# Injection structures specified by finite state transducers

Sam Buss[1], Douglas Cenzer[2], Mia Minnes[3] and Jeffrey B. Remmel[1]

[1] Department of Mathematics, University of California-San Diego, La Jolla, CA 92093-0112
*sbuss@ucsd.edu*, *jremmel@ucsd.edu*
[2] Department of Mathematics, University of Florida, Gainesville, FL 32611 *cenzer@math.ufl.edu*,
[3] Department of Computer Science and Engineering, University of California-San Diego, La Jolla, CA 92093-0404
*minnes@eng.ucsd.edu*

**Abstract.** An injection structure $\mathcal{A} = (A, f)$ is a set $A$ together with a one-place one-to-one function $f$. $\mathcal{A}$ is an FST injection structure if $A$ is a regular set, that is, the set of words accepted by some finite automaton, and $f$ is realized by a finite-state transducer. We initiate the study of FST injection structures. We show that the model checking problem for FST injection structures is undecidable which contrasts with the fact that the model checking problem for automatic relational structures is decidable. We also explore which isomorphisms types of injection structures can be realized by FST injections. For example, we completely characterize the isomorphism types that can be realized by FST injection structures over a unary alphabet. We show that any FST injection structure is isomorphic to an FST injection structure over a binary alphabet. We also prove a number of positive and negative results about the possible isomorphism types of FST injection structures over an arbitrary alphabet.
**Keywords:** computability theory, injection structures, automatic structures, finite state automata, finite state transducers.

## 1 Introduction and preliminaries

The main goal of this paper is to initiate the study of FST injection structures. Throughout this paper, we will restrict our attention to countable structures for computable languages. There has been considerable work on automatic structures for languages that contain only relation symbols. A structure, $\mathcal{A} = (A; R_0, \ldots, R_m)$, is **automatic** if its domain $A$ and all its basic relations $R_0, \ldots, R_m$ are recognized by finite automata. Independently, Hodgson [8] and later Khoussainov and Nerode [9] proved that for any given automatic structure there is an algorithm that solves the model checking problem for the first-order logic in the language of the structure. In particular, the first-order theory of the structure is decidable. In fact an even stronger result is true. We denote by $\exists^\infty$ the quantifier "there exists infinitely many" and $\exists^{(n,m)}$ the quantifier "there are $m$ many mod $n$." Then we let $FO + \exists^\infty + \exists^{(n,m)}$ denote first order logic extended with these quantifiers. Then Khoussainov, Rubin, and Stephan [13] proved the following.

**Theorem 1.** *There is an algorithm that given an automatic structure $\mathcal{A}$ and a $(FO + \exists^\infty + \exists^{(n,m)})$-formula $\phi(x_1, \ldots, x_n)$ with parameters from $\mathcal{A}$ produces an automaton recognizing those tuples $\langle a_1, \ldots, a_n \rangle$ that make the formula true in $\mathcal{A}$.*

It follows that the $(FO + \exists^\infty + \exists^{(n,m)})$-theory of any automatic structure is decidable.

Blumensath and Grädel proved a logical characterization theorem stating that automatic structures are exactly those definable in the fragment of arithmetic $(\omega; +, |_2, \leq, 0)$, where $+$ and $\leq$ have their usual meanings and $|_2$ is a weak divisibility predicate for which $x|_2 y$ if and only if $x$ is a power of 2 and divides $y$ (see [4]). In addition, for some classes of automatic structures, there are characterization theorems that have direct algorithmic implications. For example, in [7], Delhommé proved that automatic well-ordered sets all have order type strictly less than $\omega^\omega$. Using this characterization, [11] gives an algorithm which decides

* Buss was partially supported by NSF grants DMS-1101228 and CCF-1213151. Cenzer was partially supported by the NSF grant DMS-1101123. Minnes was partially supported by the NSF grant DMS-1060351.

the isomorphism problem for automatic well-ordered sets. Another characterization theorem of this ilk is that automatic Boolean algebras are exactly those that are finite products of the Boolean algebra of finite and co-finite subsets of $\omega$ [12]. Again, this result can be used to show that the isomorphism problem for automatic Boolean algebras is decidable.

Another body of work is devoted to the interaction between the representation of an automatic structure and the complexity of the model checking problem. In particular, every automatic structure has a presentation over a binary alphabet but there are automatic structures which do not have presentations over a unary (one letter) alphabet. Moreover, for automatic structures with unary presentations, the monadic second-order theory (not just the first-order theory) is decidable. There are also feasible time bounds on deciding the first-order theories of automatic structures over the unary alphabet ([3], [14]).

In this paper, we restrict our attention to *injection structures*. We begin by fixing notation and terminology. We let $\mathbb{N} = \{0, 1, 2, \ldots\}$ denote the natural numbers and $\mathbb{Z} = \{0, \pm 1, \pm 2, \ldots\}$ denote the integers. We let $\omega$ denote the order type of $\mathbb{N}$ under the usual ordering and $Z$ denote the order type of $\mathbb{Z}$ under the usual ordering. For any finite nonempty set $\Sigma$, we let $\Sigma^*$ denote the set of all words over the alphabet $\Sigma$. We let $\epsilon$ denote the empty word and for any word $w = w_1 \ldots w_n$, we let $|w| = n$ denote the length of $w$. We let $\Sigma^+ = \Sigma^* - \{\epsilon\}$ and for $n \in \mathbb{N}$, $\Sigma^{\leq n} = \{w \in \Sigma^* : |w| \leq n\}$. An injection is a one-place one-to-one function and an injection structure $\mathcal{A} = (A, f)$ consists of a set $A$ and an injection $f : A \to A$. $\mathcal{A}$ is a *permutation structure* if $f$ is a permutation of $A$. Given $a \in A$, the orbit $\mathcal{O}_f(a)$ of $a$ under $f$ is

$$\mathcal{O}_f(a) = \{b \in A : (\exists n \in \mathbb{N})(f^n(a) = b \ \vee \ f^n(b) = a)\}.$$

The order $|a|_f$ of $a$ under $f$ is $card(\mathcal{O}_f(a))$. We define the *character* $\chi(\mathcal{A})$ of an injection structure $\mathcal{A} = (A, f)$ by

$$\chi(\mathcal{A}) = \{(n, k) : \mathcal{A} \text{ has at least } n \text{ orbits of size } k\}$$

and its *type* $T(\mathcal{A})$ by

$$T(\mathcal{A}) = \{(n, k) : k \in \mathbb{N} - \{0\} \ \& \ \mathcal{A} \text{ has exactly } n \text{ orbits of order type } k\}.$$

Infinite orbits of injection structures $(A, f)$ are either of type $Z$, in which every element is in the range of $f$, or of type $\omega$, which have the form $\mathcal{O}_f(a) = \{f^n(a) : n \in \mathbb{N}\}$ for some $a \notin ran(f)$. It is easy to see that the type of an injection structure plus the finite information about the number of $Z$-orbits and $\omega$-orbits completely characterizes its isomorphism type.

The algorithmic properties of injection structures were first studied in [5] and [6]. Recall that if $\mathcal{A}$ is a structure with universe $A$ for a language $\mathcal{L}$, then $\mathcal{L}^A$ is the language obtained by expanding $\mathcal{L}$ by constants for all elements of $A$. The *atomic diagram* of $\mathcal{A}$ is the set of all quantifier-free sentences of $\mathcal{L}^A$ true in $\mathcal{A}$. A structure $\mathcal{A}$ is *computable* if its atomic diagram is computable. We call two structures *computably isomorphic* if there is a computable function that is an isomorphism between them. A computable structure $\mathcal{A}$ is *relatively computably isomorphic* to a possibly noncomputable structure $\mathcal{B}$ if there is an isomorphism between them that is computable in the atomic diagram of $\mathcal{B}$. A computable structure $\mathcal{A}$ is *computably categorical* if every computable structure that is isomorphic to $\mathcal{A}$ is computably isomorphic to $\mathcal{A}$. A computable structure $\mathcal{A}$ is *relatively computably categorical* if every structure that is isomorphic to $\mathcal{A}$ is relatively computably isomorphic to $\mathcal{A}$.

In [6], Cenzer, Harizanov, and Remmel characterized computably categorical injection structures, and showed that they are all relatively computably categorical. Among other things, they proved that a computable injection structure $\mathcal{A}$ is computably categorical if and only if it has finitely many infinite orbits. They also characterized $\Delta^0_2$-categorical injection structures as those with finitely many orbits of type $\omega$, or with finitely many orbits of type $\mathbb{Z}$. They showed that they coincide with the relatively $\Delta^0_2$-categorical structures. Finally, they proved that every computable injection structure is relatively $\Delta^0_3$-categorical. They also showed that the character of any computable injection structure is a c.e. set and that any c.e. character may be realized by a computable injection structure.

A deterministic finite automaton (DFA) is specified by the tuple $(Q, \iota, \Sigma, \delta, F)$ where $Q$ is the finite set of states, $\iota$ is the initial state, $\Sigma$ is the input alphabet, $\delta : Q \times \Sigma \to Q$ is the (possibly partial) transition

function, and $F \subseteq Q$ is the set of final, or accepting states. A DFA $M$ *accepts* a string $w$ if the last input of $w$ causes $M$ to halt in one of the accepting. The set $L(M) \subseteq \Sigma^*$ of strings accepted by $M$ is the language *recognized* by $M$. A language $L \subseteq \Sigma^*$ is said to be *regular* if it is accepted by some DFA. To recognize a relation $R \subseteq \Sigma^* \times \Sigma^*$, there are two possibilities. We can have a two-tape synchronous DFA, where the transition function $\delta : Q \times \Sigma \cup \{\square\} \times \Sigma \cup \{\square\} \to Q$ and $\square$ denotes a blank square. The blank square is needed in the case that one input is longer than the other. Then $M$ halts after reaching the end of the longer word. The other approach is to simulate this with a standard one tape DFA over the language $\Sigma \cup \{\square\} \times \Sigma \cup \{\square\}$. Automatic relations and structures have been studied by Khoussainov, Minnes, Nies, Rubin, Stephan and others [9, 12, 11, 14].

A deterministic finite-state transducer (DFST) is specified by the tuple $(Q, \iota, \Sigma, \Gamma, \delta, \tau)$ where $Q$ is the finite set of states, $\iota$ is the initial state, $\Sigma$ is the input alphabet, $\Gamma$ is the output alphabet, $\delta : Q \times \Sigma \to Q$ is the (possibly partial) transition function, and $\tau : Q \times \Sigma \to \Gamma^*$ is the (possibly partial) output function. A DFST $M$ naturally defines a (possibly partial) function, $f_M : \Sigma^* \to \Gamma^*$. We say that the DFST $M$ *realizes* a function $f$ on a set $U \subseteq \Sigma^*$ if $f_M \restriction U = f$.

**Definition 2** *An injection structure $\mathcal{A} = (A, f)$ consists of a set $A$ together with a one-to-one mapping $f : A \to A$. $\mathcal{A}$ is an* FST injection structure *if $A$ is a regular set of words in $\Sigma^*$ (for some alphabet $\Sigma$), that is, the set of words accepted by some finite automaton, and $f$ is realized by a DFST. By convention, we will assume that if $\epsilon \in A$, then $f(\epsilon) = \epsilon$.*

Next we shall use the notion of accepting deterministic finite-state transducers to give an equivalent definition of FST injection structures.

**Definition 3** *An* accepting deterministic finite-state transducer *(AFST) is specified by the tuple $(Q, \iota, \Sigma, \Gamma, \delta, \tau, F)$ where $F$ is a subset of states designated as accepting states. The function defined by this AFST has as its domain the set of words in $\Sigma^*$ such that, when processing these words, the AFST ends in a state in $F$.*

Since the isomorphism class of an injection structure is determined by its type, and the number of orbits of type $\omega$ and $\mathbb{Z}$, we seek to characterize those types which have presentations using transducers. We prove that the two kinds of transducers defined above realize the same isomorphism types of injection structures.

**Theorem 4.** *An isomorphism type of an injection structure is realized by an FST injection structure if and only if that structure can be realized by an AFST.*

*Proof.* An AFST $(Q, \iota, \Sigma, \Gamma, \delta, \tau, F)$ is naturally associated with a DFST $M$ (by omitting $F$). The automaton for the domain of the function is then the DFA $(Q, \iota, \Sigma, \delta, F)$, recognizing the set $U$. Then $(U, f_M)$ is the original FST injection structure.

Conversely, suppose the DFST $(Q, \iota, \Sigma, \delta, F)$ and the DFA $(Q', \iota', \Sigma, \delta', F')$ are given. Form the product automaton of the two (analogously to the classical automata constructions). The resulting DFST can be augmented with accepting states which are all those whose second component is an accepting state in $F$.

Let $\mathcal{A} = (A, f)$ and $\mathcal{B} = (B, g)$ be injection structures. We let $\mathcal{A} \otimes \mathcal{B} = (A \times B, f \times g)$ where for any $a \in A$ and $b \in B$, $f \times g(a, b) = (f(a), g(b))$. We let $\mathcal{A} \oplus \mathcal{B} = (\{x_1 a : a \in A\} \cup \{x_2 b : b \in b\}, f \oplus g)$ where $x_1$ and $x_2$ are new letters outside of $A \cup B$ and $f \oplus g(x_1 a) = x_1 f(a)$ and $f \oplus g(x_2 b) = x_2 g(b)$ for all $a \in A$ and $b \in B$. It is easy to see that if $\mathcal{A} = (A, f)$ and $\mathcal{B} = (B, g)$ are FST injection structures, then $\mathcal{A} \otimes \mathcal{B}$ and $\mathcal{A} \oplus \mathcal{B}$ are FST injection structures.

The main goal of this paper is explore the difference between computable injection structures and FST injection structures. Our first result is that the model checking problem for FST injection structures is not decidable. This contrasts with the fact that the model checking problem for automatic relational structures is decidable. We shall also study the possible types of FST injection structures. Whether a given type may be realized by an FST injection structure $(A, f)$ depends on a number of factors such as (i) the underlying alphabet $\Sigma$, (ii) the number of states in of the underlying transducer $T = (Q, \iota, \Sigma, \Gamma, \delta, \tau)$ for $f$, and (iii)

the nature of the output function $\tau$ of the transducer $T$. For example, we say that a transducer $T = (Q, \iota, \Sigma, \Gamma, \delta, \tau)$ *has $\epsilon$-outputs* if there is state $q \in Q$ and $a \in \Sigma$ such that $\tau(q, a) = \epsilon$. We say $T$ is *length preserving* if $\tau(q, a) \in \Gamma$ for all $q \in Q$ and $a \in \Sigma$. Thus when a length preserving transducer reads a symbol $a \in \Sigma$ in any state $q$, it outputs a single letter in $\Gamma$. We say that $(A, f)$ has full domain if $\Sigma^+ \subseteq A$.

The outline of this paper is as follows. In Section 2, we will show that the model checking problem for FST injection structures is undecidable; this is done by coding computations of reversible Turing machines into FST injection structures. In Section 3, we will show that a large class of types can be realized by FST injection structures. For example, we will construct FST injection structures as summarized in the following table. Note that the columns describing the FST realization list the parameters for the examples we build; they may not be optimal.

| **Type** | **Alphabet** | **$\epsilon$-outputs?** | **States** | **Full domain?** | **Lemma** |
|---|---|---|---|---|---|
| $1_m$ | $\{1\}$ | N | 1 | N | 1 |
| $1_\infty$ | $\{1\}$ | N | 1 | Y | 1 |
| $1_m, \omega_n$ | $\{1\}$ | N | $m+1$ | Y | 2 |
| $1_m, \omega_\infty$ | $\{1\}$ | N | $m+1$ | Y | 3 |
| $k_1$ | $\{0,1\}$ | Y | $2^{k+1}-1$ | N | 4 |
| $k_\infty$ | $\{0,1\}$ | Y | $2^{k+1}-1$ | N | 4 |
| $\zeta_1$ | $\{0,1\}$ | Y | 3 | N | 5 |
| $\zeta_\infty$ | $\{0,1\}$ | Y | 3 | N | 5 |
| $\ell_\infty$ | $\{0,1,\ldots,\ell-1\}$ | N | 1 | N | 6 |
| $\ell_1^i$ | $\{0,1,\ldots,\ell-1\}$ | N | 2 | Y | 7 |
| $\ell_\infty^i$ | $\{0,1,\ldots,\ell-1\}$ | N | 2 | Y | 7 |

Here for any positive integers $k$ and $m$, we write $k_m$ to denote the character which has $m$ orbits of size $k$ and we write $k_\infty$ to denote the character with infinitely orbits of size $k$. Similarly, we write $\omega_m$ to denote the character that has $m$ $\omega$ orbits, $\omega_\infty$ to denote the character with infinitely many $\omega$ orbits, $\zeta_m$ to denote the character which has $m$ $Z$ orbits, and $\zeta_\infty$ to denote the character with infinitely many $Z$ orbits. Then, for example, $1_m, \omega_n$ in line 3 of the table means an injection structure with $m$ orbits of size 1 and $n$ orbits of type $\omega$, $\zeta_1$ on line 7 means an injection structure with exactly one orbit which is of type $Z$, and $\zeta_\infty$ on line 8 means an injection structure with infinitely many orbits of type $Z$. We write $\ell_1^i$ for the type of injection structures with exactly one orbit of size $\ell^i$ for each $i \geq 1$. Similarly, $\ell_\infty^i$ is the type with infinitely many orbits of size $\ell^i$ for each $i \geq 1$. Of course, the fact that FST injection structures are closed under $\otimes$ and $\oplus$ means that we can use these simple types to obtain many other types which are realized by FST injection structures.

In Section 4, we shall characterize all types realized by FST injection structures over a unary alphabet as the ones already listed in the table above. In particular, it will follow that FST injection structures over a unary alphabet are not closed under disjoint union. However, we will show that any type realized by an FST injection structure can be realized by an FST injection structure over a two-letter alphabet. In Section 5, we will prove a restriction on the types that can be realized by FST injection structures which are length preserving and have full domain. In Section 6, we will address the question of when FST injection structures have automatic graphs. Finally, in Section 7, we will state some open problems.

## 2 Undecidability

In this section, we prove that the model checking problem for injection structures is undecidable. We fix the first-order language $L_f$ to have a single unary function symbol $f$. The model checking problem for $L_f$ is the problem of, given a presentation of an FST structure $\mathcal{A}$ over $L_f$ and given a first-order formula $\varphi$, deciding whether $\mathcal{A} \vDash \varphi$.

**Theorem 5.** *The model checking problem for FST injection structures is undecidable for the formula* $\exists x(f(x) = x)$.

*Proof.* We use Bennett's reversible Turing machines. Theorem 1 of [1] shows that any Turing machine can be simulated by a 3-tape Turing machine $N$ which is reversible; namely, every configuration of $N$ has at most one predecessor configuration. As a consequence, the halting problem for 3-tape reversible Turing machines is undecidable. Such a machine $N$ can be simulated by a single tape machine $M$ by letting the tape contents of $M$ encode a configuration of $N$. For this, $M$'s tape contains three "tracks", one for each tape of $N$. Each track holds a symbol from one of $N$'s tapes plus a flag indicating whether the corresponding tape head is positioned over that symbol. The state of $M$ incorporates the information of the current state of $N$ and the current symbols under $N$'s three tape heads. This allows $M$ to simulate $N$ by scanning its entire tape repeatedly from left-to-right. As $M$ scans from left-to-right, it updates the contents its tapes to reflect the change in the configuration caused by a single step of $N$. The scan from right-to-left merely returns to the left end and prepares for simulating the next step of $N$.

A configuration of $M$ can be encoded as a string $C$ over the alphabet $L_M$ which contains the tape symbols of $M$ and symbols for each state of $M$. In $C$, the symbol for $M$'s state immediately follows the symbol under the tape head. Such a string $C$ is a called a *valid configuration* provided that the tape contents as encoded by $C$ satisfy the following four conditions:

**(a)** for each of $N$'s tapes, there is exactly one symbol in $C$ indicating the presence of $N$'s tape head on that tape,
**(b)** there is one symbol in $C$ encoding $M$'s tape head position,
**(c)** the state of $M$ as encoded in $C$ correctly agrees with the relative positions of $N$'s tape heads and the symbols currently being read by $N$, and
**(d)** $C$ does not have leading or trailing blank symbols.

We omit the straightforward details, but the valid configurations satisfy the following four properties.

**(A)** Every valid configuration $C$ has a valid successor configuration, $Nxt(C)$, representing the next (also valid) configuration of $M$. $Nxt(C)$ is defined even for halting configurations of $M$ (i.e., $M$ continues running after reaching a halting configuration). By reversibility, the $Nxt$ function is injective.
**(B)** Every valid configuration of $M$ has at most one valid predecessor configuration.
**(C)** The set of valid configurations is regular.
**(D)** There is a DFST which, when given valid configuration $C$ as input, outputs the next configuration $Nxt(C)$.

Without loss of generality, when (and if) $N$ halts it first erases its output and work tape — leaving the input tape with its original contents. We are interested only in running $N$ on the empty word as input. We let $C_{\text{init}}$ represent the initial configuration for $N$. We likewise let $C_{\text{final}}$ represent the accepting configuration of $M$ which results if $N$ halts after having the empty string as input. By the convention on how $N$ halts, $C_{\text{final}}$ is a known, fixed configuration.

We give a many-one reduction from the halting problem for a 3-tape reversible machine $N$ to the question of whether an FST injection structure $\mathcal{A}$ satisfies the formula $\exists x(f(x) = x)$. The structure $\mathcal{A}$ uses the alphabet $L_M \cup \{|, \$\}$ where "|" and "$" are new symbols. The domain of $\mathcal{A}$ is the set of words of the form

$$C_1|C_2|C_3|\cdots|C_k|\$^i||, \tag{1}$$

where each $C_i$ is a valid configuration, $k \geq 1$, and $\$^i$ denotes $i$ many \$ symbols with $i \geq 0$. The domain of $\mathcal{A}$ is regular since the set of valid configurations is regular. We define the injective function $f$ as follows. For any string $v$ of the form specified by (1), $f(v)$ is defined according the following four cases.

**(1)** If $i > 0$, then $f(v)$ is equal to $C_{\text{init}}|Nxt(C_1)|Nxt(C_2)|\cdots|Nxt(C_k)|\$^{i+1}||$. Note the additional \$ symbol.
**(2)** If $i = 0$ and $k > 0$ and both $Nxt(C_{k-1})$ and $C_k$ are equal to $C_{\text{final}}$, then $f(v)$ is equal to $C_{\text{init}}|Nxt(C_1)|Nxt(C_2)|\cdots|Nxt(C_{k-1})|||$.
**(3)** If $i = 0$ and $Nxt(C_k)$ is equal to $C_{\text{final}}$, then $f(v)$ is equal to $C_{\text{init}}|Nxt(C_1)|Nxt(C_2)|\cdots|Nxt(C_k)|\$||$. Note the added \$ symbol.
**(4)** Otherwise, $f(v)$ is equal to $C_{\text{init}}|Nxt(C_1)|Nxt(C_2)|\cdots|Nxt(C_k)|||$. Note there is no \$-symbol. In this case, $Nxt(C_k)$ is not equal to $C_{\text{final}}$ since otherwise case **(3)** would apply.

By inspection, if $v$ is in the domain of $\mathcal{A}$, then so is $f(v)$. It is straightforward to see that $f$ can be computed by a deterministic finite state transducer. The DFST starts out by writing "$C_{\text{init}}|$"; it then repeatedly reads a string a "$C_i|$" while writing "$C_{Nxt(C_i)}|$". The string "$C_{\text{final}}$" is a fixed string, so the DFST can detect when it has reached the end of $v$ and which of the cases **(1)**-**(4)** applies by looking forward only a constant number of symbols. (The purpose of the extra "||" at the end of $v$ to let the DFST detect when it has reached $C_k$ with $C_k$ equal to $C_{\text{final}}$ without having to read the last symbol of $v$.)

To see that $f$ is injective, fix a string $w$ in $\mathcal{A}$ of the form (1). We must show that $f(v) = w$ can hold for at most one $v$. If $w$ has the form (1) with $i > 0$, then any $f(v) = w$ must be defined according to case **(1)** or **(3)** depending on whether $i = 1$ or $i > 1$. In either case, the injectivity of the *Nxt* function ensures that $v$ is unique. Otherwise $i = 0$. If the $C_k$ in $w$ is not equal to $C_{\text{final}}$, then case **(2)** does not apply and so case **(4)** was used to define $f(v)$. Thus the uniqueness of $v$ again follows by the injectivity of the *Nxt*. Finally, if $i = 0$ and the $C_k$ in $w$ does equal $C_{\text{final}}$, then $f(v) = w$ is defined by case **(2)**, and once again, injectivity of *Nxt* implies the uniqueness of $v$.

To finish the proof of Theorem 5, we show that $\mathcal{A} \vDash \exists x(f(x) = x)$ holds if and only if $N$ halts when started on the empty string. When calculating $f(v)$, the only way to have $f(v) = v$ is for case **(2)** to apply, since the other three cases add a symbol "|", and maybe a "$". If case **(2)** gives $f(v) = v$, then we have $C_1 = C_{\text{init}}$, and $C_j = Nxt(C_{j-1})$ for all $j$, and $C_k = C_{\text{final}}$. Thus the configurations in $v$ are a halting computation of $M$. Conversely, a halting computation of $M$ gives $v$ such that $f(v) = v$. □

It remains open whether there is a FST injection structure with undecidable first-order theory. However, we can build FST injection structures with some level of undecidability. Let $Fin(M)$ denote $\{x :$ $x$ belongs to a finite orbit of $M\}$.

**Theorem 6.** *There is a FST $N$ such that $Fin(N)$ is c.e. complete.*

*Proof.* Let $W$ be a complete c.e. set and let $A$ be the set $\{0^n : n \in W\}$. Then there is a reversible Turing machine $M$ such that for any string $\sigma \in \{0,1\}^*$, $M$ converges on $\sigma$ if and only $\sigma \in A$. We assume that $M$ is realized by a single tape Turing machine as described in Theorem 5. We can modify $M$ as follows to produce a Turing machine $T$. First, since $M$ is reversible, it is easy to see that there is Turing machine $\overline{M}$ that operates as follows. For any state $q$ of $M$, $\overline{M}$ has a state $\overline{q}$ where we assume that states of $M$ and $\overline{M}$ are disjoint. We assume that $M$ has a single start state $s$ and single final state $f$. Then the start state of $\overline{M}$ is $\overline{f}$ and the final state of $\overline{M}$ is $\overline{s}$. For any configuration $C$, let $\overline{C}$ denote the configuration obtained from $C$ by replacing the state $q$ of the configuration by $\overline{q}$. For any input $\sigma \in \{0,1\}$, suppose that $M$ starts on an initial configuration $I_\sigma$ for input $\sigma$ and proceeds through a sequence of configurations $C_1, \ldots, C_r$ and then ends with a configuration $F$ whose state is the final state $f$. Then the transition table of $\overline{M}$ is defined so that if $\overline{M}$ starts on configuration $\overline{F}$, it will proceed through the sequence of configurations $\overline{C_r}, \ldots, \overline{C_1}$ and then end in configuration $\overline{I_\sigma}$. Our Turing machine $T$ will operate as follows. It has a new start state $s_0$ and its final state is $\overline{s}$. Given an initial configuration $I_\sigma$ for input $\sigma$, it first goes to state $s$ and does not move. Then it uses the states and transition function of $M$ to proceed through the sequence of states $I_\sigma, C_1, \ldots, C_r, F$. On seeing the final state $f$ of $M$, $T$ goes to state $\overline{f}$ and does not move. This will result in configuration $\overline{F}$. Then $T$ uses the states and the transition function of $\overline{M}$ to go through the sequence of configurations $\overline{F}, \overline{C_r}, \ldots, \overline{C_1}, \overline{I_\sigma}$. Finally on seeing state $\overline{s}$, it goes to state $s_0$ and does not move so that we will end the initial configuration for input $\sigma$. Thus on a computation of $M$ with input $\sigma$ that converges, $T$ will produce a cycle of configurations.

It is easy to see that configurations of $T$ will be a regular language $L$ and we can define an FST $G$ so that on any configuration of $T$, $G$ outputs the result of applying $T$ to configuration $C$. It follows that the only cycles of $(L, G)$ correspond to converging computations of $M$. Thus if we could decide whether a configuration is part of cycle, then we can decide whether $M$ started on input $0^n$ converges. It follows that $Fin(L, G)$ is c.e. complete.

## 3 Realizing injection structures with simple types

**Lemma 1.** *Let $m \in \mathbb{N}$. There is a FST injection structure (over a unary alphabet) realizing each of the types $\{1_m\}$ and $\{1_\infty\}$.*

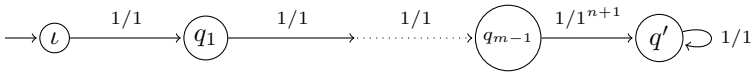
*Proof.* The identity function on $\{1^0, \ldots, 1^{m-1}\}$ and $1^*$ (respectively) has this type. Each of these domains is regular and the function can be realized by the one-state DFST $(\{\iota\}, \iota, \{1\}, \{1\}, \{(\iota, 1, \iota)\}, \{(\iota, 1, 1)\})$.


**Lemma 2.** *Let $m, n \in \mathbb{N}$. There is an FST injection structure (over a unary alphabet) realizing the type $\{1_m, \omega_n\}$.*

*Proof.* The function

$$1^i \mapsto \begin{cases} 1^i & \text{if } 0 \le i < m \\ 1^{i+n} & \text{if } i \ge m \end{cases}$$

has this type. In particular, the $m$-many 1-cycles have elements in the set $\{\epsilon, 1, \ldots, 1^{m-1}\}$; the $n$-many $\omega$-cycles can be described using arithmetic progressions with bases $m, \ldots, m+n-1$ and successive difference $n$. This function can be realized by the $(m+1)$-state DFST $(\{\iota, q_1, \ldots, q_{m-1}, q'\}, \iota, \{1\}, \{1\}, \delta, \tau)$ where $\delta(\iota, 1) = q_1$, $\delta(q_i, 1) = q_{i+1}$ for $1 \le i < m-1$, $\delta(q_{m-1}, 1) = q'$, and $\delta(q', 1) = q'$. Also, $\tau(q, 1) = 1$ for $q = \iota$ or $q = q'$ or $q = q_i$ with $1 \le i < m-1$, and $\tau(q_{m-1}, 1) = 1^{n+1}$.

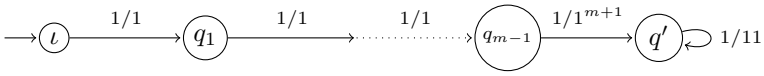



**Lemma 3.** *Let $m \in \mathbb{N}$. There is a FST injection structure (over a unary alphabet) realizing the type $\{1_m, \omega_\infty\}$.*

*Proof.* The function

$$1^i \mapsto \begin{cases} 1^i & \text{if } 0 \le i < m \\ 1^{2i} & \text{if } i \ge m. \end{cases}$$

has this type. In particular, the $m$-many 1-cycles are on the domain $\{\epsilon, 1, \ldots, 1^{m-1}\}$; the infinitely-many $\omega$-cycles can be described using arithmetic progressions each of whose base is equal to its successive difference. This function can be realized by the $(m+1)$-state DFST $(\{\iota, q_1, \ldots, q_{m-1}, q'\}, \iota, \{1\}, \{1\}, \delta, \tau)$ where $\delta(\iota, 1) = q_1$, $\delta(q_i, 1) = q_{i+1}$ for $1 \le i < m-1$, $\delta(q_{m-1}, 1) = q'$, and $\delta(q', 1) = q'$. Also, $\tau(q, 1) = 1$ for $q = \iota$ or $q = q_i$ with $1 \le i < m-1$, $\tau(q', 1) = 11$, and $\tau(q_{m-1}, 1) = 1^{m+1}$.




**Lemma 4.** *Let $k \in \mathbb{N}$ be greater than or equal to 2. The types $\{k_1\}$ and $\{k_\infty\}$ are realized by FST injection structures (with $\epsilon$-outputs) over the alphabet $\{0, 1\}$.*

*Proof.* Fix $k$ and let $\sigma^{(1)}, \ldots, \sigma^{(k)}$ be the first $k$ binary strings of length $k$ in lexicographic (dictionary) order. For example, if $k = 3$, then $\sigma^{(1)} = 000$, $\sigma^{(2)} = 001$ and $\sigma^{(3)} = 010$. Consider a transducer with $\epsilon$-outputs on $\{0, 1\}$ whose set of states is the set of all $\tau \in \{0, 1\}^{\le k}$. For each $q \in \{0, 1\}^{\le k}$, each $i \in \{0, 1\}$ the transition function and output functions are given by

$$\delta(q, i) = \begin{cases} qi & \text{if } |q| < k \\ q & \text{if } |q| = k. \end{cases} \qquad \tau(q, i) = \begin{cases} \epsilon & \text{if } |q| < k \text{ and } qi \ne \sigma^{(s)} \text{ for any } s \\ \sigma^{(s+1)} & \text{if } |q| < k \text{ and } qi = \sigma^{(s)} \text{ and } s < k \\ \sigma^{(1)} & \text{if } |q| < k \text{ and } qi = \sigma^{(k)} \\ i & \text{if } |q| = k. \end{cases}$$

For example, the transducer for $k = 3$ is given below.

It is then easy to see that if we restrict $A$ to be $\{\sigma^{(1)}, \ldots, \sigma^{(k)}\}$, then $(A, f)$ will consist of a single $k$ cycle and if we restrict $A$ to be all strings that extend one of $\sigma^{(1)}, \ldots, \sigma^{(k)}$ (a regular set), then $(A, f)$ will have infinitely many $k$ cycles.

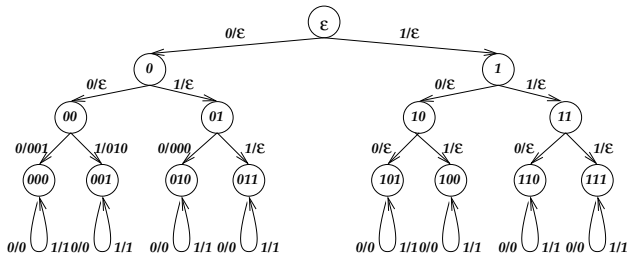



**Fig. 1.** The transducer constructed in Lemma 4 when $k = 3$.

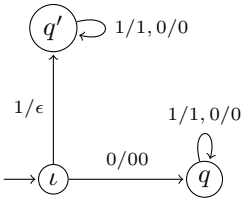
**Lemma 5.** *There are FST injection structures (with $\epsilon$-outputs) realizing the types $\{\zeta_1\}$ and $\{\zeta_\infty\}$ on domains which are regular subsets of $\{0,1\}^*$.*

*Proof.* Consider the function $1^i01^j \mapsto 1^{i-1}01^j$ for $i \geq 1$ and $0^i1^j \mapsto 0^{i+1}1^j$ for all $i$ and all $j \in \mathbb{N}$. For each fixed $m \in \mathbb{N}$, we get the chain

$$\cdots 1^i01^m \mapsto 1^{i-1}01^m \mapsto \cdots \mapsto 101^m \mapsto 01^m \mapsto 0^21^m \cdots$$

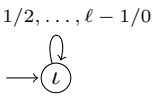
The regular set $1^*0 \cup 0^*$ corresponds to $m = 0$. The set $1^*01^* \cup 0^*1^*$ is the union of infinitely many $\zeta$-chains, one for each $m \in \mathbb{N}$.

Regardless of the domain, the function can be realized by the following 3-state DFST with $\epsilon$-outputs, $(\{\iota, q, q'\}, \iota, \{0,1\}, \{0,1\}, \delta, \tau)$ where $\delta(\iota, 1) = q'$, $\delta(\iota, 0) = q$, and $\delta(s, b) = s$ for $s \in \{q, q'\}$ and $b \in \{0,1\}$. Also, $\tau(\iota, 1) = \epsilon$, $\tau(\iota, 0) = 00$, and $\tau(s, b) = b$ for $s \in \{q, q'\}$ and $b \in \{0,1\}$.




**Lemma 6.** *Fix $\ell \in \mathbb{N}$, $\ell > 0$. There is an FST injection structure realizing the type $\{\ell_\infty\}$ over the domain $\{0, 1, \ldots, \ell - 1\}^+$.*

*Proof.* Consider the function which cycles each input letter through the letters in the alphabet. Then length is preserved and each nonempty word belongs to a cycle of length $\ell$. This function can be realized by the one-state DFST $(\{\iota\}, \iota, \{1\}, \{1\}, \{(\iota, i, \iota) : 0 \leq i < \ell\}, \{(\iota, \ell - 1, 0), (\iota, i, i + 1) : 0 \leq i < \ell - 1\})$.




**Lemma 7.** *Fix $\ell \in \mathbb{N}$, $\ell > 1$. There is a two-state DFST realizing the type $\{\ell_1^i : i \in \mathbb{N}\}$ over the domain $\{0, 1, \ldots, \ell - 1\}^*$. Namely, for each length $k$, all words of length $k$ are in a single cycle and so there is one cycle of length each power of $\ell$.*

*Proof.* Consider the function which treats strings over an $\ell$-letter alphabet as representations of integers in base-$\ell$, least significant bit first, and subtracts one. More explicitly, we associate a value $v(x_0 \cdots x_{n-1})$ with each string in $\{0, \ldots, \ell-1\}^*$ by defining.

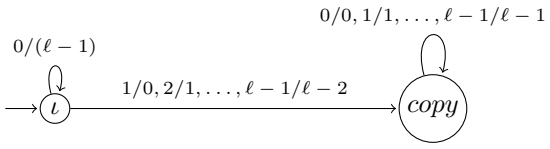
$$v(x_0 \cdots x_{n-1}) = \sum_i x_i \ell^i.$$

Then

$$f(x_0 \cdots x_{n-1}) = \begin{cases} \underbrace{(\ell-1)\cdots(\ell-1)}_{n \text{ times}} & \text{if } x_i = 0 \text{ for all } i \\ y_0 \cdots y_{n-1} & \text{such that } v(y_0 \cdots y_{n-1}) = v(x_0 \cdots x_{n-1}) - 1. \end{cases}$$

The function $f$ is realized by the two-state DFST

$$(\{\iota, c\}, \iota, \{0, 1, \ldots, \ell-1\}, \{0, 1 \ldots, \ell-1\}, \{(\iota, 0, \iota), (\iota, i, c), (c, j, c) : 1 \le i \le \ell-1, 0 \le j \le \ell-1\}, \\ \{(\iota, 0, \ell-1), (\iota, i, i-1), (c, j, j) : 1 \le i < \ell-1, 0 \le j \le \ell-1\}).$$




Of course, the fact that FST injection structures are closed under $\oplus$ and $\otimes$ means that we can use these structures to construct more complicated FST injection structures. For example if $\mathcal{A} = (A, f)$ and $\mathcal{B} = (B, g)$ are FST injection structures realizing $2^i_\infty$ and $3^i_\infty$,respectively, then it is easy to see that if $w \in A$ has orbit size $2^s$ and $v \in B$ has orbit size $3^t$, then $(w, v)$ must have orbit size $2^s 3^t$ in $\mathcal{A} \otimes \mathcal{B}$ so that will have infinitely many orbits of size $2^s 3^t$ for all $s, t \ge 1$. Similarly, if $\mathcal{C} = (C, f)$ and $\mathcal{D} = (D, g)$ are FST injection structures realizing the types $\{\omega_\infty\}$ and $\{\zeta_\infty\}$, respectively, then $\mathcal{C} \oplus \mathcal{D}$ realizes the type $\{\omega_\infty, \zeta_\infty\}$. Note in this situation, $\mathcal{C}$ and $\mathcal{D}$ are $\Delta^0_2$-categorical but not computably categorical and $\mathcal{C} \oplus \mathcal{D}$ is $\Delta^0_3$-categorical but not $\Delta^0_2$-categorical.

The type of an injection structure $(A, f)$ is said to be *bounded* if there is a finite $n$ such that the size of any finite cycle in $(A, f)$ is less than $n$. Our results show the following.

**Theorem 7.** *Any bounded type can be realized by an FST injection structure.*

**Theorem 8.** *There is an FST injection structure such that $(A, f)$ has exactly one $k$-cycle for every $k \ge 1$ and no infinite chains.*

*Proof.* First we shall build an FST injection structure $(E, g)$ such that $(E, g)$ has an exactly one $2n+3$ cycle for every $n \ge 0$ and no other cycles. The underlying alphabet $A$ will be $\{*, R, L, 1\}$ Each $2n+3$-cycle of $(E, g)$ will contain the string $*1^n*$. The next string in this cycle is the result of replacing the first $*$ with $R$ which we think of as move right symbol. This is followed by the sequence of strings that result by moving the symbol $R$ one space to the right until it is next to the second $*$ at which point the next string in the cycle is the result of replacing $R$ by $L$. We think of $L$ as a move left symbol. Then the next set of strings in the cycle is the sequence of strings that result by moving the $L$ one symbol to the left until it reaches the start. At that point, we replace $L$ by $*$ in which case we are back where we started. For example if $\alpha = *111*$,

then the cycle that contains $\alpha$ will be

$$\begin{aligned}
\alpha &= *111* \\
g(\alpha) &= R111* \\
g^2(\alpha) &= 1R11* \\
g^3(\alpha) &= 11R1* \\
g^4(\alpha) &= 111R* \\
g^5(\alpha) &= 111L* \\
g^6(\alpha) &= 11L1* \\
g^7(\alpha) &= 1L11* \\
g^8(\alpha) &= L111* \\
g^9(\alpha) &= *111*
\end{aligned}$$

Let $E = *\{1\}^* * \cup \{1\}^* R \{1\}^* * \cup \{1\}^* L \{1\}^* *$ which is clearly a regular set. We shall let $C$ stand for a copy state. That is, in state $C$, we read a letter $a$, we output $a$ and stay in state $C$. Let $S_0$ be the start state. Then in state $S_0$ do the following,

1. If we read $*$, we output $R$ and go to the copy state $C$.
2. If we read $L$, we output $*$ and go to the copy state $C$.
3. If we read 1, we output $\epsilon$ and go to state $S_1$.
4. If we read $R$, we output $\epsilon$ and go to state $S_R$.

The idea is that in state $S_0$ if we read either 1 or $R$, we output nothing but we remember where we came from. In state $S_1$, we do the following.

1. If we read 1, we output 1 and go to state $S_1$.
2. If we read $L$, we output $L1$ and go to the copy state $C$.
3. If we read $R$, we output $1R$ and go to the copy state $C$.
4. We will never read $*$ in state $S_1$ since we must see either $R$ or $L$ first, but for completeness if we read $*$, then we print $*$ and go to the copy state $C$.

Finally in state $S_R$, we do the following.

1. If we read 1, we output $1R$ and go to the copy state $C$.
2. If we read $*$, we output $L*$ and go to the copy state $C$.
3. We will never read $R$ or $L$ in state $S_R$ since we must see $*$ first, but for completeness if we read $R$ or $L$, then we print $*$ and go to the copy state.

It is easy to check that this defines a FST function $g$ such that every cycle contains $*1^n*$ for some $n$ and this cycle will be of length $2n + 3$. That is, we have one step to replace $*$ by $R$, $n$ steps to move $R$ past the 1s, one step to replace $R$ by $L$, $n$ steps to move $L$ past the 1s, and one step to replace $L$ by $*$. Thus $(E, g)$ has the desired behavior.

Next we modify the construction of $(E, g)$ to produce an FST injection structure such that $(F, h)$ has an exactly one $2n+4$ cycle for every $n \geq 0$ and no other cycles. The underlying alphabet $A$ will be $\{*, D, R, L, 1\}$. We will think of the $D$ symbol as delay symbol.

Each $2n + 4$-cycle of $(F, h)$ will contain the string $*1^n*$. Then our first step is to replace $*$ by $D$ and our second step is to replace $D$ by $R$ and then we proceed as in $(E, g)$. This will essentially add an extra string

to every cycle. For example, For example if $\alpha = *111*$, then the $(F, h)$ cycle of $\alpha$ will be

$$\begin{aligned}
\alpha &= *111* \\
h(\alpha) &= D111* \\
h^2(\alpha) &= R111* \\
h^3(\alpha) &= 1R11* \\
h^4(\alpha) &= 11R1* \\
h^5(\alpha) &= 111R* \\
h^6(\alpha) &= 111L* \\
h^7(\alpha) &= 11L1* \\
h^8(\alpha) &= 1L11* \\
h^9(\alpha) &= L111* \\
h^{10}(\alpha) &= *111*
\end{aligned}$$

Let $F = *\{1\}^**\cup\{1\}^*R\{1\}^**\cup\{1\}^*L\{1\}^**\cup D\{1\}^**$ which is clearly a regular set. Again we let $C$ be the copy state for this alphabet. Let $S_0$ be the start state. Then we only have to do the following modification in state $S_0$ do the following,

1. If we read $*$, we output $D$ and go to the copy state $C$.
2. If we read $D$, we output $R$ and go to the copy state.
3. If we read $L$, we output $*$ and go to the copy state $C$.
4. If we read 1, we output $\epsilon$ and to state $S_1$.
5. If we read $R$, we output $\epsilon$ and go to state $S_R$.

In states $S_1$ or $S_R$, we will never read $D$, but for completeness we say that if we read $D$ in either state $S_1$ or $S_R$, then we print $*$ and go to the copy state $C$.

It is easy to check that this defines a FST injection structure $(F, h)$ which has the described behavior.

Finally we let $(G, k)$ be a FST injection structure with one cycle of length 1 and one cycle of length 2. Then the FST injection structure $(E, g) \oplus (F, h) \oplus (G, k)$ has exactly one cycle of length $k$ for every $k \geq 1$.

We note that it easy to modify the FST injection structure $(A, f)$ to produce a FST injection structure $(A_k, f_k)$ for any $m \geq 2$ such that $(A_m, f_m)$ has exactly $m$ cycles of length $n$ for every $n \geq 1$. That is, let $q$ be a symbol which is not in the underlying alphabet of $A$. Then the strings of $(A_k, f_k)$ will be of the form $q^j\alpha$ where $1 \leq j \leq k$ and $\alpha \in B$. In the start state, when we read a $q$ we output $q$ and go back to the start state. Similarly, we can produce a FST injection structure $(A_\infty, f_\infty)$ such that $(A_\infty, f_\infty)$ has infinitely many cycles of length $n$ for every $n \geq 1$. That is, let $q$ be a symbol which is not in the underlying alphabet of $A$. Then the strings of $(A_\infty, f_\infty)$ will be $\{q\}^*A$ where again in the state, when we read a $q$ we output $q$ and go back to the start state.

## 4 Types over unary alphabets

**Theorem 9.** *The only types that can be realized over a unary alphabet are* $\{1_\infty\}$, $\{1_m, \omega_n\}$, *or* $\{1_m, \omega_\infty\}$.

*Proof.* Our results in Section 3 show that the types mentioned above can each be realized by FST injection structures over the alphabet $\{1\}$.

The characterization will be complete once we prove that these are the only possible types. We do so in the next two lemmas. For simplicity, we will identify $1^i$ with $i$ for this discussion. Thus the natural order on the integers $i, j$ becomes order by length on strings $1^i, 1^j$.

**Lemma 8.** *If $\mathcal{A} = (A, f)$ is an FST injection structure over $\{1\}^*$, then $f$ is monotonic, that is, $i < j$ implies $f(i) < f(j)$.*

*Proof.* For any $u, v \in A$, since $f$ is an injection, if $u < v$, then $f(u) \neq f(v)$. Moreover, by definition of FSTs, extensions can only add to the output and therefore, if $u < v$ then $f(u) < f(v)$.

**Lemma 9.** *If $\mathcal{A}$ is an FST injection structure over $\{1\}^*$, then*

*(1) $\mathcal{A}$ has no finite cycles of length greater than 1,*
*(2) $\mathcal{A}$ has no $Z$-chains, and*
*(3) if $\mathcal{A}$ has an $\omega$-chain, then $\mathcal{A}$ has only finitely many 1-chains.*

*Proof.* (1) Let $k > 1$ and suppose towards a contradiction that $w_1, w_2 = f(w_1), \ldots, w_k = f(w_{k-1})$ is a $k$-cycle. It follows that $w_1 \neq w_2$. Assume without loss of generality that $w_1 < w_2$. Then by Lemma 8, $w_2 < w_3 < \cdots < w_{k-1} < w_k$ and finally $w_k < w_1$, so that $w_1 < w_1$ by transitivity.

(2) Next suppose that $\ldots, w_{-2}, w_{-1}, w_0, w_1, \ldots$ is a $Z$-chain. Then each $w_i$ is distinct and, since there can be no infinite decreasing sequence, there must be some $i$ such that $w_{i+1} = f(w_i) > w_i$. It now follows from Lemma 8, by induction, that $w_{j+1} > w_j$ for all $j$. But then the sequence $w_0, w_{-1}, w_{-2}, \ldots$ is an infinite decreasing sequence.

(3) Finally, suppose that $w_0, w_1, \ldots$ is an $\omega$-chain. Then as in (2), there must be some $i$ such that $w_{i+1} = f(w_i) > w_i$. But it then follows from Lemma 8 that $w_{n+1} > f(w_n)$ for all $n \geq i$. Now if $j > w_i$ and $f(j) = j$, then, for some $n \geq i$, $w_n < j < w_{n+1}$. It follows that $f(j) > f(w_n) = w_{n+1} > j$, so that $\{j\}$ is not a 1-cycle.

**Corollary 1.** *Unary FST realizable structures are not closed under disjoint unions.*

*Proof.* Otherwise, one could form the disjoint union of a structure with infinitely many one-cycles and a structure with one $\omega$ chain and get a type not in this list.

**Corollary 2.** *The FO-theory of unary FST realizable injections structures is decidable.*

*Proof.* In [6], it was shown that for any injection structure $\mathcal{A}$, $Th(\mathcal{A})$ is decidable if and only if $\chi(\mathcal{A})$ is computable. The result then immediately follows from Theorem 9.

In contrast to Theorem 9, we can show that are no such restrictions on types realized by FST injection structures over a two letter alphabet.

**Theorem 10.** *Any type realizable with a finite alphabet can be realized (with $\epsilon$-outputs) using an alphabet with two symbols.*

*Proof.* Given an FST injection structure $\mathcal{A} = (A, f)$ over a finite alphabet, we may assume without loss of generality that $card(\Sigma) = card(\Gamma) = 2^n$ for some $n$ and hence we may assume that $\Sigma = \Gamma = \{0,1\}^n$. Now we can simulate the DFST using alphabet $\{0,1\}$ as follows. For each state $q$, we will have states $q^\sigma$ for every $\sigma \in \{0,1\}^{<n}$. If $|\sigma| < n-1$, then the transition is $\delta'(q^\sigma, a) = q^{\sigma a}$ and the output is $\tau'(q^\sigma, a) = \epsilon$. If $|\sigma| = n-1$, then $\delta'(q^\sigma, a) = \delta(q, \sigma a)^\epsilon$ and the output $\tau'(q^\sigma, a) = \tau(q, \sigma a)$. Thus, from state $q$, if the next $n$ bits of the input tape are given by $\sigma$, then after $n$ steps in the simulation, we will read $\sigma$ and output $\tau(q, \sigma)$.

## 5 Restrictions on types realized by FST injection structures

The main result of this section is a kind of analogue of the Pumping Lemma for length preserving FST injection structures over a finite alphabet $\Sigma$ and whose domain is $\Sigma^*$ or $\Sigma^+$.

**Theorem 11.** *Suppose that $\mathcal{A} = (\Sigma^*, f)$ is a length preserving FST injection structure where $f$ is defined by a transducer $T = (Q, \iota, \Sigma, \Gamma, \delta, \tau)$. Then there is a constant $c_{|Q|,|\Sigma|,k}$ depending on $|Q|$, $|\Sigma|$ and $k$ such that if $\mathcal{A}$ has more than $c_{|Q|,|\Sigma|,k}$ orbits of size $k$, then $\mathcal{A} = (\Sigma^*, f)$ must have infinitely many orbits of size $k$.*

*Proof.* Suppose that $\sigma^{(0)}, \ldots, \sigma^{(k-1)}$ is a $k$-cycle of $f$, i.e., $f(\sigma^{(i)}) = \sigma^{(i+1)}$ for $0 \leq i < k-1$ and $f(\sigma^{(k-1)}) = \sigma^{(0)}$. Since $\mathcal{A} = (\Sigma^*, f)$ is length preserving, we must have $|\sigma^{(0)}| = |\sigma^{(1)}| = \cdots = |\sigma^{(k-1)}|$. Now suppose that $|\sigma^{(0)}| = n > |Q|^k|\Sigma|^k + 1$. Then for each $0 \leq i \leq k-1$, let $\sigma^{(i)} = \sigma_1^{(i)} \ldots \sigma_n^{(i)}$ and let $a_j^{(i)}$ denote the state of $T$ after reading letters $\sigma_1^{(i)} \ldots \sigma_{j-1}^{(i)}$. Thus as our FST processes the string $\sigma_1^{(i)} \ldots \sigma_n^{(i)}$, the FST is in state $a_j^{(i)}$ when it reads the letter $\sigma_j^{(i)}$. It follows that there must be some $1 \leq s < t \leq n$ such that $(\sigma_s^{(i)}, a_s^{(i)}) = (\sigma_t^{(i)}, a_t^{(i)})$ for all $i = 1, \ldots, k$. That is, there are only $|Q|^k|\Sigma|^k$ possible choices for the sequence $((\sigma_s^{(1)}, a_s^{(1)}), \ldots, (\sigma_s^{(k)}, a_s^{(k)})$ as $s$ varies from 1 to $n$. Hence there must exist such an $s$ and $t$ since $n > |Q|^k|\Sigma|^k + 1$. But this means that for each $1 \leq i \leq k$, as $T$ processes $\sigma_s^{(i)} \ldots \sigma_{t-1}^{(i)}$ starting in state $a_s^{(i)}$, it will end up in state $a_s^{(i)}$ and output $\sigma_s^{(i+1)} \ldots \sigma_{t-1}^{(i+1)}$ where for $i+1$, we take the addition modulo $k$. It then follows that for all $j \geq 1$ as $T$ processes $(\sigma_s^{(i)} \ldots \sigma_{t-1}^{(i)})^j$ starting in state $a_s^{(i)}$, it will end up in state $a_s^{(i)} = a_t^{(i)}$ and output $(\sigma_s^{(i+1)} \ldots \sigma_{t-1}^{(i+1)})^j$. Hence for all $j \geq 1$, the sequences

$$\sigma_1^{(i)} \ldots \sigma_{s-1}^{(i)} (\sigma_s^{(i)} \ldots \sigma_{t-1}^{(i)})^j \sigma_t^{(i)} \ldots \sigma_n^{(i)}$$

for $i = 1, \ldots, k$, will also be in a $k$-cycle in $\mathcal{A}$. Since we are assuming the domain of $\mathcal{A}$ is all of $\Sigma^*$, all these cycles are in $\mathcal{A}$ so that $\mathcal{A}$ would have infinitely many $k$-cycles.

Our argument allows us to give a simple bound for $c_{|Q|,|\Sigma|,k}$. That is there are

$$\sum_{i=0}^{|Q|^k|\Sigma|^k+1} |\Sigma|^i = \frac{|\Sigma|^{|Q|^k|\Sigma|^k+2} - 1}{|\Sigma| - 1}$$

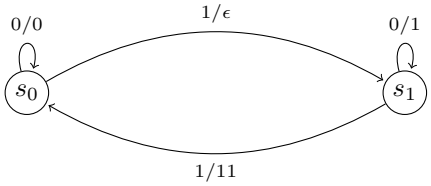
strings in $\Sigma^*$ of length $\leq |Q|^k|\Sigma|^k + 1$. If each of these strings were involved in a $k$-cycle, then we would have $\lfloor \frac{|\Sigma|^{|Q|^k|\Sigma|^k+2}-1}{k(|\Sigma|-1)} \rfloor$ $k$ cycles. Thus $c_{|Q|,|\Sigma|,k} = \lfloor \frac{|\Sigma|^{|Q|^k|\Sigma|^k+2}-1}{k(|\Sigma|-1)} \rfloor$ and so if $\mathcal{A}$ has more than $c_{|Q|,|\Sigma|,k}$, then we will have at least one $k$-cycle of type described above so that $\mathcal{A}$ would have infinitely many $k$-cycles.

We note that the proof of Theorem 11 used the fact that the domain of $\mathcal{A}$ was all of $\Sigma^*$ to ensure that all the strings $\sigma_1^{(i)} \ldots \sigma_{s-1}^{(i)} (\sigma_s^{(i)} \ldots \sigma_{t-1}^{(i)})^j \sigma_t^{(i)} \ldots \sigma_n^{(i)}$ for $j \geq 1$ are in the domain of $\mathcal{A}$. However, we did not need that $T$ is length preserving, but only that certain types of cycles in $T$ are length preserving. That is, a state-cycle $C$ in $T$ consists of state $s \in S$ and string of symbols $\sigma_1 \ldots \sigma_n \in \Sigma$ such that when $T$ processes $\sigma_1 \ldots \sigma_i$ starting state $s$, it ends in state $s_i$ for $1 \leq i \leq n$, $s_1, \ldots, s_n$ are pairwise distinct, and $s_n = s$. We then say that a state-cycle $C$ is *state-cycle length preserving* if when $T$ processes $\sigma_1 \ldots \sigma_n$ starting in state $s$, the corresponding output is of length $n$. We say that $T$ has *state-cycle length preserving cycles* if all state-cycles of $C$ are state-cycle length preserving. This condition would ensure that for each string $\sigma_s^{(i)} \ldots \sigma_{t-1}^{(i)}$ in the argument above, the output string $\sigma_s^{(i+1)} \ldots \sigma_{t-1}^{(i+1)}$ will have length $t-s$. Note that the individual $\sigma_j^{(i)}$s might be strings rather than symbols, when $T$ is state-cycle length preserving.

Thus we obtain the following corollary from our proof of Theorem 11.

**Corollary 3.** *Suppose that $\mathcal{A} = (\Sigma^*, f)$ is an FST injection structure such that any cycle of $f$ consists of strings of the same length and where $f$ is defined by a transducer $T = (Q, \iota, \Sigma, \Gamma, \delta, \tau)$ that is state-cycle length preserving. Then there is a constant $c_{|Q|,|\Sigma|,k}$ depending on $|Q|$, $|\Sigma|$ and $k$ such that if $\mathcal{A}$ has more than $c_{|Q|,|\Sigma|,k}$ orbits of size $k$, then $\mathcal{A}$ must have infinitely many orbits of size $k$.*

It is easy to construct a transducer $T$ which satisfies the hypothesis of Corollary 3 but is not length preserving. For example, the following transducer has this property.



Here is another corollary to the proof of Theorem 11.

**Corollary 4.** *There is a computable type $\chi$ which is not recognized by any FST injection structure satisfying the conditions of Corollary 3.*

*Proof.* Consider the computable function $c(q,s,k) = c_{q,s,k}$, assumed to be monotonic, and, for each $k$, let $n(k)$ be larger than $c(q,s,k)$ for each $q,s \leq k$.

Suppose, towards a contradiction, that there is an FST injection structure realizing the type which has $n(k)$ many $k$-cycles for each finite $k$. Let $q$ be the number of states in the DFST realizing this structure and let $s$ be the number of letters in the alphabet. Then for $k = max\{q,s\}$, if the structure has $n(k)$ $k$-cycles, it must have infinitely many $k$-cycles, contradicting the assumption that this transducer realizes the given type.

A stronger result can be proved if we require the FST to be length preserving.

**Theorem 12.** *There is a computable type $\chi$ with at most one $k$-cycle for every finite $k$ and which is not realized by any length preserving FST injection structure.*

*Proof.* It is easy to see that we can effectively determine whether a transducer $T = (Q, \iota, \Sigma, \Gamma, \delta, \tau)$ defines a length preserving injection structure by looking at the definition of $\tau$. It follows that we can effectively list all length preserving FST injection structures by listing their corresponding FSTs $T_0, T_1, \ldots$.

Now suppose that $T_i = (Q_i, \iota_i, \Sigma_i, \Gamma_i, \delta_i, \tau_i)$ and $(A_i, f_i)$ is the injection structure determined by $T_i$. Note that $S_i = (Q_i, \iota_i, \Sigma_i, \delta_i)$ is a deterministic finite automaton (DFA) so that it follows from the standard pumping for $DFA$s that we can effectively decide if the language accepted by $S_i$, namely $A_i$, is infinite. It follows from results of Cenzer, Harizanov, and Remmel [5] that every c.e. character is the character of some computable injection structure so that we need only produce a character $\chi = \{(k,1) : k \in B\}$ for some c.e. set $B$ that is not the character of $(A_i, f_i)$ for any $i$. We enumerate a computable $B$ with this property in stages. We let $B_s$ denote the set of elements of enumerated into $B$ by the end of stage s. Our construction will ensure that $|B_s| = s+1$ so that a computable injection with character $\chi$ will be infinite.

**Stage 0.** If $A_0$ is finite, then let $B_0 = \{1\}$. If $A_0$ is infinite, then we list the strings in $A_0$ first by length and within strings of the same length lexicographically as $a_0^{(0)}, a_1^{(0)}, \ldots$. Since $T_0$ is length preserving, we can effectively compute the orbit $\mathcal{O}_{f_0}(a_s^{(0)})$ for any $a_s^{(0)}$ for any $s$. Then we let $k_0 = |\mathcal{O}_{f_0}(a_0^{(0)})|$ and let $B_0 = \{k_0 + 1\}$. Our construction will ensure that no element smaller that $k_0 + 1$ will be in $B$ so that $(A_0, f_0)$ will not be isomorphic to any injection structure with character $\chi$.

**Stage s+1.** Assume that $B_s = \{b_0 < b_1 \cdots < b_s\}$ and for all $i \leq s$, either (i) $A_i$ is finite, (ii) $(A_i, f_i)$ has two distinct orbits of the same size, or (iii) $(A_i, f_i)$ has an orbit of size $d_i$ where $d_i < b_s$ and $d_i \notin B_s$. If $A_{s+1}$ is finite, then we let $b_{s+1} = b_s + 1$. If $A_{s+1}$ is infinite, then we list the strings in $A_{s+1}$ first by length and within strings of the same length lexicographically as $a_0^{(s+1)}, a_1^{(s+1)}, \ldots$. Since $T_{s+1}$ is length preserving, we can effectively compute the orbit $\mathcal{O}_{f_{s+1}}(a_t^{(s+1)})$ for any $a_t^{(s+1)}$ for any $t$. Then we compute the orbits $\mathcal{O}_{f_{s+1}}(a_0^{(s+1)}), \mathcal{O}_{f_{s+1}}(a_1^{(s+1)}), \ldots$ until either we find $i = j$ such that $|\mathcal{O}_{f_{s+1}}(a_i^{(s+1)})| = |\mathcal{O}_{f_{s+1}}(a_j^{(s+1)})|$ in which case we set $b_{s+1} = b_s + 1$ or we find an $i$ such that $|\mathcal{O}_{f_{s+1}}(a_i^{(s+1)})| > b_s$ in which case we set $b_{s+1} = |\mathcal{O}_{f_{s+1}}(a_i^{(s+1)})| + 1$. Then we let $B_{s+1} = \{b_0 < b_1 \cdots < b_s < b_{s+1}\}$.

It is easy to see that $B$ is an infinite computable set and that for all $i$ either $A_i$ is finite, $(A_i, f_i)$ has two orbits of the same size, or $(A_i, f_i)$ has an orbit of size $k$ which is not in $B$. It then follows that if $(A, f)$ is a computable injection structure with character $\chi$, then $(A, f)$ is not isomorphic of $(A_i, f_i)$ for any $i$.

## 6 Graphs of FST injection structures

It is natural to ask whether there is a difference between FST injection structures and automatic structures $\mathcal{A} = (A, gr(f))$ where $gr(f)$ is the graph of the function $f$, i.e., the set of all pairs $(a, f(a))$ for $a \in f$. Since

an automatic structure $\mathcal{A} = (A, gr(f))$ has a decidable theory, it would follow that if $gr(f)$ is recognizable by a DFA, then $(A, f)$ would also have a decidable theory. However, it is easy to construct a DFST with no $\epsilon$-outputs whose graph is not recognizable by a 2-tape synchronous automaton.

**Theorem 13.** *There is an FST realizable injection structure $(A, f)$ over a unary alphabet such that the graph of $f$ is not recognizable by a DFA.*

*Proof.* For example, consider the function $1^i \mapsto 1^{2i}$. The graph of this function does not satisfy the Constant Growth Lemma [15] and hence is not automatic.

On the other hand, it is easy to see that if $T = (Q, \iota, \Sigma, \Gamma, \delta, \tau)$ is length preserving, i.e. $\tau(q, a) \in \Gamma$ for all $q \in Q$ and $a \in \Sigma$, then we can use the transition diagram for $\delta$ and $\tau$ to construct of DFA which accepts the graph $(w, \tau(w))$ for all $w$ accepted by the DFA $(Q, \iota, \Sigma, \delta)$. Thus we have the following theorem.

**Theorem 14.** *If a relation (not necessarily an injection, or even a function) is realized by a DFST all of whose moves are length preserving (one output symbol for each input symbol), then the graph of the relation is recognizable by a 2-tape synchronous automaton.*

In fact, we can prove a stronger theorem

**Theorem 15.** *If $(A, f)$ is an FST injection structure where $f$ is realized by a DFST which has length preserving cycles, then the graph of the relation is recognizable by a 2-tape synchronous automaton and, hence, $(A, f)$ has a decidable theory.*

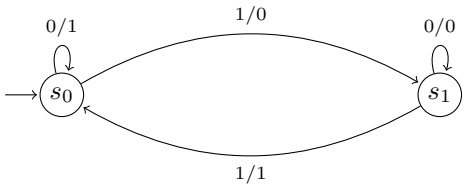
## 7 Open questions and further research

There are two major questions that we have not yet been able to answer.

1. Does every FST injection structure $(A, f)$ have a decidable theory?
2. Can one classify the types that can be realized by FST injection structures?

For question (1), we have a partial result, Theorem 15. Namely, that if $f$ is realized by a transducer with length preserving cycles, the answer is yes.

Question (2) seems to be a much harder question. We should note that determining the types of injection structures realized by very simple FSTs can be challenging. For example, consider the FST injection structure $\mathcal{D}_2 = (\{0, 1\}^+, f)$ where $f$ is specified by the following transducer $T_2$ where $s_0$ is the start state.




Computational evidence suggests that for all $n \geq 1$, all strings of $\sigma$ such that $2^n \leq |\sigma| < 2^{n+1}$ are contained in $2^{n+1}$ cycles. However we have not been able to prove this even though $\mathcal{D}_2$ has a decidable theory. We have the following partial result.

**Theorem 16.** *All strings of the form $0^k$ where $2^n \leq k < 2^{n+1}$ are contained in $2^{n+1}$ cycles in $\mathcal{D}$. Moreover, all elements of $A$ lie in cycles whose length is a power of 2.*

*Proof.* For any $k$, let $0^k, f(0^k), f^2(0^k), \ldots f^{n_k}(0^k)$ be the orbit of $0^k$ in $\mathcal{D}$ and let $a_{k,p}$ be state of the transducer after processing $f^p(0^k)$. It will be useful to picture this information in an array as follows

$$\begin{array}{c:l} 0^k & : a_{k,0} \\ f(0^k) & : a_{k,1} \\ f^2(0^k) & : a_{k,2} \\ \vdots & \vdots \\ f^{n_k}(0^k) & : a_{k,p}. \end{array}$$

For example, the first few such arrays are as follows.

$$\begin{array}{c:c:c:l} & & & 0000 : s_0 \\ & & & 1111 : s_0 \\ & 00 : s_0 & 000 : s_0 & 0101 : s_0 \\ 0 : s_0 & 11 : s_0 & 111 : s_1 & 1001 : s_0 \\ 1 : s_1 & 01 : s_1 & 010 : s_1 & 0001 : s_1 \\ & 10 : s_1 & 100 : s_1 & 1110 : s_1 \\ & & & 0100 : s_1 \\ & & & 1000 : s_1. \end{array}$$

We shall prove by induction that all elements lie in a cycle whose length is a power of 2. Clearly this is true for words of length 1. Now suppose that $|\sigma| \geq 1$ and $\sigma$ is in a $2^r$-cycle specified by

$$\begin{array}{c:l} \sigma & : b_0 \\ f(\sigma) & : b_1 \\ \vdots & \vdots \\ f^{2^r-1}(\sigma) & : b_{2^r-1}. \end{array}$$

Since processing 0 keeps the transducer in the same state and processing 1 always changes the state, it follows that $b_i = s_0$ if and only if $f^i(\sigma)$ has an even number of 1s. Similarly if $i \in \{0,1\}$, then processing $i$ in state $s_0$ will output $1-i$ and processing $i$ in state $s_1$ will output $i$. Thus if we consider the cycle of $\sigma i$ where $i \in \{0,1\}$, then $f^{2^r}(\sigma i)$ will equal $\sigma i$ if $b_0, \ldots, b_{2^r-1}$ contains an even number of $s_0$s and will equal $\sigma(1-i)$ if $b_0, \ldots, b_{2^r-1}$ contains an odd number of $s_0$s. It follows that $\sigma i$ is part a $2^r$ cycle if $b_0, \ldots, b_{2^r-1}$ contains an even number of $s_0$s and is part of $2^{r+1}$ cycle if $b_0, \ldots, b_{2^r-1}$ contains an odd number of $s_0$s.

We claim that for all $n \geq 1$, $0^{2^n}$ is in a $2^{n+1}$ cycle whose final states are $s_0^{2^n} s_1^{2^n}$ and that $0^{2^n+2^n-1}$ is in a $2^{n+1}$ cycle whose final states are $s_0 s_1^{2^{n+1}-1}$. We proceed by induction on $n$. We have verified this for $n = 1$.

Now consider the cycle of $0^{2^n+2^n-1}0 = 0^{2^{n+1}}$. Since the final states for the first $2^{n+1}$ steps of applying $f$ to $0^{2^{n+2}-1}$ are $s_0 s_1^{2^{n+1}-1}$, it is easy to see that final elements of applying $f$ to $0^{2^{n+1}}$ will be $01^{2^{n+1}}$ so that $f^{2^{n+1}}(0^{2^{n+2}}) = 0^{2^{n+1}-1}1$. Moreover, it follows that $f^i(0^{2^{n+1}})$ will have an even number of 1s for all $0 \leq i < 2^{n+1}$ so that final states for those strings will be $s_0^{2^{n+1}}$ since they will contain an even number of 1s. Similarly, the final elements of the result of the first $2^{n+1}$ steps of applying $f$ to $0^{2^{n+1}-1}1$ will be $10^{2^{n+1}-1}$ so that $f^{2^{n+1}}(0^{2^{n+2}-1}1) = 0^{2^{n+1}-1}0$. Moreover the final states of these strings will be $s_1^{2^{n+1}}$ since they will all contain an odd number of strings. Hence the cycle of $0^{2^{n+1}}$ is a cycle of length $2^{n+2}$ whose final states are $s_0^{2^{n+1}} s_0^{2^{n+1}}$.

Next consider the first $2^{n+1}$ steps of applying $f$ to $0^{2^{n+1}+2^{n+1}-1} = 0^{2^{n+2}-1}$. Since the final states for first $2^{n+1}$ steps of applying $f$ are $s_0^{2n+1}$, the resulting final states and strings will be

$$\begin{array}{c:l} 0^{2^{n+1}}0^{2^{n+1}-1} & : s_0 \\ f(0^{2^{n+1}})f(0^{2^{n+1}-1}) & : s_1 \\ f^2(0^{2^{n+1}})f^2(0^{2^{n+1}-1}) & : s_1 \\ \vdots & \vdots \\ f^{2^{n+1}-1}(0^{2^{n+1}})f^{2^{n+1}-1}(0^{2^{n+1}-1}) & : s_1. \end{array}$$

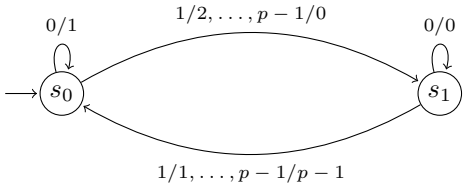
Since $f^{2^{n+1}}(0^{2^{n+1}-1}) = 0^{2^{n+1}-1}$ , when we apply $f$ again, we first get $f^{2^{n+1}}(0^{2^{n+1}})$ which will end in state $s_1$ and then we will see a string of 0s which will just be copied in state $s_1$. It follows for the next $2^{n+1}$ steps of applying $f$, we will get

$$\begin{aligned} f^{2^{n+1}}(0^{2^{n+1}})0^{2^{n+1}-1} &: s_1 \\ f^{2^{n+1}+1}(0^{2^{n+1}})0^{2^{n+1}-1} &: s_1 \\ \vdots &\quad \vdots \\ f^{2^{n+2}-1}(0^{2^{n+1}})0^{2^{n+1}-1} &: s_1. \end{aligned}$$

As $f^{2^{n+2}}(0^{2^{n+1}}) = 0^{2^{n+1}}$, it follows that the cycle of $0^{2^{n+2}-1}$ is of length $2^{n+2}$ whose final states are $s_0 s_1^{2^{n+2}-1}$.

Let $\overline{\mathcal{D}}_2 = (\{0,1\}^+, g)$ where $g$ is the function induced by the same transducer $T_2$ except that the start state is $s_1$. In this case for any $\sigma \in \{0,1\}^+$, the cycle of $0^n 1\sigma$ will just be $0^n 1\sigma, 0^n 1 f(\sigma), 0^n 1 f^2(\sigma), \ldots$. As before we can show that all cycles are powers of 2 so that since we know $\mathcal{D}$ has $2^n$-cycles for all $n \geq 1$, $\overline{\mathcal{D}}_2 = (\{0,1\}^+, g)$ will have infinitely many $2^n$-cycles for all $n \geq 1$.

Generalizing $T_2$ gives a family of two-state DFSTs $T_p$ over the alphabet $\{0, \ldots, p-1\}$ pictured below.




We can prove the following theorem about such transducers.

**Theorem 17.** *Let $\mathcal{D}_p = (\{0, \ldots, p-1\}^+, f_p)$ be the FST injection structure where $f_p$ is induced by $T_p$. If $p \geq 3$ is prime, then for all $n \geq 1$, all strings of length $n$ form a $p^n$-cycle in $\mathcal{D}_p$. Thus, the type $p_1^n$ is realized by this FST injection structure.*

*Proof.* We shall show by induction on $n$ that all strings of length $n$ are part of $p^n$ cycle. This is clear for $n = 1$. That is, since the start state is $s_0$, it is easy to see that $f_p(i) = i+1$ for $0 \leq i \leq p-2$ and $f_p(p-1) = 0$. Thus $0, 1, \ldots, p-1, 0$ is $p$-cycle in $\mathcal{D}_p$. Next suppose that strings of length $n$ form an $p^n$-cycle in $\mathcal{D}_p$. Let $\sigma^{0,n}, \ldots, \sigma^{p^n-1,n}, \sigma^{0,n}$ be this $p^n$-cycle where $\sigma^{0,n} = 0^n$. Let $s^{j,n}$ denote the final state of the FST when we process $\sigma^{j,n}$ starting in state $s_0$. For example, $\sigma^{0,n} = 0^n$ so that $\sigma^{1,n} = 1^n$ and $s^{0,n} = 0$.

For any string $\sigma \in \{0, \ldots, p-1\}^*$ and any $0 \leq i \leq p-1$, let $|\sigma|_i$ denote the number of $i$'s which occur in $\sigma$ and

$$|\sigma|_{\neq 0} = |\sigma|_1 + |\sigma|_2 + \cdots + |\sigma|_{p-1}.$$

Then for any $j$, when we process $\sigma^{j,n}$ starting in state $s_0$, we will end in state $s_0$ if $|\sigma^{j,n}|_{\neq 0}$ is even and we will end in state $s_1$ if $|\sigma^{j,n}|_{\neq 0}$ is odd. Note that if $n = 2k$, then the number of of strings of length $n$ such that $|\sigma|_{\neq 0}$ is odd equals

$$\binom{2k}{2k-1}(p-1)^{2k-1} + \binom{2k}{2k-3}(p-1)^{2k-3} + \cdots + \binom{2k}{1}(p-1)$$

which clearly is an even number. Similarly, if $n = 2k+1$, then the number of of strings of length $n$ such that $|\sigma|_{\neq 0}$ is odd equals

$$\binom{2k+1}{2k+1}(p-1)^{2k+1} + \binom{2k+1}{2k-1}(p-1)^{2k-1} + \cdots + \binom{2k+1}{1}(p-1)$$

which is also an even number. It follows that in the sequence of state $s^{0,n}, s^{1,n}, \ldots, s^{p^n-1,n}$ the number of $s_1$s is equivalent to $i \mod p$ for some $0 < i \leq p-1$ which means that the number $s_0$s in the sequence is

equivalent to $p - i \mod p$ for some $0 < i \leq p - 1$. Now suppose that we fix an $t$ such that $0 \leq t \leq p - 1$ and we apply $f_p\ p^n$ times starting with the string $0^n t$. Then we will get a sequence

$$\sigma^{0,n} a^{0,t}, \sigma^{1,n} a^{1,t}, \ldots, \sigma^{p^n-1} a^{p^n-1,t}$$

where $a^{0,t} = t$ since after processing $0^n$ we end in state $s_0$. For $b \geq 0$, $a^{b+1,t}$ equals $a^{b,t}$ is $s^{b,n} = s_1$ and $a^{b+1,t}$ equals $a^{b,t} + 1 \mod p$ is $s^{t,n} = s_0$. This implies that if we apply $f_p\ p^n$ times starting with the string $0^n t$, we will end up with the string $0^n(t + p - i \mod p)$. Thus if we start with the string $0^{n+1}$ and apply $f_p\ p^n$ times will produce the string $0^n(p - i)$. If we then apply $f_p$ another $p^n$ times, we will end up with the string $0^n(2(p - i) \mod p)$. It follows that if we start with the string $0^{n+1}$ and apply $f_p\ kp^n$ times, we end up with the string $0^n(k(p - i) \mod p)$. Since $p \geq 3$ and $p$ is prime, then we it will take $p^{n+1}$ applications of $f_p$ before we can return to $0^{n+1}$. Hence, $0^{n+1}$ must be part of a $p^{n+1}$-cycle which means that the cycle determined by $0^{n+1}$ must consist of all the strings of length $n + 1$.

Finally we should observe that our proof shows that if we choose $s_1$ as the start state for the FST in Theorem 17, then for any $k \geq 0$, the strings $0^k 10^n$ will generate a $p^n$-cycle where each string in the cycle starts with $0^k 1$. It follows that if we restrict the set of strings of the form $0^k 1\sigma$ where $k > 0$ and $\sigma \in \{0, 1, \ldots, p - 1\}^+$, then we will obtain an FST injection structure such that there are infinitely many $p^n$-cycles for each $n \geq 1$.

## References


1. Bennett, C.H. Logical Reversibility of Computation. *IBM J. Res. and Develop.*, 525-532, 1973.
2. Ash, C. and Knight, J. *Computable Structures and the Hyperarithmetical Hierarchy*, Elsevier, Amsterdam, 2000.
3. Blumensath, A. Automatic Structures, Diploma Thesis, RWTH Aachen, 1999.
4. Blumensath, A. and Grädel, E. Automatic Structures. *Proc. 15th LICS*, 51-62, 2000.
5. D. Cenzer, V. Harizanov, and J.B. Remmel, Effective categoricity of injection structures, in *Models of Computation in Context, CIE 2011*, B. Loewe, D. Normann, I. Soskov, A. Soskova, editors, Springer Lecture Notes in Computer Science 6735 (2011), pp. 51–60
6. D. Cenzer, V. Harizanov, and J.B. Remmel, Computability-theoretic properties of injection structures, to appear in *Algebra and Logic*.
7. Delhommé, C. Automaticité des ordinaux et des graphes homogènes. *C.R. Acadèmie des sciences Paris, Ser. I* **339**, 5-10, 2004.
8. Hodgson, B.R. On Direct Products of Automaton Decidable Theories. *Theoret. Comp. Sci.* **19**, 331-335, North-Holland, 1982.
9. Khoussainov, B. and Nerode, A. Automatic presentations of structures. LNCS **960**, 367-392, 1995.
10. Khoussainov, B. and Shore, R.A. Effective Model Theory: The Number of Models and Their Complexity. *Models and Computability, Invited Papers from LC 1997* (S.B. Cooper and J.K. Truss, eds.) LMSLNS **259**, 193-240, Cambridge University Press (Cambridge, England), 1999.
11. Khoussainov, B., Rubin, S. and Stephan, F. On automatic partial orders. *Proc. 18th LICS*, 168-177, 2003.
12. Khoussainov, B., Nies, A., Rubin, S. and Stephan, F. Automatic Structures: Richness and Limitations. *Proc. 19th LICS*, 44-53, 2004.
13. Khoussainov, B., Rubin, S. and Stephan, F. Automatic linear orders and trees. *ACM Trans. on Comput. Logic* **6** Number 4, 675-700, 2005.
14. Khoussainov, B., Liu, J. and Minnes, M. Unary Automatic Graphs: An Algorithmic Perspective. *Proc. TAMC* LNCS **4978**, 548-559, 2008.
15. Rubin, S. *Automatic Structures*, PhD Thesis, University of Auckland, 2004.
16. R.I. Soare, *Recursively Enumerable Sets and Degrees*, (Springer, Berlin, 1987).