The only instructions required in the construction are ι^T_{rotl}, ι^T_{rotr}, ι^T_{mvl}, ι^T_{mvr}, $\iota^T_{=0}$, $\iota^P_{\mathsf{val=0}}$, $\iota^Q_{\mathsf{val=n-1}}$, $\iota^Z_{\mathsf{val=n/2}}$ and eight additional instructions for each of the three binary counters. Since L is a fixed language, the control unit has c cells for a constant $c \in \mathbb{N}$ (independent of n), and U has $n + 9\lceil \log n \rceil$ cells: n cells for the tape T and $\lceil \log n \rceil$ cells for each of the three tapes of the three binary counters. Therefore, the number of cells of V is $n + 9\lceil \log n \rceil + c$. □

Corollary 24. *There exists a sequence of transformation semigroups $(T_n)_{n\in\mathbb{N}}$ with a fixed number of generators such that T_n has n states and the $\mathcal{R}$-height (resp. $\mathcal{L}$-height, $\mathcal{J}$-height) of T_n is in $\Omega(2^n/n^{9.5})$.*

Proof. For the $\mathcal{R}$-height, the result is an immediate consequence of Theorem 23, Proposition 13 and Proposition 4. The statement also holds for $\mathcal{J}$-height because every $\mathcal{R}$-chain also is a $\mathcal{J}$-chain; see e.g. [10, Proposition 1.4]. An equivalent statement for the $\mathcal{L}$-height follows from Proposition 5 and the fact that all instructions used in the construction are injective. By Stirling's formula, we have $\binom{n}{n/2} \in \Omega(2^n/n^{0.5})$; see [11, 6]. Thus, we obtain the desired bound. Note that the bound in Theorem 23 is for $n + 9\lceil \log n \rceil + \mathcal{O}(1)$ cells and not just n cells. This yields the factor n^9 in the denominator. □

We can now prove our second main result.

Proof of Theorem 2. In view of Proposition 6 and Proposition 3, the theorem immediately follows from Corollary 24. □

5 Unary and Binary Alphabets

We now consider Green's relations in finite cyclic transformation semigroups and in finite 2-generated transformation semigroups. The following lemma is a useful tool when analyzing cyclic subsemigroups of transformation semigroups.

Lemma 25. *Let x be an element of a transformation semigroup on a set of states Q with $|Q| = n$ and let $i \geqslant 0$. Then $Q \cdot x^{i+1} \subseteq Q \cdot x^i$. In particular, $Q \cdot x^n = Q \cdot x^{n-1}$.*

Proof. For $i = 0$, the statement is trivial. Let $i \geqslant 1$ and $q \in Q \cdot x^{i+1}$. Then there exists some $p \in Q \cdot x^i$ such that $p \cdot x = q$. By induction, we have $p \in Q \cdot x^{i-1}$ and thus, $q = p \cdot x \in Q \cdot x^i$, as desired.

For the second part, note that if $Q \cdot x^{i+1} = Q \cdot x^i$ for any $i \in \{0, \ldots, n-1\}$, we also obtain $Q \cdot x^{i+k+1} = x^{i+k}$ for all $k \geqslant 0$ and, in particular, $Q \cdot x^n = Q \cdot x^{n-1}$. Suppose for the sake of contradiction, that no such i exists. Then, by the previous observation, $Q \supsetneq Q \cdot x^1 \supsetneq Q \cdot x^2 \supsetneq \cdots \supsetneq Q \cdot x^n$, contradicting the fact that $Q \cdot x^n$ contains at least one element and that $|Q| = n$. □

Proposition 26. *The number of $\mathcal{J}$-classes of a cyclic transformation semigroup on n states is at most $n - 1$.*

Proof. Let T be a cyclic transformation semigroup on a set Q of n states. Let $x \in T$ such that $\{x^i \mid i \geqslant 1\} = T$. By Lemma 25, we have $Q \cdot x^n = Q \cdot x^{n-1}$ which implies that for $\omega = n!$ the mapping x^ω is the identity on $Q \cdot x^{n-1}$, i.e., $x^{n-1+\omega} = x^{n-1}$. Therefore, $x^i \mathrel{\mathcal{J}} x^{n-1}$ whenever $i \geqslant n - 1$. □

Proposition 27. *There exists a cyclic partial transformation semigroup on n states with $\mathcal{R}$-height n.*

Proof. Consider the partial transformation semigroup on $Q = \{1, \ldots, n\}$ generated by the single partial transformation $x \colon Q \to Q$ that maps each element $i \in \{1, \ldots, n-1\}$ to $i + 1$ and is undefined on n. We have $\left|Q \cdot x^i\right| = n - i$ for all $i \in \{1, \ldots, n\}$ and thus, $(x^1, x^2, \ldots, x^n)$ is an $\mathcal{R}$-chain. □

Together with Proposition 4 and the observation that in cyclic semigroups, the relations $\leqslant_\mathcal{R}$, $\leqslant_\mathcal{L}$ and $\leqslant_\mathcal{J}$ coincide, we obtain the following result.

Corollary 28. *In cyclic transformation semigroups, the bound $n - 1$ is tight for the number of $\mathcal{R}$-classes (resp. $\mathcal{L}$-classes, $\mathcal{J}$-classes) and for the $\mathcal{R}$-height (resp. $\mathcal{L}$-height, $\mathcal{J}$-height). In cyclic partial transformation semigroups, the bound n is tight for the number of $\mathcal{R}$-classes (resp. $\mathcal{L}$-classes, $\mathcal{J}$-classes) and for the $\mathcal{R}$-height (resp. $\mathcal{L}$-height, $\mathcal{J}$-height).*

This corollary contrasts the exponential lower bound proved in Section 4.4 where 32 generators where required. We now investigate the case of two generators which closes the gap between unary alphabets and alphabets with at least 32 elements.

Proposition 29. *There exists a 2-generated partial transformation semigroup on $n+3$ elements with $n!$ different $\mathcal{R}$-classes (resp. $\mathcal{L}$-classes, $\mathcal{J}$-classes).*

Proof. Let $Q = \{0, 1, \ldots, n-1\} \cup \{p, q, r\}$. Let $x\colon Q \to Q$ be the partial transformation such that $i \cdot x = 1 - i$ for $i \in \{0, 1\}$, $i \cdot x = i$ for $i \in \{2, 3, \ldots, n-1\}$, $p \cdot x = q$, $q \cdot x = r$ and $r \cdot x$ undefined. Let $y\colon Q \to Q$ be the transformation with $i \cdot y = (i+1) \bmod n$ for all $i \in \{0, 1, \ldots, n-1\}$ and with $p \cdot y = q \cdot y = p$. Again, $r \cdot y$ is undefined.

We consider the partial transformation semigroup P generated by x and y. For each bijection $f\colon \{0, 1, \ldots, n-1\} \to \{0, 1, \ldots, n-1\}$, there exists an element $w \in P$ such that the restriction of w to $\{0, 1, \ldots, n-1\}$ is f. Since one can always express such an element as a product over the generators x and y and eliminate all factors x^2, we can assume without loss of generality that $p \cdot w \in \{p, q\}$. Multiplying each such element by $y^n x^2$ yields $n!$ elements from pairwise different $\mathcal{R}$-classes. □

Since $n! \geqslant n^{\lfloor n/2 \rfloor} \in n^{\Theta(n)}$ for all $n \geqslant 1$, our result immediately implies the following lower bound.

Corollary 30. *There exists a sequence of 2-generated transformation semigroups $(T_n)_{n \in \mathbb{N}}$ such that T_n has n states and the number of $\mathcal{R}$-classes (resp. $\mathcal{L}$-classes, $\mathcal{J}$-classes) of T_n is in $n^{\Omega(n)}$.*

We now present a construction which, when combined with the main result from Section 4.4, allows for obtaining lower bounds for the length of chains in the case of binary alphabets.

Proposition 31. *Let T be an m-generated partial transformation semigroup on n states. Then there exists a 2-generated partial transformation semigroup U on mn states such that the $\mathcal{R}$-height of U is at least the $\mathcal{R}$-height of T. Moreover, if T is injective, then U is also injective.*

Proof. Suppose that T is generated by $\Sigma = \{a_0, a_1, \ldots, a_{m-1}\}$ and let x, y be the transformations on $Q \times \Sigma$ defined by

$$(q, a_i) \cdot x = (q \cdot a_i, a_i) \text{ and}$$
$$(q, a_i) \cdot y = (q, a_{(i+1) \bmod m})$$

for all $q \in Q$ and for all $i \in \{0, \ldots, m-1\}$. To see that the $\mathcal{R}$-height of the transformation semigroup U generated by $\{x, y\}$ is at least the $\mathcal{R}$-height of T, note that any $\mathcal{R}$-chain of T can be converted to an $\mathcal{R}$-chain of U by mapping a_i to $y^i x y^{m-i}$.

By construction, the transformation y is injective, and x is injective if and only if every transformation from Σ is injective. □

Corollary 32. *There exists a sequence of 2-generated transformation semigroups $(U_n)_{n\in\mathbb{N}}$ such that U_n has n states and the $\mathcal{R}$-height (resp. $\mathcal{L}$-height, $\mathcal{J}$-height) of U_n is in $2^{\Omega(n)}$.*

Proof. This follows from a combination of Theorem 23, Proposition 13, Proposition 31 and Proposition 4. The statement also holds for the $\mathcal{J}$-height because every $\mathcal{R}$-chain also is a $\mathcal{J}$-chain; see e.g. [10, Proposition 1.4]. For $\mathcal{L}$-height, the statement follows from Proposition 5 and the fact that all instructions are injective. □

6 Summary and Outlook

We investigated the size of certain parameters related to Green's relations in finite transformation semigroups. In particular, we gave upper and lower bounds for the number of equivalence classes and the length of chains. The special cases of fixed-size alphabets, unary alphabets and binary alphabets were covered as well.

Some problems remain open. For example, we were able to obtain lower and upper bounds for the $\mathcal{R}$-height of a transformation semigroup on a given number of elements which are tight up to a polynomial gap. However, for the $\mathcal{L}$-height, we are not aware of any upper bound which is substantially better than the naïve bound n^n. Proving lower bounds for the $\mathcal{L}$-height seems to require entirely new techniques: token computations cannot be used to prove lower bounds beyond 2^n and in the setting of $\mathcal{L}$-chains, a decrease in the number of tokens does not imply a $>_{\mathcal{L}}$-descend. It would also be interesting to see whether the lower and upper bounds for the $\mathcal{R}$-height can be further tightened, although we expect any substantial improvements to be highly technical.

The construction presented in the proof of Proposition 31 increases the number of states by a constant factor. As a result, an exponential gap between the lower and upper bound remains in the case of binary alphabets.

Acknowledgments. We thank the anonymous referees for valuable suggestions which helped to improve the presentation of the paper. We also thank Mikhail Volkov for inspiring comments which led to the addition of Section 5.

References

[1] H. W. Becker and J. Riordan. The arithmetic of Bell and Stirling numbers. *American Journal of Mathematics*, 70(2):385–394, 1948.

[2] D. Berend and T. Tassa. Improved bounds on Bell numbers and on moments of sums of random variables. *Probability and Mathematical Statistics*, 30(2):185–205, 2010.

[3] Ch. Brandl and H. U. Simon. Complexity analysis: Transformation monoids of finite automata. In I. Potapov, editor, *DLT 2015, Proceedings*, volume 9168 of *Lecture Notes in Computer Science*, pages 143–154. Springer-Verlag, 2015.

[4] O. Carton and M. Michel. Unambiguous Büchi automata. *Theoretical Computer Science*, 297(1):37–81, 2003.

[5] S. Eilenberg. *Automata, Languages, and Machines*, volume B. Academic Press, 1976.

[6] W. Feller. *An Introduction to Probability Theory and Its Applications: Volume 1.* John Wiley & Sons, New York, 1957.

[7] L. Fleischer and M. Kufleitner. Green's relations in finite transformation semigroups. In *CSR 2017, Proceedings*, volume 10304 of *LNCS*, pages 112–125. Springer, 2017.

[8] M. Ganardi, D. Hucke, and M. Lohrey. Querying regular languages over sliding windows. In *FSTTCS 2016, Proceedings*, volume 65 of *LIPIcs*, pages 18:1–18:14. Dagstuhl Publishing, 2016.

[9] M. Holzer and B. König. On deterministic finite automata and syntactic monoid size. *Theoretical Computer Science*, 327(3):319–347, Nov. 2004.

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

[11] H. Robbins. A remark on Stirling's formula. *The American Mathematical Monthly*, 62:26–28, 1955.