Regularity Preserving but not Reflecting Encodings

Jörg Endrullis
Department of Computer Science
VU University Amsterdam
Amsterdam, The Netherlands
Email: j.endrullis@vu.nl

Clemens Grabmayer
Department of Computer Science
VU University Amsterdam
Amsterdam, The Netherlands
Email: c.a.grabmayer@vu.nl

Dimitri Hendriks
Department of Computer Science
VU University Amsterdam
Amsterdam, The Netherlands
Email: r.d.a.hendriks@vu.nl

***Abstract*—Encodings, that is, injective functions from words to words, have been studied extensively in several settings. In computability theory the notion of encoding is crucial for defining computability on arbitrary domains, as well as for comparing the power of models of computation. In language theory much attention has been devoted to regularity preserving functions.**

A natural question arising in these contexts is: Is there a bijective encoding such that its image function preserves regularity of languages, but its preimage function does not? Our main result answers this question in the affirmative: For every countable class $\mathcal{L}$ of languages there exists a bijective encoding f such that for every language $L \in \mathcal{L}$ its image $f[L]$ is regular.

Our construction of such encodings has several noteworthy consequences. Firstly, anomalies arise when models of computation are compared with respect to a known concept of implementation that is based on encodings which are not required to be computable: Every countable decision model can be implemented, in this sense, by finite-state automata, even via bijective encodings. Hence deterministic finite-state automata would be equally powerful as Turing-machine deciders.

A second consequence concerns the recognizability of sets of natural numbers via number representations and finite automata. A set of numbers is said to be recognizable with respect to a representation if an automaton accepts the language of representations. Our result entails that there exists a number representation with respect to which every recursive set is recognizable.

I. Introduction

In order to define computability of number-theoretic functions through computational models that operate on strings of symbols from an alphabet Σ (rather than defining computability directly via recursion schemes) one usually employs *(number) representations*, that is, injective functions $r : \mathbb{N} \to \Sigma^*$. A function $f : \mathbb{N} \to \mathbb{N}$ is called *r-computable* (*computable by a Turing machine using representation r*) if there exists a Turing-computable function $\varphi : \Sigma^* \to \Sigma^*$ such that $\varphi \circ r = r \circ f$. For representations r that are informally computable (i.e., there is a machine-implementable algorithm that always terminates, and computes r), it can be argued on the basis of Church's thesis (similar as e.g. in [21, p. 28]) that r-computability does not depend on the specific choice of r, and coincides with partial recursiveness.

Shapiro [26] studied the influence that (unrestricted) bijective representations r have on the notion of r-computability. He found that the only functions that are r-computable with respect to all bijective representations r are the almost constant and almost identity functions; and that there are functions that are not r-computable for any representation r. Furthermore, he defines 'acceptable' number representations: a bijective representation r is called 'acceptable' if the successor function lifted to the r-coded natural numbers is Turing computable. He goes on to show that, a representation r is acceptable, if and only if r-computability coincides with partial recursiveness.

In this paper we focus on the notion of computability by *finite automata* of *sets* of natural numbers. In particular, we investigate how number representations determine the sets of natural numbers that are computable by finite-state automata. Such sets are called 'recognizable': a set $S \subseteq \mathbb{N}$ is called *r-recognizable* (*recognizable with respect to representation r*), if there is a finite automaton that for all $n \in \mathbb{N}$ decides membership of n in S when $r(n)$ is given to it as input.

We are interested in comparing representations r with respect to their computational power as embodied by the r-recognizable sets. This idea gives rise to a hierarchy via a subsumption preorder between representations: $r_1 : \mathbb{N} \to A^*$ subsumes $r_2 : \mathbb{N} \to B^*$ if all r_2-recognizable sets are also r_1-recognizable. There are several natural questions concerning this preorder; to name a few:

(i) When does a number representation subsume another?
(ii) Is the hierarchy proper: do there exist representations r_1 and r_2 such that r_1 subsumes r_2, but not vice versa?
(iii) Is there a representation that subsumes all others?
(iv) Is every (injective) number representation subsumed by a bijective number representation?
(v) What classes $\mathcal{C} \subseteq \wp(\mathbb{N})$ of sets of natural numbers are recognizable with respect to a number representation?

As our computational devices are finite automata, all of these questions boil down to problems in language theory. In particular the comparison of number representations is intimately connected with *encodings*, injective mappings from words to words, that have the property that their image function preserves regularity of languages. For bijective number representations $f : \mathbb{N} \to A^*$ and $g : \mathbb{N} \to B^*$, we have that f subsumes g if and only if the set function

$$(f \circ g^{-1})[\_]$$

preserves regularity of languages; here we use the notation $h[\_]$ to denote the image function of a function h. Regularity preserving functions play an important role in different areas of computer science, and have been studied extensively. An important result in this area is the work [17], [15] of Pin and

Silva, providing a characterization of regularity preservation of preimage functions in terms of uniformly continuous maps on the *profinite topology* [15, Cor. 6.2].

A natural question that presents itself then is the following:

> *Are there bijective functions $f : \Sigma^* \to \Sigma^*$ whose image function $f[\_]$ preserves regularity of languages, but whose preimage function $f^{-1}[\_]$ does not?*

For *bijective* word functions we experienced this to be a very challenging question, which to the best of our knowledge, has remained unanswered in the literature. Using the results of [17], [15], it can equivalently be formulated as follows:

> *Are there bijective functions $f : \Sigma^* \to \Sigma^*$ such that f is uniformly continuous, but f^{-1} is not uniformly continuous in the profinite topology?*

Concerning recognizable sets and the hierarchy of number representations, the question translates to:

> *Are there bijective number representations f and g such that f strictly subsumes g?*

If this were not the case, subsumption would imply equivalence for bijective number representations, and the hierarchy would collapse.

Our main result (Theorem 8), which allows us to answer all of the above questions, is the following:

Main Theorem. *For every countable class $\mathcal{L} \subseteq \wp(\Sigma^*)$ of languages over a finite alphabet Σ, and for every alphabet Γ with $|\Gamma| \geq 2$, there exists a bijective encoding $f : \Sigma^* \to \Gamma^*$ such that for every language $L \in \mathcal{L}$ its image $f[L]$ is regular.*

With respect to computability theory and recognizable sets of natural numbers, this result can be restated as follows:

> *For every countable decision model $\mathcal{M} \subseteq \wp(\mathbb{N})$, there exists a bijective representation $f : \mathbb{N} \to \Sigma^*$ such that every set $M \in \mathcal{M}$ is f-recognizable.*

As a direct consequence, when allowing for arbitrary bijective number representations, we find the unsought:

> *Finite automata are as strong as Turing-machine deciders.* (↯)

That is, there is a bijective representation such that finite automata can recognize any computable set of natural numbers.

Our result also has consequences in the context of the work by Boker and Dershowitz on comparing the power of computational models, as described below. Models over different domains are typically compared with the help of encodings that translate between different number representations. In order to prevent encodings from changing the nature of the problem, they are usually required to be 'informally algorithmic', 'informally computable', or 'effective' (see e.g. [21, p. 27]). However, the latter concepts are rather vague, and in any case non-mathematical. Therefore they are unsatisfactory from the viewpoint of a rigorous conceptual analysis.

In the formal approach for comparing models of computation proposed by Boker and Dershowitz in [3], [5], [4], encodings are merely required to be injective. On the basis of this stipulation, a computational model $\mathcal{M}_2$ is defined to be 'at least as powerful as' $\mathcal{M}_1$, denoted by $\mathcal{M}_1 \lesssim \mathcal{M}_2$, if there exists an encoding $\rho : \Sigma_1^* \to \Sigma_2^*$ with the property that for every function f computed by $\mathcal{M}_1$ there is a function g computed by $\mathcal{M}_2$ such that the following diagram commutes:

$$\begin{array}{ccc} \Sigma_1^* & \xrightarrow{\ \rho\ } & \Sigma_2^* \\ {\scriptstyle f \in \mathcal{M}_1}\downarrow & & \downarrow{\scriptstyle g \in \mathcal{M}_2} \\ \Sigma_1^* & \xrightarrow[\ \rho\ ]{} & \Sigma_2^* \end{array} \qquad \mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$$

(In order to highlight the encoding used, $\mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$ is written.) Although encodings are not required to be (informally) computable, this approach works quite well in practice.

However, in analogy to what we found for recognizability, one runs into the anomaly (↯) again, namely when comparing the power of decision models with the preorder $\lesssim$. Our main result implies $\mathcal{C} \lesssim \mathsf{DFA}$ for every countable class of decision problems $\mathcal{C}$, where DFA denotes the class of deterministic finite-state automata. Even stronger, it follows that there is a bijective encoding ρ such that $\mathcal{C} \lesssim_\rho \mathsf{DFA}$. As a consequence we obtain that $\mathsf{TMD} \lesssim_\rho \mathsf{DFA}$ holds for the class TMD of Turing-machine deciders, and a bijective encoding ρ.

Further Related Work

For a general introduction to automata and regular languages we refer to [22], [9]. We briefly mention related work with respect to regularity preserving functions apart from work [17], [15] of Pin and Silva that we have already discussed above. The works [27], [10], [14], [23], [11], [24], [13] investigate regularity preserving relations; in particular, [24] provides a characterization of prefix-removals that preserve regularity. In [16], Pin and Sakarovitch study operations and transductions that preserve regularity. In [12], Kozen gives a characterization of word functions over a one-letter alphabet whose preimage function preserves regularity of languages. The paper [2] by Berstel, Boasson, Carton, Petazzoni and Pin characterizes language preserving 'filters'; a filter is a set $F \subseteq \mathbb{N}$ used to delete letters from words of the language as indexed by elements of the filter.

II. Preliminaries

We use standard terminology and notation, see, e.g., [1]. Let Σ be an alphabet, i.e., a finite non-empty set of symbols. We denote by Σ^* the set of all finite words over Σ, and by ε the empty word. The set of infinite sequences over Σ is $\Sigma^{\mathbb{N}} = \{\sigma \mid \sigma : \mathbb{N} \to \Sigma\}$ with $\mathbb{N} = \{0, 1, 2, \ldots\}$, the set of natural numbers.

A deterministic finite-state automaton (DFA) is a tuple $A = \langle Q, \Sigma, \delta, q_0, F\rangle$ consisting of a finite set of states Q, an input alphabet Σ, a transition function $\delta : Q \times \Sigma \to Q$, an initial state $q_0 \in Q$, and a set $F \subseteq Q$ of accepting states. The transition function δ is extended to $\delta^* : Q \times \Sigma^* \to Q$ by

$$\delta^*(q, \varepsilon) = q \qquad \delta^*(q, aw) = \delta^*(\delta(q, a), w)\,,$$

for all states $q \in Q$, letters $a \in \Sigma$ and words $w \in \Sigma^*$. We will write just δ for δ^*. A word $w \in \Sigma^*$ is accepted by A if $\delta(q_0, w)$, the state reached after reading w, is an accepting state. We write $\mathrm{Lang}(A)$ for the language accepted by the automaton A, i.e., $\mathrm{Lang}(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$.

A DFA with output (DFAO) is a tuple $\langle Q, \Sigma, \delta, q_0, \Delta, \lambda\rangle$, with the first four components as in the definition of a DFA, but with, instead of a set of accepting states, an output alphabet Δ, and an output function $\lambda : Q \to \Delta$. A DFAO $A = \langle Q, \Sigma, \delta, q_0, \Delta, \lambda\rangle$ realizes a function mapping words over Σ to letters in Δ; we denote this function also by A, that is, we define $A : \Sigma^* \to \Delta$ by

$$A(w) = \lambda(\delta(q_0, w)) \qquad \text{for all } w \in \Sigma^*.$$

A DFA $\langle Q, \Sigma, \delta, q_0, F\rangle$ can thus be viewed as a DFAO $\langle Q, \Sigma, \delta, q_0, \{0, 1\}, \chi_F\rangle$ where χ_F is the characteristic function of F; instead of a state being accepting or not, it has output 1 or 0 respectively.

Two automata A and A' over the same input alphabet Σ are equivalent on a set $X \subseteq \Sigma^*$ if $A(x) = A'(x)$ for all $x \in X$. We use this functional notation also for a DFA A, stipulating: $A(x) = 1$ ($A(x) = 0$) iff A accepts x (A does not accept x).

A partition P of a set U is a family of sets $P \subseteq \wp(U)$ such that $\varnothing \notin P$, $\bigcup_{A\in P} A = U$, and for all $A, B \in P$ with $A \neq B$, $A \cap B = \varnothing$.

The pigeon hole principle (PHP) states that if n pigeons are put into m pigeonholes with $n > m$, then at least one pigeonhole contains more than one pigeon. PHP for infinite sets is that if infinitely many pigeons are put into finitely many holes, then one hole must contain infinitely many pigeons.

Let A, B, X, and Y be sets, with $X \subseteq A$ and $Y \subseteq B$. For a function $A \to B$ we write $f[X]$ for the image of X under f, that is, $f[X] = \{f(x) \mid x \in X\}$. Likewise, we write $f^{-1}[Y]$ for the preimage of Y under f, i.e., $f^{-1}[Y] = \{x \mid f(x) \in Y\}$.

A function $F : \wp(\Sigma^*) \to \wp(\Gamma^*)$ *preserves regularity* if $F(L)$ is regular whenever L is a regular language.

III. Main Results

In this section, we prove our main results. We work towards Theorem 8 stating that for every countable class $\mathcal{L}$ of languages there exists a bijective encoding $f : \Sigma^* \to \Gamma^*$ such that $f[L]$ is regular for every language $L \in \mathcal{L}$. We first prove the existence of injective encodings with (a weakening of) this property (Lemma 2), and then strengthen this result to bijective functions (Lemma 6). From Theorem 8 we then obtain the existence of bijective functions that are regularity preserving but not regularity reflecting, Theorem 9.

For injective encodings we do not require that the images $f[L]$ are regular. For our construction, we use the weaker property that $f[L]$ is recognizable among $f[\Sigma^*]$. This means that there exists an automaton that for every $w \in \Sigma^*$, on the input of the code $f(w)$ decides whether $w \in L$. This leads to the notion of 'relatively regular in' which also is crucial for recovering the 'fiber lemma' for c-automatic sequences, see Lemma 26.

Definition 1. Let $L, M \subseteq \Sigma^*$ be formal languages over the alphabet Σ, with $L \subseteq M$. Then L *is relatively regular in* M if there exists a regular language R such that $L = R \cap M$.

So a regular language $S \subseteq \Sigma^*$ is relatively regular in Σ^*.

Lemma 2. *Let Σ and Γ be finite alphabets, with $|\Gamma| \geq 2$. Let $\mathcal{L} \subseteq \wp(\Sigma^*)$ be a countable set of formal languages. There exists an injective function $f : \Sigma^* \to \Gamma^*$ such that for every $L \in \mathcal{L}$, $f[L]$ is relatively regular in $f[\Sigma^*]$.*

Proof. Let $L_1, L_2, L_3, \ldots$ be an enumeration of $\mathcal{L}$, and let $v_0, v_1, v_2, \ldots$ be an enumeration of Σ^*. For $i \geq 1$, we write $\chi_i : \Sigma^* \to \{0, 1\}$ for the characteristic function of L_i, that is,

$$\chi_i(v) = \begin{cases} 1 & \text{if } v \in L_i \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } v \in \Sigma^*.$$

Without loss of generality we assume $\Gamma = \{0, 1, \ldots, k-1\}$ for some $k \geq 2$. Define the function $f : \Sigma^* \to \Gamma^*$ by

$$f(v_n) = \chi_1(v_n)\,\chi_2(v_n) \cdots \chi_n(v_n)\,, \qquad \text{for all } n \in \mathbb{N}.$$

For every $i = 1, 2, \ldots$, we construct a DFAO A_i and show that $f[L_i] = \mathrm{Lang}(A_i) \cap f[\Sigma^*]$ witnessing that $f[L_i]$ is relatively regular in $f[\Sigma^*]$, as required.

Fix an arbitrary integer $i \geq 1$. Define $A_i = \langle Q, \Gamma, \delta, 0, \lambda\rangle$ where $Q = \{0, 1, \ldots, i-1\} \cup \{i_0, i_1\}$, $\Gamma = \{0, 1, \ldots, k-1\}$, $\delta : Q \times \Gamma \to Q$ is defined by

$$\begin{aligned} \delta(q, a) &= q + 1 \quad \text{for all } q \in \{0, 1, \ldots, i-2\} \text{ and } a \in \Gamma, \\ \delta(i-1, 0) &= i_0 \\ \delta(i-1, a) &= i_1 \qquad \text{for all } a \in \Gamma \setminus \{0\}, \\ \delta(i_j, a) &= i_j \qquad \text{for all } j \in \{0, 1\} \text{ and } a \in \Gamma, \end{aligned}$$

and $\lambda : Q \to \Gamma$ is defined, for all $q \in \{0, 1, \ldots, i-1\}$, by

$$\lambda(q) = \chi_i(v_q) \qquad \lambda(i_0) = 0 \qquad \lambda(i_1) = 1\,.$$

We show that $f[L_i] = \mathrm{Lang}(A_i) \cap f[\Sigma^*]$; equivalently, for all $v \in \Sigma^*$,

$$v \in L_i \iff f(v) \in \mathrm{Lang}(A_i)\,.$$

Let $v \in \Sigma^*$, so $v = v_n$ for some $n \in \mathbb{N}$. Note that $|f(v_n)| = n$. Hence, if $n \leq i$, the automaton A_i is in state n after having read the word $f(v_n)$ and outputs $\chi_i(v_n)$. If $n \geq i$, then after having read $f(v_n)$, the automaton is in state i_j, where j is the i-th letter of $f(v_n)$, that is, $j = \chi_i(v_n)$. In both cases we get $v_n \in L_i$ if and only if $f(v_n) \in \mathrm{Lang}(A_i)$. □

For lifting the result of Lemma 2 from injective to bijective encodings, we need some preliminary notions and results.

Definition 3. Let U be a set and $I, C \subseteq U$. The set C *is attracted to* I if $C \subseteq I$ whenever $C \cap I$ is finite. For a partition E of U, we say that E *is attracted to* I, when, for every $C \in E$, C is attracted to I.

Equivalently, E is attracted to I if every finite C is included in I and every infinite C has an infinite intersection with I.

Lemma 4. *Let $E = \{C_0, C_1 \ldots, C_{n-1}\}$ be a finite partition of Σ^* with $C_i \subseteq \Sigma^*$ a regular set for every $i \in \{0, 1, \ldots, n-1\}$. Let $I \subseteq \Sigma^*$ and assume that E is attracted to I. For every DFA A there exists a DFA A' such that A' is equivalent to A on I and the refined partition*

$$E' = \{C'_{i,j} \mid i \in \{0, 1, \ldots, n-1\},\ j \in \{0, 1\}\}$$

is attracted to I, where, for $i \in \{0, 1, \ldots, n-1\}$, $j \in \{0, 1\}$,

$$C'_{i,j} = C_i \cap \{u \in \Sigma^* \mid A'(u) = j\}\,.$$

Proof. We start with $A' = A$, and repeatedly adapt A' (and therewith E') until E' is attracted to I, in such a way that equivalence with A is upheld.

Assume that E' is not attracted to I. Then there exist $i \in \{0, 1, \ldots, n-1\}$ and $j \in \{0, 1\}$ such that $C'_{i,j} \setminus I \neq \varnothing$ but $C'_{i,j} \cap I$ is finite. Without loss of generality, assume that $j = 0$. Since $C_i = C'_{i,j} \cup C'_{i,1-j}$ it follows that $C_i \setminus I \neq \varnothing$ and hence $C_i \cap I$ is infinite by assumption. By the pigeonhole principle (for infinite sets) it follows that $C'_{i,1} \cap I$ is infinite. Since C_i is a regular set and A' is a finite automaton, it follows that $C'_{i,0}$ is regular as it is the intersection of two regular sets. As $C'_{i,0} \cap I$ is finite, also $C'_{i,0} \setminus I$ is regular. As a consequence we can change the finite automaton A' to accept the words in $C'_{i,0} \setminus I$ (and otherwise to behave as before). This adaptation preserves equivalence, and we now have that $C'_{i,j}$ is attracted to I for $j = 0, 1$, since, after the adaptation, $C'_{i,0} \cap I = C'_{i,0}$ and $C'_{i,1} \cap I$ is infinite. We repeat the procedure until $C'_{i,j}$ is attracted to I for every $i \in \{0, 1, \ldots, n-1\}$ and $j \in \{0, 1\}$. □

The following lemma, Lemma 6, is a key contribution of our paper. It states that every injection $f : A \to \Gamma^*$ (with A some countably infinite set) can be transformed into a bijection $g : A \to \Gamma^*$ such that, for all $L \subseteq A$, $g[L]$ is a regular language whenever $f[L]$ is relatively regular in the image $f[A]$. Before proving the lemma, we sketch the construction. We construct g as the limit of a sequence of adaptations of f. Roughly speaking, we make f 'more bijective' in every step. We let

- $v_0, v_1, v_2, \ldots$ be an enumeration of A, and
- $w_0, w_1, w_2, \ldots$ be an enumeration of Γ^*.

Figure 1 sketches an injective function f; an arrow from v_i to w_j indicates that $f(v_i) = w_j$.

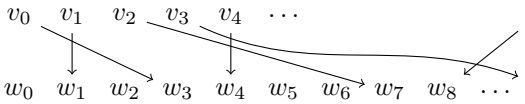

Fig. 1. Starting situation: the function f.

The idea is that we change the target of arrows such that all words w_i for $i \in \mathbb{N}$ become part of the image. For every natural number $n = 0, 1, 2, \ldots$ we will pick (while avoiding repetitions) a word v_{k_n} (for $k_n \in \mathbb{N}$) from the input domain and then adapt f by stipulating $v_{k_n} \mapsto w_n$. Then, in the limit, every word $w \in \Gamma^*$ will be in the image. (In order for the limit of the adaptation to be a function, we also need to guarantee that every v_{k_n} will be picked precisely once.)

The crucial point of the construction is the following: when changing arrows, we need to ensure that

($\star$) the limit of the process preserves relative regularity.

Note that for bijective functions $g : A \to \Gamma^*$ we have that $g[L]$ is relatively regular in the image $g[A] = \Gamma^*$ if and only if $g[L]$ is regular. Thus if we can ensure ($\star$), the resulting bijective function g will have the desired property.

How to pick the v_{k_n} for ensuring ($\star$)? Let $A_0, A_1, A_2, \ldots$ be an enumeration of all finite automata over the alphabet Γ. We write $u \sim_n v$ if for every $i < n$, the automaton A_i accepts the word u if and only if it accepts the word v. We then pick for every natural number $n \in \mathbb{N}$, a word v_{k_n} such that $f(v_{k_n}) \sim_n w_n$ (and the word v_{k_n} has not been picked before). In other words, we pick v_{k_n} such that the first n automata $A_0, A_1, \ldots, A_{n-1}$ cannot distinguish $f(v_{k_n})$ from the image w_n after the adaptation $v_{k_n} \mapsto w_n$. This choice guarantees that every automaton A_i ($i \in \mathbb{N}$) is only affected by a finite number of adaptations, namely the first i transformation steps. For every further adaptation ($j > i$), the behavior of the automaton A_i is taken into account for the choice of v_{k_j}, and as a consequence the modification $v_{k_j} \mapsto w_j$ preserves the acceptance behavior of A_i. Then for the limit g of the adaptation process we have for almost all $n \in \mathbb{N}$ that A_i accepts $f(v_n)$ if and only if A_i accepts $g(v_n)$. In order to guarantee that every v_i will be picked eventually and that the adaptation preserves injectivity, we pick among the suitable candidates for v_{k_n} the one which appears first in the enumeration $w_0, w_1, \ldots$.

Remark 5. There is a caveat here that we will ignore in this sketch of the construction. We actually need to make sure that a word v_{k_n} with these properties exists. To ensure this, the equivalence classes with respect to $\sim_n$ must be attracted to the image of f. This is in general not the case, but we can employ Lemma 4 to adapt the automata outside of the image of f. We refer to the proof of Lemma 6 for the details.

We explain this process at the example of the function f given in Figure 1. For the first step $n = 0$, we want to adapt f such that w_0 becomes part of the image of f. Note that the relation $\sim_0$ relates all words of Γ^*. As a consequence we can pick any word v_{k_0}. We choose $v_{k_0} = v_1$ since the image $f(v_1)$ appears first in the sequence $w_0, w_1, \ldots$, and we adapt the function f by $v_1 \mapsto w_0$.

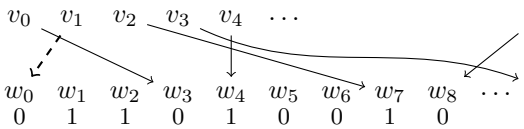

Fig. 2. Result of the first adaptation of the function f.

The result of this first adaptation is shown in Figure 2. For the second step ($n = 1$) we want that w_1 becomes part of the image. Now $\sim_1$ relates words that have equal behavior with respect to acceptance by the automaton A_0. The numbers 0 and 1 below the words w_i in Figure 2 indicate whether A_0

accepts the word w_i (1) or not (0). The word w_1 is accepted by A_0 and likewise are $f(v_2)$ and $f(v_4)$. Among these candidates, we choose $v_{k_1} = v_4$ since $f(v_4)$ appears first in the sequence $w_1, w_2, \ldots$. We modify the function f by setting $v_4 \mapsto w_1$.

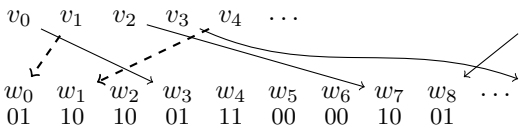

Fig. 3. Result of the second adaptation of the function f.

The result of the second adaptation is shown in Figure 3. Now $n = 2$ and $\sim_2$ relates words that have equal acceptance behavior with respect to automata A_0 and A_1. To this end, we now write $A_0(w_i)\, A_1(w_i)$ below each word w_i in Figure 3. The word w_2 is accepted by A_0 but rejected by A_1. The only (displayed) candidate for v_{k_2} exhibiting the same behavior is $f(v_2)$. The result of this third adaptation is shown in Figure 4.

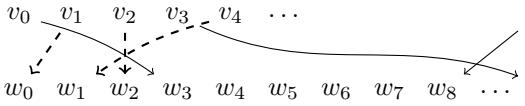

Fig. 4. Result of the third adaptation of the function f.

This process continues for every $n \in \mathbb{N}$, and the limit of this process is a bijective function g with the desired properties. The construction is made precise in the proof of Lemma 6.

Lemma 6. *Let A be a countably infinite set. For every injection $f : A \to \Gamma^*$ there exists a bijection $g : A \to \Gamma^*$ such that for all $L \subseteq A$, if $f[L]$ is relatively regular in the image $f[A]$, then $g[L]$ is a regular language.*

Proof. Let $f : A \to \Gamma^*$ be an injective function. Let

- $v_0, v_1, v_2, \ldots$ be an enumeration of the set A, and let
- $w_0, w_1, w_2, \ldots$ be an enumeration of the set Γ^*, and let
- $A_0, A_1, A_2, \ldots$ be an enumeration of all finite-state automata over Γ.

We abstract away from the domain by defining $c : \mathbb{N} \to \Gamma^*$ by

$$c(i) = f(v_i) \quad \text{for all } i \in \mathbb{N}.$$

In this proof, will construct a bijective function $d : \mathbb{N} \to \Gamma^*$ such that for all $S \subseteq \mathbb{N}$, if $c[S]$ is relatively regular in the image $c[\mathbb{N}]$, then $d[S]$ is a regular language.

Then we can define the function $g : A \to \Gamma^*$ by

$$g(w_i) = d(i) \qquad \text{for all } i \in \mathbb{N}.$$

The bijectivity of g follows immediately from bijectivity of d, and for every $L \subseteq A$ with $f[L]$ is relatively regular in $f[A]$, we have: let $S \subseteq \mathbb{N}$ such that $L = \{\, v_i \mid i \in S \,\}$, then $c[S] = f(L)$ is relatively regular in $c[\mathbb{N}] = f[A]$, and hence $d[S] = g[L]$ is regular. As a consequence, it suffices to construct a function d with the properties above.

For each $i \in \mathbb{N}$ we will define a finite-state automaton A'_i with properties described below. For every $n \in \mathbb{N}$, we define the equivalence relation $\sim_n \subseteq \Gamma^* \times \Gamma^*$ by

$$u \sim_n u' \iff \forall i \in \{0, 1, \ldots, n-1\}.\, A'_i(u) = A'_i(u')\,,$$

for all $u, u' \in \Gamma^*$. So $\sim_n$ relates words that are not distinguished by the automata $A'_0, A'_1, \ldots, A'_{n-1}$. The relation $\sim_n$ gives rise to a partition

$$E_n = \{\, [u]_{\sim_n} \mid u \in \Gamma^* \,\}$$

where $[u]_{\sim_n} = \{u' \in \Gamma^* \mid u \sim_n u'\}$ is the equivalence class of u with respect to $\sim_n$.

We are ready to carry out the central construction. We first describe what objects we will define and their main properties. Then we give specifications of these objects, and afterwards we show that the objects are well-defined and that the properties hold.

We are going to define, by induction on $n \in \mathbb{N}$,

(i) a natural number $k_n \in \overline{K}_n$, where

$$\overline{K}_n = \mathbb{N} \setminus K_n \qquad K_n = \{k_0, k_1, \ldots, k_{n-1}\}\,,$$

such that

$$c(k_n) \sim_n w_n\,. \tag{1}$$

(ii) a bijective map $d_n : \mathbb{N} \to I_n$, where

$$I_n = c[\overline{K}_n] \cup W_n \qquad W_n = \{w_0, w_1, \ldots, w_{n-1}\}\,,$$

(iii) an automaton A'_n, such that

(a) $\forall u \in I_n.\, A_n(u) = A'_n(u)$, and
(b) E_{n+1} is attracted to I_{n+1}.

We will now give specifications of (i)–(iii), and thereafter show that they are well-defined, and satisfy the above mentioned properties.

(i) We define $k_n \in \overline{K}_n$ as follows:

$$k_n \;=\; c^{-1}(\min([w_n]_{\sim_n} \cap c[\overline{K}_n]))\,,$$

where $\min$ denotes the minimum with respect to the order $<$ on Γ^* given by the enumeration $w_0 < w_1 < w_2 < \cdots$.

(ii) The map $d_n : \mathbb{N} \to I_n$ is defined for all $i \in \mathbb{N}$ by

$$d_n(i) = \begin{cases} w_j & \text{if } i = k_j \text{ for some } j \in \{0, 1, \ldots, n-1\}\,, \\ c(i) & \text{if } i \in \overline{K}_n. \end{cases}$$

(iii) For A'_n we choose the automaton A' guaranteed to exist by Lemma 4 where the lemma is invoked with $A = A_n$, $I = I_{n+1}$, and $E = E_n$.

We note that $\sim_0 = \Gamma^* \times \Gamma^*$, and, by definition of $\sim_n$, it follows that

$$E_n \text{ is finite and every } C \in E_n \text{ is a regular language.} \tag{2}$$

We prove that (i)–(iii) are well-defined, and that the following properties hold:

(a) $c[\overline{K}_n] \cap W_n = \varnothing$,
(b) E_n is attracted to I_n,

(c) d_n is a bijection, and
(d) k_n is well-defined.

We first show that, for every $n \in \mathbb{N}$, items (c) and (d) follow from (a) and (b).

(c) We show that $d_n : \mathbb{N} \to I_n$ is bijective. Surjectivity of d_n is immediate by definition of I_n. To show injectivity, assume there exist $i_1, i_2 \in \mathbb{N}$ such that $d_n(i_1) = d_n(i_2)$. From (a) and the definition of d_n, it follows that either $i_1, i_2 \in K_n$ or $i_1, i_2 \in \overline{K}_n$. If $i_1, i_2 \in K_n$, then there are $j_1, j_2 \in \{0, 1, \ldots, n-1\}$ such that $i_1 = k_{j_1}$ and $i_2 = k_{j_2}$. Then it follows that $w_{j_1} = d_n(i_1) = d_n(i_2) = w_{j_2}$, which can only be in case if $j_1 = j_2$. Hence $i_1 = k_{j_1} = k_{j_2} = i_2$. If $i_1, i_2 \in \overline{K}_n$, then $c(i_1) = d_n(i_1) = d_n(i_2) = c(i_2)$. By injectivity of c we have $i_1 = i_2$.

(d) To see that k_n is well-defined it suffices to show that $[w_n]_{\sim_n} \cap c[\overline{K}_n]$ is non-empty. By (b) we know that either $|[w_n]_{\sim_n} \cap I_n| = \infty$, or $[w_n]_{\sim_n} \subseteq I_n$:
  - For $|[w_n]_{\sim_n} \cap I_n| = \infty$, it follows that $|[w_n]_{\sim_n} \cap c[\overline{K}_n]| = \infty$.
  - For $[w_n]_{\sim_n} \subseteq I_n$, we get $w_n \in ([w_n]_{\sim_n} \cap I_n)$. Since $I_n = c[\overline{K}_n] \cup W_n$, and $w_n \notin W_n$, (a) entails $w_n \in c[\overline{K}_n]$, and hence $[w_n]_{\sim_n} \cap c[\overline{K}_n] \neq \varnothing$.

  We note that, indeed, (1) follows by the choice of k_n.

We now prove (a) and (b) by induction on $n \in \mathbb{N}$. For the base case we have:

(a) $c[\overline{K}_0] \cap W_0 = \varnothing$ since $W_0 = \varnothing$, and
(b) $E_0 = \{\Gamma^*\}$ is attracted to $I_0 = c[\mathbb{N}]$.

For the induction step, let $n \in \mathbb{N}$ be arbitrary. We assume $c[\overline{K}_n] \cap W_n = \varnothing$ and that E_n is attracted to I_n (induction hypothesis). We first prove the implication

$$w_n \in c[\overline{K}_n] \implies w_n = c(k_n)\,. \tag{3}$$

If $w_n \in c[\overline{K}_n]$, also $w_n \in ([w_n]_{\sim_n} \cap c[\overline{K}_n])$. Since $c[\overline{K}_n] \cap W_n = \varnothing$ by induction hypothesis, w_n is the smallest element in $[w_n]_{\sim_n} \cap c[\overline{K}_n]$. Hence $w_n = \min([w_n]_{\sim_n} \cap c[\overline{K}_n])$ and $c(k_n) = w_n$.

(a) By the induction hypothesis we have $c[\overline{K}_n] \cap W_n = \varnothing$, and so we have $c[\overline{K}_{n+1}] \cap W_n = \varnothing$. To see that $c[\overline{K}_{n+1}] \cap W_{n+1} = \varnothing$, it suffices to show $w_n \notin c[\overline{K}_{n+1}]$. Assume, to derive a contradiction, that $w_n \in c[\overline{K}_{n+1}]$. Then also $w_n \in c[\overline{K}_n]$. Hence, by (3), we have $w_n = c(k_n)$. This contradicts $w_n \in c[\overline{K}_{n+1}]$, since, by injectivity of c, we have $c[\overline{K}_{n+1}] = c[\overline{K}_n] \setminus \{c(k_n)\} = c[\overline{K}_n] \setminus \{w_n\}$.

(b) We have to prove that E_{n+1} is attracted to I_{n+1}. We first show that

$$E_n \text{ is attracted to } I_{n+1}. \tag{4}$$

By injectivity of c we have $I_{n+1} = c[\overline{K}_n] \setminus \{c(k_n)\} \cup W_n \cup \{w_n\}$. Moreover, since by induction hypothesis $c[\overline{K}_n] \cap W_n = \varnothing$, we have $c[\overline{K}_n] \setminus \{c(k_n)\} \cup W_n \cup \{w_n\} = (c[\overline{K}_n] \cup W_n) \setminus \{c(k_n)\} \cup \{w_n\}$. Hence $I_{n+1} = (I_n \setminus \{c(k_n)\}) \cup \{w_n\}$. Let $C \in E_n$ be arbitrary. By induction hypothesis we know that C is attracted to I_n; we distinguish the following two cases:

  - If $|C \cap I_n| = \infty$, then $|C \cap I_{n+1}| = \infty$. Hence C is attracted to I_{n+1}.
  - Assume $C \subseteq I_n$. If $c(k_n) \notin C$, then clearly $C \subseteq I_{n+1}$. If $c(k_n) \in C$, then $C = [w_n]_{\sim_n}$ since $c(k_n) \sim_n w_n$ due to (1). From $w_n \in C \subseteq I_n$, we obtain $w_n \in c[\overline{K}_n]$. Hence, by (3) we have $w_n = c(k_n)$ and so $I_n = I_{n+1}$.

  This concludes the proof of (4).

  Recall that A'_n is the automaton A' obtained by invoking Lemma 4 with $A = A_n$, $I = I_{n+1}$, and $E = E_n$. Both requirements of the lemma are established above, in (2) and (4), so the automaton A'_n is well-defined. Let, moreover, E' be the resulting partition obtained in the lemma. Lemma 4 guarantees that E' is attracted to I_{n+1}. Moreover, by definition of E_{n+1}, we have $E' = E_{n+1}$. Hence E_{n+1} is attracted to I_{n+1}.

We now establish that every natural number k will be picked as a k_n eventually. That is, for every $k \in \mathbb{N}$ there exists $n \in \mathbb{N}$ such that $k_n = k$. Let $k \in \mathbb{N}$. For every $n \in \mathbb{N}$, the set $c[\overline{K}_n]$ is the image under c of all natural numbers that have not yet been picked as a k_i for $0 \leq i \leq n-1$. Since $w_0, w_1, \ldots$ is an enumeration of Γ^*, there exists $n \in \mathbb{N}$ such that $c(k) \in W_n$. Hence $k \notin \overline{K}_n$ since $c[\overline{K}_n] \cap W_n = \varnothing$ by property (a). Thus k will be picked eventually.

We define the encoding $d : \mathbb{N} \to \Gamma^*$ by

$$d(i) = \lim_{n \to \infty} d_n(i) \qquad (i \in \mathbb{N})$$

These limits are well-defined since, for every $i \in \mathbb{N}$: there exists $n \in \mathbb{N}$ with $i = k_n$, and we have $d_m(i) = c(i)$ for all $m \leq n$ and $d_m(i) = w_n$ for all $m > n$.

From the above, it follows that the function d has the following property:

$$d(k_n) = w_n \quad \text{for all } n \in \mathbb{N}\,. \tag{5}$$

We now show that d is indeed a bijection. Since by construction, every $w_0, w_1, w_2, \ldots$ occurs precisely once in the image of d, d is injective. Moreover, we have seen above, that the set $\{k_0, k, k_2, \ldots\}$, the domain of d, contains all natural numbers. Hence d is a bijection.

Finally, we show that for all $L \subseteq \mathbb{N}$, if $c[L]$ is relatively regular in the image $c[\mathbb{N}]$, then $d[L]$ is a regular language. Let $L \subseteq \mathbb{N}$ be such that $c[L]$ is relatively regular in $c[\mathbb{N}]$. Then there exists a regular language R such that $c[L] = R \cap c[\mathbb{N}]$. Let $m \in \mathbb{N}$ be such that $R = \mathrm{Lang}(A_m)$. Then for all $n \in \mathbb{N}$,

$$n \in L \iff c(n) \in \mathrm{Lang}(A_m)\,. \tag{6}$$

By the above construction we have for all $w \in I_m$

$$w \in \mathrm{Lang}(A'_m) \iff w \in \mathrm{Lang}(A_m)\,.$$

By definition, I_m coincides with $c[\mathbb{N}]$ for all but finitely many words. Hence for almost all $n \in \mathbb{N}$

$$c(n) \in \mathrm{Lang}(A_m) \iff c(n) \in \mathrm{Lang}(A'_m)\,. \tag{7}$$

By the definition of d, and property (1), we find:

$$d(k_n) = w_n \sim_n c(k_n) \qquad \text{for all } n \in \mathbb{N}\,.$$

Due to $\sim_n \subseteq \sim_m$ for every $n \geq m$, we obtain that

$$d(k_n) \sim_m c(k_n) \qquad \text{for all } n \geq m\,,$$

and hence, since $\mathbb{N} = \{k_n : n \in \mathbb{N}\}$ holds, that

$$d(i) \sim_m c(i) \qquad \text{for almost all } i \in \mathbb{N}\,.$$

Hence for almost all n we have

$$\begin{aligned} d(n) \in \mathrm{Lang}(A'_m) &\iff c(n) \in \mathrm{Lang}(A'_m) \\ &\iff c(n) \in \mathrm{Lang}(A_m) \qquad \text{by (7)} \\ &\iff n \in L \qquad \text{by (6).} \end{aligned}$$

As d is bijective, we obtain

$$w \in \mathrm{Lang}(A'_m) \iff w \in d(L)$$

for almost all $w \in \Gamma^*$. Hence $d(L)$ differs only by finitely many elements from a regular language and is consequently itself regular. □

The following proposition states that, under certain conditions, the bijective function g constructed in Lemma 6 is computable. Obviously, the injective function f that is lifted to g must be computable to start with. Moreover, we need to be able to decide for regular languages whether their intersection with the image of f is empty, finite or infinite. This enables us, in case of a finite intersection, to compute this intersection, and to decide whether the equivalence classes E_n are attracted to the image of f (and I_n). This suffices to ensure computability of g constructed in the proof of Lemma 6.

Proposition 7. *Let A be a computable, countably infinite set and let $f : A \to \Gamma^*$ be a computable injection such that for every regular language R (given by an automaton), emptiness and finiteness of the set $R \cap f[\Gamma^*]$ is decidable. There exists a computable bijection $g : A \to \Gamma^*$ such that for all $L \subseteq A$, if $f[L]$ is relatively regular in the image $f[A]$, then $g[L]$ is a regular language.*

Proof. By close inspection of the proofs of Lemmas 4 and 6 it can be established that the construction of the encoding g from the encoding f preserves computability under the stated conditions and by choosing computable enumerations $v_0, v_1, \ldots$ and $w_0, w_1, \ldots$ of A and Γ^*, respectively. □

We are ready to state our main results.

Theorem 8. *Let Σ and Γ be finite alphabets, with $|\Gamma| \geq 2$. Let $\mathcal{L} \subseteq \wp(\Sigma^*)$ be a countable set of formal languages. There exists a bijective function $g : \Sigma^* \to \Gamma^*$ such that for every $L \in \mathcal{L}$, the image $g[L]$ is a regular language.*

Proof. By Lemmas 2 and 6. □

The function constructed in the proof of Theorem 8 is not guaranteed to be computable. What is more, if $\mathcal{L}$ contains the recursive languages, then there is no *computable* function with the properties as stated in Theorem 8.

The following theorem justifies the title of this paper.

Theorem 9. *There exists a computable bijective function $f : \Sigma^* \to \Gamma^*$ such that the image function $f[\_]$ is regularity preserving, but the preimage function $f^{-1}[\_]$ is not.*

Without the requirement on the function f to be computable, we could prove the statement as follows: Let $\mathcal{L} \subseteq \wp(\Sigma^*)$ be the (countable) set of all recursive languages over Σ. By Theorem 8 there exists a bijective mapping $f : \Sigma^* \to \Gamma^*$ such that $f[L]$ is regular for all $L \in \mathcal{L}$. Then, clearly, $f[\_]$ is regularity preserving while $f^{-1}[\_]$ is not. However, the function f obtained in this way is not computable. To obtain a computable function, we argue as follows.

Proof of Theorem 9. Let $\Sigma = \{0, 1\}$. We define a 'balancedness' function $b : \Sigma^* \to \Sigma$ for all $w \in \Sigma^*$ by $b(w) = 1$ if the word w contains an equal number of zeros and ones, and $b(w) = 0$ otherwise. Then we define $f : \Sigma^* \to \Sigma^*$ by

$$f(w) = b(w)w$$

for every $w \in \Sigma^*$. Then we have:

(i) For every regular language $L \subseteq \Sigma^*$, we have that $f[L]$ is relatively regular in $f[\Sigma^*]$ (an automaton can simply ignore the first letter).

(ii) However, the function $f^{-1}[\_]$ is not preserving (relative) regularity. To see this, let $X = \{w \mid b(w) = 1\}$. Clearly $f^{-1}[f[X]] = X$ is not regular. But $f[X]$ is relatively regular in $f[\Sigma^*]$, since $f[X]$ consists precisely of those words in $f[\Sigma^*]$ that start with letter 1.

We now invoke Proposition 7 for lifting f to a computable, bijective g. The conditions of the proposition are satisfied since f is computable, and the image $f[\Sigma^*]$ of f is a context-free language. The intersection of a context-free language with a regular language is context-free, and finiteness and emptiness are decidable. The proposition guarantees the existence of a computable, bijective function $g : \Sigma^* \to \Sigma^*$ such that for every $L \subseteq \Sigma^*$ that is relatively regular in $f[\Sigma^*]$ we have that $g[L]$ is regular. Then by (i) we have that $g[\_]$ preserves regularity. From (ii) it follows that $g[X]$ is regular while $g^{-1}[g(X)] = X$ is not. Hence $g^{-1}[\_]$ does not preserve regularity. □

We strengthen the statement of Theorem 9 by extending it to the preservation of membership in countable classes of languages that include the regular languages. For this, we use the following stipulation. For an arbitrary set $\mathcal{S}$ of languages over A, we say that a function $F : \wp(A^*) \to \wp(B^*)$ preserves membership in $\mathcal{S}$ if, for all languages over A, $L \in \mathcal{S}$ implies $F(L) \in \mathcal{S}$.

Corollary 10. *Let $\mathcal{S} \subseteq \wp(\Sigma^*)$ be a countable set of languages that includes the regular languages. Then there exists a bijective function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f[\_]$ preserves membership in $\mathcal{S}$, but $f^{-1}[\_]$ does not.*

Proof. Let $\mathcal{S}' = \mathcal{S} \cup \{X\}$ where X is a language not in $\mathcal{S}$. By Theorem 8 there exists a bijective function $f : \Sigma^* \to \Gamma^*$ such that $f[L]$ is regular for all $L \in \mathcal{S}'$. Then, $f[\_]$ preserves

membership in $\mathcal{S}$ while $f^{-1}[\_]$; note that $f(X)$ is regular and hence $f(X) \in \mathcal{S}$, but $f^{-1}[f(X)] = X \notin \mathcal{S}$. □

IV. Consequences for Comparing Models of Computation

It turns out that our main results have some remarkable consequences in the context of comparing computational models using concepts proposed by Boker and Dershowitz in a series of publications [3], [5], [4]. The authors summarize their goal as follows:

> "We seek a robust definition of relative power that does not itself depend on the notion of computability. It should allow one to compare arbitrary models over arbitrary domains via a quasi-ordering that successfully captures the intuitive concept of computational strength. [...][1]" [5]

This motivation leads them to specific choices of simple and liberal conditions on encodings. Encodings are typically used to translate between models of computation that act on different domains. So an encoding $\rho : D_1 \to D_2$ can facilitate the simulation of the input-output behavior of a machine M_1 belonging to a model $\mathcal{M}_1$ with domain D_1 by a machine M_2 from a model $\mathcal{M}_2$ with domain D_2:

$$\begin{array}{ccc} D_1 & \xrightarrow{\rho} & D_2 \\ {\scriptstyle M_1}\downarrow & & \downarrow{\scriptstyle M_2} \\ D_1 & \xrightarrow[\rho]{} & D_2 \end{array}$$

To prevent codings from participating too strongly in the simulation of a computation on a machine M_1 through a computation on a machine M_2 (and thereby from substantially alleviating, for the simulating machine M_2, the task that is solved by the simulated machine M_1), codings are usually required to be computable in some sense. Frequently, one of the following two restrictions are stipulated (see for example Rogers' classic book [21, p. 27,28]):

(1) Codings must be 'informally algorithmic', 'informally computable', or 'effective' in the sense that they can be carried out by an in principle mechanizable procedure.
(2) Codings are required to be computable with respect to a specific model, for example by a Turing machine.

Boker and Dershowitz reject such prevalent stipulations:

> "Effectivity is a useful notion; however, it is unsuitable as a general power comparison notion. The first, informal approach is too vague, while the second can add computational power when dealing with subrecursive models and is inappropriate when dealing with non-recursive models." [4]

As a consequence, they go on to use classes of encodings that do not constrain (at least not explicitly) the cost that is necessary to compute an encoding. In particular, they define three concepts of comparison (see Definition 17 below) that are, respectively, based on:[2]

(i) encodings (injective functions) without any additional requirement;
(ii) encodings that are 'decent' with respect to the simulating model $\mathcal{M}_2$ that shoulders the simulation, in the sense that $\mathcal{M}_2$ is able to recognize the image of the coding;
(iii) bijective encodings.

We will show that each of these concepts admits some quite counterintuitive consequences. At first these anomalies pertain only to decision models, the subclass of all models that only obtain 'yes'/'no' as computation result. But it turns out these phenomena apply also to more broad classes of models.

In order to formally state our results, we repeat here the basic definitions in [5], [4], and extend them by straightforward adaptations for decision models.

By abstracting away from all intensional aspects of models of computation that concern mechanistic aspects of stepwise computation processes, Boker and Dershowitz define a model extensionally as an arbitrary set of (extensionally represented) partial functions over some domain.

Definition 11 ([5, Def. 2.1])**.** A *model of computation* is a pair $\mathcal{M} = \langle D, \mathcal{F} \rangle$, where D is a set of elements, the *domain of* $\mathcal{M}$, and $\mathcal{F} = \{f \mid f : D \to D \cup \{\bot\}\}$ is a set of functions with $\bot \notin D$. We write $\mathrm{dom}_{\mathcal{M}}$ for the domain of $\mathcal{M}$. (We assume that $\bot$ is a fixed element not contained in the domain of any model.)

We define decision models as models of computation consisting of total functions that yield a definite 'yes'/'no' answer.

Definition 12. A *decision model* (model of computation for decision models) is a model of computation $\mathcal{M} = \langle D, \mathcal{F} \rangle$, such that $\{0, 1\} \subseteq D$, and $f[D] \subseteq \{0, 1\}$, for all $f \in \mathcal{F}$. We say that $\mathcal{M}$ is *countable* if $\mathcal{F}$ is countable.

Now codings between domains of models are defined.

Definition 13 ([5, Def. 2.2])**.** Let D_1 and D_2 be domains of models of computation. A *coding (from D_1 to D_2)* is an injective function $\rho : D_1 \cup \{\bot\} \to D_2 \cup \{\bot\}$ such that $\rho(x) = \bot$ if and only if $x = \bot$, for all $x \in D_1$.

For codings ρ between decision models it could be desirable to demand that $\rho(0) = 0$ and $\rho(1) = 1$. We do not to take up this restriction, for a pragmatic reason connected to the definition of 'simulation' immediately below. If namely a non-constant function f in a decision model $\mathcal{M}_1$ is simulated via ρ by a function g in a decision model $\mathcal{M}_2$, then it follows that either $\rho(0) = 0$ and $\rho(1) = 1$, or $\rho(0) = 1$ and $\rho(1) = 0$. In both cases it can be said that decisions taken by f are faithfully modelled by corresponding decisions taken by g.

[1] This passage continues: "Eventually, we want to be able to prove statements like 'analogue machines are strictly more powerful than digital devices', even though the two models operate over domains of different cardinalities."

[2] Note that encodings of the notions (i) and (ii) here are more liberal than those in (1) and (2) above, and that therefore the use of such encodings does not address the concern raised in the preceding quotation regarding the addition of computational power when dealing with subrecursive models.

Definition 14 ([5, Def. 2.3])**.** Let $\mathcal{M}_1 = \langle D_1, \mathcal{F}_1 \rangle$ and $\mathcal{M}_2 = \langle D_2, \mathcal{F}_2 \rangle$ be models of computation. Let ρ be a coding from D_1 to D_2. We define:

(i) For $g \in \mathcal{F}_2$ and $f \in \mathcal{F}_1$ we say that g *simulates* f *via* ρ if $g \circ \rho = \rho \circ f$ holds, as in the following diagram:

$$\begin{array}{ccc} D_1 & \xrightarrow{\rho} & D_2 \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ D_1 & \xrightarrow[\rho]{} & D_2 \end{array}$$

(ii) $\mathcal{M}_2$ *simulates* $\mathcal{M}_1$ *via* ρ, $\mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$, if for every $f \in \mathcal{F}_1$ there is a $g \in \mathcal{F}_2$ such that g simulates f via ρ.

The 'decency' requirement for codings mentioned before is defined as follows in [4]. There, Boker stresses that this requirement follows classic definitions of computable groups by Rice [19, p. 298] and Rabin [18, p. 343].

Definition 15 ([4, Def. 52])**.** Let $\mathcal{M}_1 = \langle D_1, \mathcal{F}_1 \rangle$ and $\mathcal{M}_2 = \langle D_2, \mathcal{F}_2 \rangle$ be models of computation. A coding ρ from D_1 to D_2 is called *decent* with respect to $\mathcal{M}_2$ if the image $\rho[D_1]$ can be recognized by $\mathcal{M}_2$, in the sense that there is a total function $g \in \mathcal{F}_2$ such that $\rho[D_1] = g[D_2]$ and for all $y \in D_2$, we have $g(y) = y$ if and only if $y \in \rho[D_1]$.

We note that according to Definition 15 a coding ρ can be decent with respect to a decision model $\mathcal{M}_2$ only if $\rho[D_1] \subseteq \{0, 1\}$. Since ρ is injective, $|D_1| \leq 2$ follows, and so $\mathcal{M}_1$ can only be a rather trivial model. Therefore we adapt the notion of decency in an obvious way to accommodate decision models.

Definition 16. Let $\mathcal{M}_1 = \langle D_1, \mathcal{F}_1 \rangle$ and $\mathcal{M}_2 = \langle D_2, \mathcal{F}_2 \rangle$ be models of computation. A coding ρ from D_1 to D_2 is called *decent** with respect to $\mathcal{M}_2$ if the image $\rho[D_1]$ can be recognized by $\mathcal{M}_2$, in the sense that there is a total function $g \in \mathcal{F}_2$ and an element $d \in D_2$ such that for all $y \in D_2$, we have $g(y) = d$ if and only if $y \in \rho[D_1]$.

With the concepts 'model of computation' and 'simulation' defined, Boker and Dershowitz introduce three comparison preorders for models, which are based on three classes of codings as mentioned above. In addition to the preorder induced by decent codings, we also define a variant preorder induced by decent* codings.

Definition 17 ([4, Def. 52])**.** Let $\mathcal{M}_1 = \langle D_1, \mathcal{F}_1 \rangle$ and $\mathcal{M}_2 = \langle D_2, \mathcal{F}_2 \rangle$ be models of computation. We define:

(i) $\mathcal{M}_2$ *is at least as powerful as* $\mathcal{M}_1$, denoted by
$$\mathcal{M}_1 \lesssim \mathcal{M}_2\,,$$
if $\mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$ for some ρ.

(ii) $\mathcal{M}_2$ *is at least as powerful as* $\mathcal{M}_1$ *via a decent coding*, which we denote by
$$\mathcal{M}_1 \lesssim_{\text{decent}} \mathcal{M}_2\,,$$
if $\mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$ for some decent coding ρ with respect to $\mathcal{M}_2$. $\mathcal{M}_2$ *is at least as powerful as* $\mathcal{M}_1$ *via a decent** *coding*, which we denote by
$$\mathcal{M}_1 \lesssim_{\text{decent}^*} \mathcal{M}_2\,,$$
if $\mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$ for some decent* coding ρ with respect to $\mathcal{M}_2$.

(iii) $\mathcal{M}_2$ *is at least as powerful as* $\mathcal{M}_1$ *via a bijective coding*, which we denote by
$$\mathcal{M}_1 \lesssim_{\text{bijective}} \mathcal{M}_2\,,$$
if $\mathcal{M}_1 \lesssim_\rho \mathcal{M}_2$ for some bijective coding ρ.

With these definitions in place, we are now able to state, and prove, our results concerning the comparison of decision models. Let Γ be an alphabet with $\{0, 1\} \subseteq \Gamma$. We write $\mathsf{DFA}(\Gamma) = \langle \Gamma^*, \mathcal{D} \rangle$ for the decision model corresponding to DFAs, so with $\mathcal{D}$ the set of characteristic functions of regular languages over Γ, that is,

$$\mathcal{D} = \Big\{ f : \Gamma^* \to \{0,1\} \;\Big|\; \begin{array}{l} \exists M \text{ DFA } \forall w \in \Gamma^* \\ \quad (\, f(w) = 1 \Leftrightarrow M \text{ accepts } w\,) \end{array} \Big\}.$$

The model $\mathsf{TMD}(\Gamma)$ over input alphabet Γ of Turing-machine deciders is defined analogously.

The proposition below is an easy consequence of Lemma 2.

Proposition 18. *Let* Σ, Γ *be alphabets, where* $\{0, 1\} \subseteq \Gamma$. *Then for every countable decision model* $\mathcal{M}$ *with domain* Σ^*,

$$\mathcal{M} \;\lesssim \mathsf{DFA}(\Gamma) \tag{8}$$

holds, that is, deterministic finite state automata with input alphabet Γ *are at least as powerful as* $\mathcal{M}$.

Proof. Every decision model $\mathcal{M} = \langle \Sigma^*, \mathcal{F} \rangle$ with domain Σ^* and with countable set $\mathcal{F}$ of computed functions corresponds to the countable set $\mathcal{L}_\mathcal{M} = \{L_f \mid f \in \mathcal{F}\}$ of languages that are defined, for $f \in \mathcal{F}$, as $L_f = \{w \in \Sigma^* \mid f(w) = 1\}$. By Lemma 2 there exists an injective function $\rho : \Sigma^* \to \Gamma^*$ such that $\rho[L_f]$ is relatively regular in $\rho[\Sigma^*]$, for all $f \in \mathcal{F}$. Now it is straightforward to verify that ρ is a coding between the domains of the models $\mathcal{M}$ and $\mathsf{DFA}(\Gamma)$ that facilitates the simulation of every $f \in \mathcal{F}$ by a function $g : \Gamma^* \to \{0, 1\}$ that denotes the acceptance/non-acceptance behavior of a deterministic finite-state automaton with input alphabet Γ. This shows (8). □

Proposition 19. *Let* Γ *be an alphabet with* $\{0, 1\} \subseteq \Gamma$. *Then there is a coding* $\rho : \Gamma^* \to \Gamma^*$ *such that* $\mathsf{TMD}(\Gamma) \lesssim_\rho \mathsf{DFA}(\Gamma)$ *holds. But any such a coding* ρ *cannot be computable.*

Proof. The main statement follows from Proposition 18. That ρ cannot be computable can be seen as follows. Suppose that ρ is computable. Let $A_0, A_1, \ldots$ and $w_0, w_1, \ldots$ be recursive enumerations of all finite automata and words over Σ. Then the language $L = \{w_n \mid n \in \mathbb{N},\, \rho(w_n) \notin \mathrm{Lang}(A_n)\}$ is Turing computable, but $\rho[L]$ is not regular. □

This statement can be strengthened to a bijective, and therefore (see the proof) also decent*, simulation with finite-state automata by using Lemma 6 and Theorem 8.

Corollary 20. *Let* Σ *and* Γ *be alphabets, where* $\{0, 1\} \subseteq \Gamma$. *Then for every countable decision model* $\mathcal{M}$ *with domain* Σ^* *the following two statements hold:*

(i) $\mathcal{M} \;\lesssim_{\text{bijective}} \mathsf{DFA}(\Gamma)$,

(ii) $\mathcal{M} \lesssim_{\mathrm{decent}^*} \mathsf{DFA}(\Gamma)$.

That is, deterministic finite-state automata with input alphabet Γ are at least as powerful as the model $\mathcal{M}$, both via bijective and via decent* codings.

Proof. Statement (i) follows by an argumentation analogous to the proof of Proposition 18 in which the use of Lemma 2 is replaced by an appeal to our main theorem, Theorem 8.

Statement (ii) follows directly from statement (i): For bijective codings $\rho : \Sigma^* \to \Gamma^*$, the image of ρ coincides with Γ^*, which is trivially recognizable by a finite-state automaton. □

Remark 21. As mentioned above (just before Definition 16), decent codings in the sense of [4] do not form a sensible notion for decision models. However, for every countable decision model $\mathcal{M}$ and every alphabet Γ with $\{0,1\} \subseteq \Gamma$ it holds that

$$\mathcal{M} \lesssim_{\mathrm{decent}} \mathsf{DFA}_{\mathrm{id}}(\Gamma)$$

where $\mathsf{DFA}_{\mathrm{id}}(\Gamma)$ is the extension of $\mathsf{DFA}_{\mathrm{id}}(\Gamma)$ by adding the identity function $\mathrm{id} : \Gamma^* \to \Gamma^*$.

Sequential finite-state transducers $\mathsf{FST}(\Gamma)$ (see e.g. [22]) over alphabet Γ form a natural computational model that extends $\mathsf{DFA}_{\mathrm{id}}(\Gamma)$. Thus for an alphabet Γ with $\{0,1\} \subseteq \Gamma$ we also obtain the very counterintuitive result:

$$\mathcal{M} \lesssim_{\mathrm{decent}} \mathsf{FST}(\Gamma)\,,$$

that is, sequential finite-state transducers are at least as strong via a decent coding as every countable decision model. (Here we consider finite-state transducers that are able to recognize the end of a word.) In particular, every Turing-machine decider can be simulated by a sequential finite state transducer via a decent coding.

Remark 22. These results raise the question, whether these anomalies only concern decision models. In particular, one may wonder whether the comparison of *computational models* avoids counterintuitive results when additional requirements are imposed on the models that are compared. A candidate requirement would be to enforce that the output of the models must have an infinite range. Or, even stronger, we could require the following property: A class of models $\mathcal{M} = \langle D, \mathcal{F} \rangle$ is *image-complete* if for every non-empty computable set $I \subseteq D$ there exists $f \in \mathcal{F}$ such that $f[D] = I$.

Let $\mathcal{T} = \langle \Sigma^*, \mathcal{F} \rangle$ consist of all Turing machines such that for every $f \in \mathcal{F}$ we have there exists a finite set $L_f \subseteq \Sigma^*$ and for all $x \in \Sigma^*$ we have $f(x) \in \{x\} \cup L_f \cup \{\bot\}$. Thus functions $f \in \mathcal{T}$ map words either to themselves or into a finite set that may depend on f. The class $\mathcal{T}$ is a natural model because it can be implemented by a recursively enumerable set of Turing machines[3]. Note that $\mathcal{T}$ is a strict extension of Turing-machine deciders and it is an image-complete model. This can be seen as follows: Let $I \subseteq \Sigma^*$ be any non-empty computable set. Let $i \in I$ and define the function $f : \Sigma^* \to \Sigma^*$ for all $x \in \Sigma^*$ by: $f(x) = x$ if $x \in I$ and $f(x) = i$, otherwise. Then $f[\Sigma^*] = I$ and $f \in \mathcal{T}$.

Then the model $\mathcal{T}$ can be simulated by two-way sequential, finite-state transducers [8] via decent codings:

$$\mathcal{T} \lesssim_{\mathrm{decent}} \text{2-}\mathsf{FST}(\Gamma)\,.$$

We have already argued that the Turing-machine deciders can be simulated by $\mathsf{FST}(\Gamma)$. This can easily be generalized to a finite number of output words L (finite-state transducers can output words instead of only symbols). Now we assume that one symbol $w \in L$ symbolizes the identity output; then the two-way finite-state transducer, instead of producing this output word, can walk back to the beginning of the input and reproduce the input word as output word.

Our results suggest that there are definite limitations to the concepts of power comparison for models of computation by Boker and Dershowitz. These concepts have an 'absolute' flavor insofar as they do not formulate any explicit constraints on the computability of encodings used for simulations. The counterintuitive consequences pertain primarily to decision models (yet this is a blurry concept, see Remark 22), and do not extend to models that include all partial-recursive functions (see Corollary 24 below). Yet they demonstrate that these comparison concepts lack the desired robustness.

We note that our results are not the first indications of anomalies. In [5, Example 5.1] Boker and Dershowitz show that Turing-machine deciders are not a complete model of computation, in the sense that this model can be strictly extended to incorporate a non-recursive set. Our results strengthen this example naturally in the following three ways: (i) to bijective, and decent encodings, (ii) to use finite automata instead of Turing-machine deciders, and (iii) to arbitrary countable models as extensions. This is because, as we have shown, finite automata can be extended, via decent codings, to any countable decision model, and consequently the same holds for Turing-machine deciders. Hence, even decent encodings facilitate the simulation of all Turing-machine deciders by finite automata, more precisely, by finite-state transducers.

The following theorem is an easy consequence of the concepts developed by Shapiro in [26]. He calls a number representation $r : \mathbb{N} \to \Sigma^*$ *acceptable* if it is bijective, and if the successor function $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$ can be simulated by a Turing machine on the representations, that is, if the lifting $\mathsf{succ}^r : \Sigma^* \to \Sigma^*$ of succ with the property:

$$\mathsf{succ}^r(r(n)) = r(\mathsf{succ}(n))\,, \qquad \text{for all } n \in \mathbb{N},$$

is Turing computable.

Theorem 23. *A bijective encoding $f : \Sigma^* \to \Gamma^*$ is computable if and only if there is an acceptable number representation r such that $f \circ r$ is acceptable as well.*

This theorem implies that Corollary 20 (i) does not generalize to models that compute the partial-recursive functions.

[3]The idea is to enumerate all Turing machines and finite sets L_f, and adapt the machines to check on termination whether the output is in L_f and otherwise make sure that the output is equal to the input. In this way, we obtain all machines that are necessary to implement $\mathcal{T}$.

Corollary 24. *There is no bijective encoding $\rho : \Sigma^* \to \Sigma^*$ such that $\mathsf{TM}(\Sigma) \lesssim_\rho 2\text{-}\mathsf{FST}(\Sigma)$ holds, where $\mathsf{TM}(\Sigma)$ is the model of Turing machines over alphabet Σ.*

While certainly more investigation is needed, we also interpret our results as follows. For comparing the computational power of models of computation over different domains, it is crucial to make clear how computational power should be measured for the purpose at hand. After having settled on a reasonable measure, this measure can then be used to constrain the computational power of admissible codings that may act as a trustworthy intermediary between the models.

V. Consequences for Generalized Automaticity

Finite-state automata can be used to generate infinite sequences, see [1]. This is usually done using the standard base-k representation $(n)_k \in \{0, 1, \ldots, k-1\}^*$ of the natural numbers $n \in \mathbb{N}$. A sequence $\sigma \in \Delta^{\mathbb{N}}$ is called *k-automatic* if, for some deterministic finite-state automaton A with output (DFAO, see Section II) we have $\sigma(n) = A((n)_k)$ for all $n \in \mathbb{N}$.

This concept has been generalized in several ways, where different number representations are fed to the automaton, see, e.g., [25], [20], [7]. This motivates the study of automaticity with respect to arbitrary number representations, which is part of work in progress of the present authors with Clemens Kupke, Larry Moss, and Jan Rutten.

Definition 25. Let $c : \mathbb{N} \to \Gamma^*$ be a number representation. A sequence $\sigma \in \Delta^{\mathbb{N}}$ over an alphabet Δ is *c-automatic* if there exists a DFAO A such that $\sigma(n) = A(c(n))$ for all $n \in \mathbb{N}$.

Lemma 26. *Let $c : \mathbb{N} \to \Gamma^*$ be a number representation. A sequence $\sigma \in \Delta^{\mathbb{N}}$ is c-automatic if and only if for every $a \in \Delta$ the 'fiber' $\{c(n) \mid \sigma(n) = a\}$ is relatively regular in $c[\mathbb{N}]$.*

Proof. Along the lines of Lemma 5.2.6 in [1]. □

Corollary 27. *For every injective function $c : \mathbb{N} \to \Gamma^*$ there is a bijective function $d : \mathbb{N} \to \Gamma^*$ such that for every $\sigma \in \Delta^{\mathbb{N}}$ we have: if σ is c-automatic, then σ is also d-automatic.*

Proof. Let $c : \mathbb{N} \to \Gamma^*$ be an injection. Let $d : \mathbb{N} \to \Gamma^*$ be the bijective function obtained from c by Lemma 6. Let $\sigma \in \Delta^{\mathbb{N}}$ be c-automatic. We show that σ is d-automatic by an application of Lemma 26. Let $a \in \Delta$. Then the fiber $\{c(n) \mid \sigma(n) = a\}$ is relatively regular in $c[\mathbb{N}]$ by Lemma 26. By Lemma 6 we obtain that the fiber $\{d(n) \mid \sigma(n) = a\}$ is regular (and consequently relatively regular in $d[\mathbb{N}] = \Gamma^*$). Hence, by Lemma 26, σ is d-automatic. □

The following proposition shows that the implication in Corollary 27 cannot be strengthened to equivalence of c- and d-automaticity: not for all injective c there is a bijective d so that c-automaticity and d-automaticity coincide.

Proposition 28. *Define the representation $c : \mathbb{N} \to \{0, 1\}^*$ by $c(n) = 0^{n!}$. Then we have:*

(i) For every sequence $\sigma \in \Delta^{\mathbb{N}}$, σ is c-automatic if and only if σ is ultimately constant.

(ii) For every bijection $d : \mathbb{N} \to \{0, 1\}^*$, there is a d-automatic sequence that is not ultimately constant.

Proof. We first prove the two implications of (i).

(⇒) Let $\sigma \in \Delta^{\mathbb{N}}$ be c-automatic. That is, for some automaton A, $\sigma(n) = A(c(n))$ for all $n \in \mathbb{N}$. As there are finitely many states in A, there exists $n_0, \ell \in \mathbb{N}$ such that $\ell > 0$ and $\delta(q_0, 0^n) = \delta(q_0, 0^{n+\ell})$ for all $n \geq n_0$, where δ is the transition function of A and q_0 its starting state. Let $m_0 \in \mathbb{N}$ be the smallest integer such that $m_0! \geq n_0 + \ell$. Then we have $\delta(q_0, 0^{m!}) = \delta(q_0, 0^{m_0!})$ for all $m \geq m_0$. The reason is that $m_0! \geq n_0$ and, for all $m \geq m_0$, $m!$ is a multiple of ℓ (so that $m! = m_0! + k\ell$ for some $k \in \mathbb{N}$).

(⇐) Let $\sigma \in \Delta^{\mathbb{N}}$ be ultimately constant, that is, there exists $n_0 \in \mathbb{N}$ such that $\sigma(n) = \sigma(n_0)$ for all $n \geq n_0$. Let $m = n_0!$. We define an automaton A with states $q_0, q_1, \ldots, q_m$, and transition function δ defined by $\delta(q_i, 0) = q_{i+1}$ for all $i \in \{0, 1, \ldots, m-1\}$ and $\delta(q_m, 0) = q_m$. For the output of q_i ($0 \leq i \leq m$) we take $\sigma(j)$ if $i = j!$. The output of the other states is irrelevant. Clearly we now have $\sigma(n) = A(c(n))$ for all $n \in \mathbb{N}$, and so σ is c-automatic.

For (ii), let $d : \mathbb{N} \to \{0\}^*$ be a bijective encoding. Let A be an automaton such that $A(0^{2n}) = 0$ and $A(0^{2n+1}) = 1$ for all $n \in \mathbb{N}$, and define $\sigma \in \{0, 1\}^{\mathbb{N}}$ by $\sigma(n) = A(d(n))$ for $n \in \mathbb{N}$. Note that σ is d-automatic by definition. As d is bijective, there are infinitely many $m \in \mathbb{N}$ such that $d(m)$ is of the form 0^{2n}, and there are infinitely many $m \in \mathbb{N}$ such that $d(m)$ is of the form 0^{2n+1}. Hence σ is not ultimately constant. □

The following corollaries are reformulations of our main result, Theorem 8, for recognizability and, more generally, automaticity, respectively.

Corollary 29. *For every countable class $\mathcal{C}$ of sets of natural numbers there is a bijective function $r : \mathbb{N} \to \Sigma^*$ such that every $S \in \mathcal{C}$ is r-recognizable (i.e., there is a finite automaton deciding membership $n \in S$ on the input of $r(n)$).*

Corollary 30. *Let Γ, Δ be finite alphabets with $|\Gamma| \geq 2$. For every countable class $\mathcal{S} \subseteq \Delta^{\mathbb{N}}$ of infinite sequences over Δ there exists a bijective number representation $d : \mathbb{N} \to \Gamma^*$ such that every $\sigma \in \mathcal{S}$ is d-automatic.*

We briefly comment on the relation with Cobham's Theorem [6], which states that a sequence is ultimately periodic if and only if it is both k- and ℓ-automatic for multiplicatively independent $k, \ell \in \mathbb{N}$. For example, if a sequence is both 2- and 3-automatic, then it is ultimately periodic. As Cobham's Theorem only pertains to standard number representations, it does not contradict the following easy consequence of Corollary 30: there exists a bijective ternary number representation $d : \mathbb{N} \to \{0, 1, 2\}^*$ such that every 2-automatic sequence is d-automatic.

VI. Conclusion and Further Questions

Our main result, Theorem 8, states that for every countable class $\mathcal{L} \subseteq \wp(\Sigma^*)$ of languages over a finite alphabet Σ, and

for every alphabet Γ with more than two symbols, there exists a bijective encoding $f : \Sigma^* \to \Gamma^*$ such that for every language $L \in \mathcal{L}$ its image $f[L]$ under f is regular.

Furthermore we have shown that this result has a number of noteworthy consequences in language theory for regularity preserving functions, in computability theory for a concept for comparing the power of models of computation, and in the theory of automatic sequences for a generalization of this concept with respect to arbitrary number representations:

(A) There exists a computable bijective function $f : \Sigma^* \to \Gamma^*$ such that the image function $f[\_]$ of f is regularity preserving, but the preimage function $f^{-1}[\_]$ is not (Theorem 9).

(B) In the sense of [5], finite-state automata are as powerful as any countable decision model (e.g., Turing-machine deciders) (Proposition 18). This even holds for the strongest notion of comparison in [5], namely that with respect to bijective encodings (Corollary 20). Similar counterintuitive consequences also affect computational models beyond decision models.

(C) For every countable class $\mathcal{C}$ of sets of natural numbers there is a bijective number representation $r : \mathbb{N} \to \Sigma^*$ such that every set $S \in \mathcal{C}$ is r-recognizable (i.e., there is a finite automaton deciding membership $n \in S$ on the input of $r(n)$) (Corollary 29).
More generally, for every countable class $\mathcal{S} \subseteq \Delta^{\mathbb{N}}$ of infinite sequences over a finite alphabet Δ, there exists a bijective number representation $d : \mathbb{N} \to \Gamma^*$ such that every $\sigma \in \mathcal{S}$ is d-automatic (Corollary 30).

These results also answer the questions in Section I concerning the hierarchy of number representations. From (A) it follows that the hierarchy is proper: there are bijective representations r_1 and r_2 such that r_1 subsumes r_2, but not vice versa. From (C) it follows that every countable class $\mathcal{C} \subseteq \wp(\mathbb{N})$ of languages is contained in the countable class of all r-recognizable languages, for some representation r. Moreover, (C) implies that every injective number representation is subsumed by a bijective number representation, and that no representation subsumes all others, since the class of r-recognizable sets of natural numbers is always countable.

We conclude with two questions:

- How far can computable bijective $f : \Sigma^* \to \Gamma^*$ extend the class of recognizable languages, that is, what classes $\mathcal{L}_f = \{L \subseteq \Sigma^* \mid f[L] \text{ is a regular language}\}$ can we obtain for a computable bijective f? For example, is there a computable (bijective) encoding that makes precisely all context-free languages recognizable?
- Rigo [20] describes a class of number representations that characterizes the morphic sequences. Our results entail the existence of a bijective representation $r : \mathbb{N} \to \Sigma^*$ such that every morphic sequence is r-automatic. Is there a computable bijective representation r such that *precisely* the morphic sequences are r-automatic?

ACKNOWLEDGMENT

We want to thank Nachum Dershowitz and Udi Boker for their remarks and a discussion about our results in Section IV, as well as for several pointers to specific parts of their papers.

REFERENCES

[1] J.-P. Allouche and J. Shallit. *Automatic Sequences: Theory, Applications, Generalizations*. Cambridge University Press, New York, 2003.

[2] J. Berstel, L. Boasson, O. Carton, B. Petazzoni, and J.-É. Pin. Operations preserving regular languages. *Theoretical Computer Science*, 354(3):405–420, 2006.

[3] U. Boker. Comparing Computational Power. Master's thesis, Tel Aviv University, 2004.

[4] U. Boker. *The Influence of Domain Interpretations on Computational Models*. PhD thesis, Tel Aviv University, 2008.

[5] U. Boker and N. Dershowitz. Comparing Computational Power. *Logic Journal of the IGPL*, 14(5):633–647, 2006.

[6] A. Cobham. On the Base-Dependence of Sets of Numbers Recognizable by Finite Automata. *Mathematical Systems Theory*, 3(2):186–192, 1969.

[7] J. Endrullis, C. Grabmayer, and D. Hendriks. Mix-Automatic Sequences. In *Proc. of the 7th International Conference on Language and Automata Theory and Applications (LATA 2013)*, number 7810 in LNCS, 2013.

[8] J. Engelfriet and H. J. Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *Transactions of the American Mathematical Society*, 2(2):216–254, 2001.

[9] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 2nd edition, 2000.

[10] S. R. Kosaraju. Finite State Automata with Markers. In *Proc. 4th Annual Princeton Conference on Information Sciences and Systems*. Princeton, 1970.

[11] S. R. Kosaraju. Regularity preserving functions. *SIGACT News*, 6(2):16–17, 1974.

[12] D. Kozen. On regularity-preserving functions. *Bulletin of the European Association for Theoretical Computer Science*, pages 131–138, 1996.

[13] P. Květoň and V. Koubek. Functions preserving classes of languages. In *Proc. Conf. on Developments in Language Theory*, pages 81–102. World Scientific, 1999.

[14] A. B. Matos. Regularity-preserving letter selections. DCC-FCUP Interal Report.

[15] J.-É. Pin. Profinite Methods in Automata Theory. In *Proc. of the 26th Symposium on Theoretical Aspects of Computer Science (STACS 2009)*, pages 31–50. IBFI Schloss Dagstuhl, 2009.

[16] J.-É. Pin and J. Sakarovitch. Some operations and transductions that preserve rationality. In *Proc. 6th GI-Conference Theoretical Computer Science*, volume 145 of *Lecture Notes in Computer Science*, pages 277–288. Springer, 1983.

[17] J.-É. Pin and P. V. Silva. A topological approach to transductions. *Theoretical Computer Science*, 340(2):443–456, 2005.

[18] M. O. Rabin. Computable algebra, general theory and theory of computable fields. *Transactions of the American Mathematical Society*, 95(2):341–360, 1960.

[19] H. G. Rice. Recursive and recursively enumerable orders. *Transactions of the American Mathematical Society*, 83(2):277–300, 1956.

[20] M. Rigo. Generalization of automatic sequences for numeration systems on a regular language. *Theoretical Computer Science*, 244(1-2):271–281, 2000.

[21] H. Rogers. *Theory of Recursive Functions and Effective Computability*. MacGraw–Hill, 1967.

[22] J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[23] J. I. Seiferas. A note on prefixes of regular languages. *SIGACT News*, 6(1):25–29, 1974.

[24] J. I. Seiferas and R. McNaughton. Regularity-preserving relations. *Theoretical Computer Science*, 2(2):147–154, 1976.

[25] J. Shallit. A Generalization of Automatic Sequences. In *6th Symposium on Theoretical Aspects of Computer Science (STACS 1989)*, volume 349 of *LNCS*, pages 156–167. Springer, 1989.

[26] S. Shapiro. Acceptable notation. *Notre Dame Journal of Formal Logic*, 23(1):14–20, 1982.

[27] R. E. Stearns and J. Hartmanis. Regularity preserving modifications of regular expressions. *Information and Control*, 6(1):55–69, 1963.