

Deciding whether the frontier of a regular tree is scattered

Stephen L. Bloom*

Department of Computer Science

Stevens Institute of Technology

Hoboken, NJ 07030

Zoltán Ésik†

Institute for Informatics

University of Szeged

Szeged, Hungary

1. Introduction

While thinking of how to generalize some facts about ordinal words (labelings of ordinals) in [BlCho01] to linearly ordered words (labelings of linear orders), the authors rediscovered the natural classes of the “regular words” and the “scattered regular words”, described here. It turns out that these words are isomorphic to the frontiers of regular trees, considered earlier by Courcelle [Cour78], Heilbrunner [Heil80], and Thomas [Thom86]. The current paper contains some new descriptions of this class related to properties of regular sets of binary strings, and uses finite automata to decide various natural questions concerning these words. In particular, we show that there is a polynomial time algorithm to decide, given a DFA which determines a regular word, whether this word is scattered.

*Partially supported by NSF grant 0119916.

†Partially supported by BRICS, Aalborg, Denmark, and NSF grant 0119916 and the National Foundation of Hungary for Scientific Research, grant T35163.

2. Preliminaries

A linearly ordered set, or “linear order”, is usually denoted $P = (P, \leq_P)$, or just P, Q , etc. We let $\mathbf{1}$ denote a one element linearly ordered set. ω denotes the usual ordering on the nonnegative integers, ω^{op} denotes the usual order on the negative integers, isomorphic to the reverse of ω , and \mathbb{Q} denotes the linearly ordered set of the rational numbers. A rather complete reference for facts about linear orders is the book [Ro82].

In this paper, we assume all “alphabets” are initial subsets $\{a_1, \dots, a_n\}$ of the countable set $\{a_1, a_2, \dots\}$. We sometimes use a, b or $0, 1$ to denote a_1, a_2 , respectively. For a nonnegative integer n , we let $[n]$ denote the set $\{1, 2, \dots, n\}$, so that $[0]$ is the empty set. A countable set is either finite or countably infinite.

By a **word on the alphabet** A , we mean a labeled linearly ordered countable set. (We have no need here for labelings of uncountable linear orders.) So, more formally, a word on A is a triple, (L_u, \leq_u, u) , where (L_u, \leq_u) , called the **underlying order** of the word u , is a countable linearly ordered set, and $u : L_u \rightarrow A$ is a function. We abbreviate the triple (L_u, \leq_u, u) by just u . Two words u, v are **isomorphic** if there is a bijection $f : L_u \rightarrow L_v$ such that for all $x, y \in L_u$,

$$\begin{aligned} x \leq_u y &\iff f(x) \leq_v f(y), & \text{and} \\ v(f(x)) &= u(x). \end{aligned}$$

We usually identify isomorphic words.

We will be concerned with operations $u, v \mapsto uv$, $u \mapsto u^\omega$, $u \mapsto u^{\omega^{op}}$, $(u_1, \dots, u_k) \mapsto [u_1, \dots, u_k]^\eta$, $k \geq 1$, on words, and corresponding operations on linear orders. Each of these operations is defined by means of **word substitution**. First, we consider linear orders.

Definition 2.1. Suppose that (L, \leq) is a linear order, and for each $x \in L$, let (K_x, \leq_x) be a linear order. The ordering $\sum_{x \in L} K_x$, the **generalized sum** of the orders K_x , is defined as follows: the underlying set is the set of pairs (k, x) with $x \in L$ and $k \in K_x$ ordered by:

$$(k, x) \leq (k', x') \iff x < x' \text{ or } (x = x' \text{ and } k \leq k').$$

Definition 2.2. Let u be a word on the alphabet $A = \{a_1, \dots, a_n\}$, and let v_{a_i} be a word on the alphabet B , for each $i \in [n]$. The alphabets A, B need not be the same. We define $w = u(a_1/v_{a_1}, \dots, a_n/v_{a_n})$, the word obtained by substituting v_{a_i} for each occurrence of a_i in u as follows. L_w is the linear order $\sum_{x \in L_u} L_{u(x)}$, defined just above, labeled as follows:

$$w(k, x) := v_{u(x)}(k), \quad x \in L_u, k \in L_{v_{u(x)}}.$$

We call a word on a finite linear order a **string**, and use the usual notion for them, so that for example, aba denotes the string u on the 3 element chain, say $1 < 2 < 3$, such that $u(1) = u(3) = a$ and $u(2) = b$. In particular, the empty word λ is a string. We let A^* denote the set of all strings on the alphabet A , and write A^+ for $A^* - \{\lambda\}$. For $a \in A$, we let a^ω denote the word with underlying order the ordinal ω whose value at each point is the letter a ; similarly, $a^{\omega^{op}}$ is the word with underlying order ω^{op} whose value at each point is a .

Definition 2.3. The **product** of u, v , written uv , is $w(a/u, b/v)$, where $w = ab$. The **right omega power** of the word u , written u^ω , is $w(a/u)$, where $w = a^\omega$. The **left omega power**, written $u^{\omega^{op}}$, of a word u is $w(a/u)$ where $w = a^{\omega^{op}}$.

The reason for the terminology “right” and “left” omega power is the following. The word u^ω is the initial solution (in the sense of [Cour78]) in the class of words of the equation in the variable x ,

$$x = ux.$$

Since x appears to the right of u , the result is called the right omega power of u . Similarly, $u^{\omega^{op}}$ is the initial solution of

$$x = xu.$$

Suppose that (P, \leq_P) and (Q, \leq_Q) are linear orders. When $L = \{1, 2\}$ is the two-element linear order, and $K_1 = P$, $K_2 = Q$, the generalized sum $\sum_{x \in L} K_x$ is written $P + Q$. If $L = \omega$ and $K_i = P$, for each $i \in \omega$, the linear order $\sum_{i \in \omega} K_i$ is written $P \times \omega$; if $L = \omega^{op}$, and $P_i = P$, for each $i < 0$, the linear order $\sum_{i \in \omega^{op}} P_i$, is written $P \times \omega^{op}$. (Rosenstein [Ro82] uses the notations $P \cdot \omega$ and $P \cdot \omega^{op}$, but we prefer the indicated notation, since \cdot is used for other purposes.)

Corollary 2.1. Let u, v be words. The underlying order of uv is $L_u + L_v$; the underlying order of u^ω is $L_u \times \omega$, and the underlying order of $u^{\omega^{op}}$ is $L_u \times \omega^{op}$. \square

We need the following fact, proved in [Ro82], Theorems 7.11 and 7.13.

Lemma 2.1. For any nonempty finite set $A = \{a_1, \dots, a_n\}$ there is a word (P, \leq, ρ_n) on A , whose underlying order is infinite, with no least or greatest element, which has the following property. For any $x < y$ in P , and for each $i \in [n]$, there is some $z \in P$ with $x < z < y$ and $\rho_n(z) = a_i$. Further, any two such words are isomorphic.

The underlying order of the word ρ_n is isomorphic to the rationals, \mathbb{Q} . One possible concrete description of ρ_n is the following: the underlying set is the set of rationals of the form $p/(n+1)^k + i/(n+1)^{k+1}$, for all positive integers k , all integers p and all positive integers $i \in [n]$. Any rational has at most one such representation. We let the letter a_i be assigned to $p/(n+1)^k + i/(n+1)^{k+1}$.

Another description of ρ_n is given in Example 3.2 below.

Now, we define the **shuffle** of the finite sequence (u_1, \dots, u_n) of words by:

$$[u_1, \dots, u_n]^\eta := \rho_n(a_1/u_1, \dots, a_n/u_n). \quad (1)$$

In particular, we may write

$$\rho_n = [a_1, \dots, a_n]^\eta.$$

Using the word ρ_n , we may define the shuffle of linear orders.

Definition 2.4. Let (P, \leq, ρ_n) be the word in Lemma 2.1. Suppose that L_i is a linear order, for $i \in [n]$. Then the **shuffle** of the linear orders L_1, \dots, L_n is $\sum_{x \in P} K_x$, where $K_x = L_{\rho_n(x)}$. We denote this linear order by $[L_1, \dots, L_n]^\eta$.

Proposition 2.1. If L_i is the underlying order of the word u_i , $i \in [n]$, then the underlying order of the word $[u_1, \dots, u_n]^\eta$ is $[L_1, \dots, L_n]^\eta$. \square

Definition 2.5. We call a linear order (L, \leq) **quasi dense** if there is an injective, order-preserving function $\mathbb{Q} \rightarrow L$. A linear order (L, \leq) is **scattered** if it is not quasi dense. A linear order (L, \leq) is **dense** if whenever $x < y$ in L , there is some $z \in L$ with $x < z < y$. (Thus, if (L, \leq) is dense and L has at least two elements, then (L, \leq) is quasi dense.) A word (L_u, \leq_u, u, A) is scattered (or quasi dense, or dense, respectively) when its underlying linear order is.

Note that any word isomorphic to a scattered (or dense, or quasi dense) word is also scattered (or dense, or quasi dense, respectively). For example, all of the nonempty linear orders $[L_1, \dots, L_k]^\eta$ are quasi dense.

Definition 2.6. The **regular words** on the alphabet A are those in the least class of words containing the single letter words $a_i \in A$, closed under the operations of product, right and left omega power, and shuffle. A **regular expression** over A is either a letter in A , or an expression of the form

$$uv, u^\omega, u^{\omega^{op}}, [u_1, \dots, u_k]^\eta,$$

where u, v, u_j are regular expressions, for $j \in [k]$. The word denoted by a regular expression is defined in the obvious way. The **size** $|w|$ of a regular expression w is defined by induction as follows:

$$\begin{aligned} |a_i| &:= 1, & a_i \in A \\ |uv| &:= 1 + |u| + |v| \\ |u^\omega| = |u^{\omega^{op}}| &:= 1 + |u| \\ |[u_1, \dots, u_n]^\eta| &:= 1 + \sum_{i=1}^n |u_i|. \end{aligned}$$

Excluding the empty word from the class of regular words makes many formulations simpler. Note that we are not concerned with regular *sets* of linear words - only one word at a time. Sets of labeled linear orders accepted by generalized finite automata have been considered recently in [BruyCar, BruyCar2, Car].

The regular expressions just defined were used in Heilbrunner [Heil80], extending those used by Courcelle [Cour78]. Neither Courcelle nor Heilbrunner use the term “regular word”. Courcelle showed that any word (“arrangement” is the term he used) is, up to isomorphism, the frontier of a leaf labeled complete binary tree, i.e., a binary tree whose non-leaf nodes have both a left and right successor. (Sometimes these trees are called “full binary trees”.) He considered solving equations in the category of words; the initial solution to equations is the frontier of a regular tree. Courcelle then described those systems that determine the scattered regular words (see below), and introduced what he called regular expressions

to denote these words (no shuffle operation is involved). Heilbrunner gave an algorithm to solve all such equations, and introduced the regular expressions above to denote the solutions. We have taken the liberty of giving the name “regular word” to a word denoted by the wider class of regular expressions.

Let G denote the set of words

$$ab, a^\omega, a^{\omega^{op}}, \rho_1, \rho_2, \dots, \rho_n, \dots$$

For an alphabet A , let G_A denote the least class of words containing $G \cup A$ closed under substitution. Let $W(A)$ denote the set of words containing only letters in the alphabet A .

Proposition 2.2. The regular words on the alphabet A are precisely the words in $G_A \cap W(A)$.

Proof. It is enough to show that the regular words are closed under substitution. This may be proved by induction on the structure of the regular expression used to denote the word. \square

Proposition 2.3. The underlying linear orders of the regular words is the least class **RL** of linear orders containing the singleton **1** and closed under sum, shuffle, and $P \mapsto P \times \omega$ and $P \mapsto P \times \omega^{op}$. \square

Definition 2.7. We call a linear order **regular** if it is isomorphic to an order in the class **RL**.

Remark 2.1. There is a logical connection, described in [Ro82], Theorem 7.20, between all countable linear orders and the orders in **RL**.

A **prefix code** C (or “code” for short) is a *nonempty* collection of strings on $\{0, 1\}$ such that no string in C is a proper prefix of any other string in C . A prefix code is **complete** if for any string u not in C , $C \cup \{u\}$ is not a prefix code. A code is **regular** if it is a regular subset of $\{0, 1\}^*$.

It is well known that one may represent the vertices of any binary tree by a prefix-closed subset of the strings on the alphabet $\{0, 1\}$. A collection C of strings is a (complete) prefix code iff C is the set of leaves of a (complete, or ‘full’) binary tree.

The next fact was pointed out by Heilbrunner [Heil80], who showed that the regular words (together with the empty word) on A are the components to initial solutions of systems of fixed point equations of the form

$$\begin{aligned} x_1 &= u_1 \\ &\vdots \\ x_n &= u_n, \end{aligned}$$

where u_i are words in $(A \cup \{x_1, \dots, x_n\})^*$, such that no u_i is x_j , for any j .

Recall that a leaf-labeled regular tree over the alphabet A is a tree, whose leaves are labeled by letters in A , which has a finite number of subtrees. A tree is “locally finite” if for each node v , there is some path from v to a leaf. The frontier of a tree t is the word on A whose underlying linear order is the set of leaves of t ordered lexicographically, labeled as in the tree. From now on, a tree is assumed to have its leaves labeled by letters in A . The following proposition is fundamental.

Proposition 2.4. [Heil80] A word u on the alphabet A is regular iff u is the frontier of a regular locally finite, binary tree t whose leaves C form a regular complete prefix code. \square

We will need a slight extension of this result.

Proposition 2.5. A word u on the alphabet A is regular iff u is the frontier of a regular locally finite, binary tree t whose leaves C form a regular prefix code, not necessarily complete.

Proof. We need prove only one direction. Suppose that u is the frontier of a regular tree t whose interior nodes have one or two successors. We need to find a regular, locally finite, complete binary tree whose frontier is isomorphic to u .

We will describe the desired tree as the unfolding of a finite directed graph G . The graph is obtained in two steps. First, the tree t determines a finite edge labeled, directed graph $H = (V, E)$, whose nodes are all subtrees of t . Say that these trees are e_1, \dots, e_n , with roots r_1, \dots, r_n . If r_i has a left successor r_l , then there is an edge $e_i \rightarrow e_l$ labeled 0 in H ; similarly if r_i has a right successor. If r_i is a leaf, it has outdegree 0 in H . Since the tree t is locally finite, there is no cycle in H containing only vertices of outdegree 1. Now let \sim be the least equivalence on the vertices of H such that $u \sim v$ if there is a path $u \rightsquigarrow v$ in H such that each vertex on the path has outdegree one, *except perhaps the last*. Then, if $[v]$ denotes the \sim -equivalence class of v , $[v]$ contains exactly one node of degree either 2 or 0. Indeed, let $v = v_0, v_1, \dots, v_s$ be a shortest path in G from v to a vertex v_s of degree 0. If each vertex v_i , for each $i < s$, has degree 1, then $v_s \in [v]$. Otherwise, there is some vertex v_i on the path of degree 2, so $v_i \in [v]$. Thus, for each vertex v in H there is vertex v' of outdegree either 0 or 2 such that $v \sim v'$. Also, if v and v' have degree 0 or 2 and $v \sim v'$, then $v = v'$. Last, if v is a vertex of outdegree 0, then $[v]$ is labeled by the label of the leaf v .

Now, let G be the directed graph whose nodes are the \sim -equivalence classes $[v]$ of vertices in H . If $v \sim v'$ and v' has outdegree 2, then there is an edge $[v] \rightarrow [w]$ in G labeled 0 (or 1, respectively), if there is an edge $v' \rightarrow y$ labeled 0 (resp. 1) in H and $[y] = [w]$. Those equivalence classes $[v]$ containing a vertex of outdegree 0 are labeled by the label of the leaf they contain.

Now G is an edge labeled directed graph in which each vertex has either outdegree 2 or 0; those of outdegree 0 are labeled by letters. The unfolding of G is a locally finite, regular, complete binary tree, whose frontier is isomorphic to the frontier of t . \square

3. Language theoretic characterizations

We now turn to language theoretic characterizations of the regular words. Characterizations of scattered regular words will be considered in the following section. We observe (in Proposition 3.2) that regular words on an n -letter alphabet are determined up to isomorphism by a partition of a regular complete prefix code into n pairwise disjoint regular prefix codes. We prove, more generally, (in Proposition 3.3) that a regular word on an n -letter alphabet is determined up to isomorphism by any n pairwise disjoint regular subsets of $\{0, 1\}^*$ (whose union is nonempty). Then, in Proposition 3.5, we show that there is an algorithm to produce, for each regular expression w , an “ A -automaton” $M(w)$ accepting a

complete prefix code which determines a word isomorphic to the word denoted by w . The size of $M(w)$ is proportional to the number of symbols in w .

Any subset X of $\{0, 1\}^*$ is linearly ordered by the **lexicographic order**: for $u, v \in X$,

$$u \leq_\ell v \iff \exists u_1, u_2, w ((v = uw) \text{ or } (u = w0u_1 \text{ and } v = w1u_2)).$$

Example 3.1. Let W be the regular set $W = 1^*0$. Then, the lexicographic order on W is isomorphic to ω , via the map

$$1^n0 \mapsto n.$$

Similarly, ω^{op} is isomorphic to $(0^*1, \leq_\ell)$.

Remark 3.1. Suppose that $B_n = \{b_1, \dots, b_n\}$ is an n -element alphabet, ordered by $b_1 < b_2 < \dots < b_n$. Then the strings on B_n are also linearly ordered by the lexicographic order:

$$u \leq_\ell v \iff \exists u_1, u_2, w ((v = uw) \text{ or } (u = wb_iu_1 \text{ and } v = wb_ju_2 \text{ and } i < j)).$$

The lexicographic order on $\{0, 1\}^*$ has the universal property that every countable linear order embeds in it. In fact, a stronger result holds.

Proposition 3.1. For any nonempty, countable linear order (L, \leq) there is a complete prefix code $P \subseteq \{0, 1\}^*$ such that (L, \leq) is isomorphic to (P, \leq_ℓ) .

Proof. Indeed, there is such a subset isomorphic to the usual ordering of the rational numbers. One such set is

$$(11 + 0)^*10. \tag{2}$$

(See [Thom86].) Furthermore, there is an order preserving embedding of any countable linear order into the rationals. Since the strings in $(11 + 0)^*10$ form a complete prefix code, we see that any countable linear order is isomorphic to the lexicographic order on some prefix code. Courcelle [Cour78] proved that for any prefix code P and any nonempty word $u = (P, \leq_\ell, u)$ there is a complete prefix code C and a word $w = (C, \leq_\ell, w)$ such that u is isomorphic to w . (The argument given in Proposition 2.5 can be extended to give a different proof of this result.) \square

Proposition 3.2. Suppose that C is a (complete) prefix code and \leq_ℓ is the lexicographic order. Suppose that C is partitioned into R_1, \dots, R_n , where each set $R_i, i \in [n]$, is regular. Then there is a regular word $u(R_1, \dots, R_n) = (C, \leq_\ell, u)$, such that $u(x) = a_i$ iff $x \in R_i, i \in [n]$. Conversely, for any regular word w on A , there is a family R_1, \dots, R_n of disjoint regular subsets of $\{0, 1\}^*$ such that $R_1 \cup \dots \cup R_n$ is a complete prefix code and w is isomorphic to $u(R_1, \dots, R_n)$.

Proof. Both directions follow immediately from the fact that, up to isomorphism, regular words are frontiers of (complete) locally finite, regular trees. See Propositions 2.4 and 2.5. The set of words labeling leaves with a particular label is a regular subset of $\{0, 1\}^*$. In fact, it follows from Theorem 4.11.1

of [Cour83] that a locally finite binary tree t over A is regular iff, for each $a \in A$ the set of binary words which are leaves of t labeled a is regular. \square

We note that prefix codes are not necessary to obtain a regular word. In fact, we have the following.

Proposition 3.3. A word w is regular iff there is a family $R_i, i \in [n]$, of pairwise disjoint regular subsets of $\{0, 1\}^*$ whose union is nonempty such that w is isomorphic to the word $(\bigcup_{i \in [n]} R_i, \leq_\ell, u)$, such that $u(x) = a_i \iff x \in R_i, i \in [n]$.

Proof. We need prove only that any n pairwise disjoint regular sets whose union is nonempty determine a regular word as indicated.

Let $L = \bigcup_{i \in [n]} R_i$. The set L is regular, but is not necessarily a prefix code. Thus, we first replace L by a prefix free set of strings on the ordered alphabet $B_3 = \{-1, 0, 1\}$. We then apply Proposition 3.2.

Define the set \hat{L} as $L(-1)$, the set of words obtained by putting -1 on the right of each word in L . Then \hat{L} is a prefix code and a regular subset of the strings on B_3 . Now if x and $xy \in L$, then $x(-1) \leq_\ell xy(-1)$ in \hat{L} . (Recall Remark 3.1.) Also, if $x0y$ and $x1z$ are in L , then $x0y(-1) \leq_\ell x1z(-1)$ in \hat{L} . Thus $x \mapsto x(-1)$ is an order isomorphism $(L, \leq_\ell) \rightarrow (\hat{L}, \leq_\ell)$. Now define the function $w : \hat{L} \rightarrow A$ as follows.

$$w(x(-1)) := u(x).$$

Thus, the words w and u are isomorphic.

Now we let $\varphi : \{-1, 0, 1\}^+ \rightarrow \{0, 1\}^+$ be the unique semigroup morphism determined by:

$$\begin{aligned} \varphi(-1) &:= 00 \\ \varphi(0) &:= 01 \\ \varphi(1) &:= 10. \end{aligned}$$

Note that $\varphi(x) <_\ell \varphi(y)$ when $x < y \in \{-1, 0, 1\}$. Since both sets \hat{L} and $\{\varphi(-1), \varphi(0), \varphi(1)\}$ are prefix free, it follows that $\varphi(\hat{L})$ is a prefix code and $\varphi : (\hat{L}, \leq_\ell) \rightarrow (\varphi(\hat{L}), \leq_\ell)$ is an order isomorphism. Thus, the word v is isomorphic to u , where

$$v(\varphi(x)) := w(x), \quad x \in \hat{L}.$$

Since $\varphi(\hat{L})$ is regular, we have shown that u is isomorphic to a regular word whose underlying order is the lexicographic ordering of a regular prefix code. Thus, by Proposition 2.5, u is isomorphic to a regular word whose underlying order is a regular, complete prefix code. \square

Remark 3.2. Essentially the same argument shows the following. A word w is regular iff there is a family $R_i, i \in [n]$, of pairwise disjoint regular subsets of strings on an ordered alphabet b_1, \dots, b_k , with $b_1 < \dots < b_k$, whose union is nonempty such that w is isomorphic to the word $(\bigcup_{i \in [n]} R_i, \leq_\ell, u)$, where $u(x) = a_i \iff x \in R_i, i \in [n]$.

Before introducing A -automata, we review some terminology. We will need automata on only the binary input alphabet. A deterministic, finite automaton (DFA) with alphabet $\{0, 1\}$, is a 4-tuple $M = (Q, q_0, \delta, F)$, where Q is a finite set of states, $\delta : Q \times \{0, 1\} \rightarrow Q$ is the transition function, $q_0 \in Q$ is the initial state and $F \subseteq Q$ is the set of final states. We immediately extend the transition function to a function $\delta : Q \times \{0, 1\}^* \rightarrow Q$ in the usual way. M is accessible if for each state q there is some string $x \in \{0, 1\}^*$ such that $\delta(q_0, x) = q$; a state q is coaccessible if there is some string $x \in \{0, 1\}^*$ such that $\delta(q, x)$ is final. The behavior of a state q is the set of strings x such that $\delta(q, x) \in F$. The language $\mathcal{L}(M)$ accepted or determined by the DFA M is the behavior of the initial state.

Definition 3.1. For an n -element alphabet A , an A -automaton

$$M = (Q, q_0, \delta, F)$$

is a DFA with n final states, labeled f_1, \dots, f_n , at least one of which is accessible, and a sink state \perp such that $\delta(f_i, x) = \perp = \delta(\perp, x)$, for $x \in \{0, 1\}$, $i \in [n]$. An A -automaton M determines the word (L, \leq_ℓ, μ_M) where $L = \mathcal{L}(M)$, and, for $x \in L$, $\mu_M(x) = a_i$ iff $\delta(q_0, x) = f_i$.

We will show how to produce, for each regular word u , an A -automaton M such that μ_M is isomorphic to u .

Proposition 3.4. For any A -automaton M , $\mathcal{L}(M)$ is a (regular) prefix code and μ_M is a regular word.

Proof. The first claim is trivial and the second follows from Proposition 3.3. □

Example 3.2. Let A be the k letter alphabet $\{a_1, \dots, a_k\}$. We define an A -automaton determining the word ρ_k (see Lemma 2.1).

The (non sink) states are divided into three groups: c_0, \dots, c_{k-1} , d_0, \dots, d_{k-1} and the final states f_1, \dots, f_k . The initial state is c_0 . The states c_i are used to count the number of 0's modulo k ; the states d_i are intermediate states. The transition function is defined as follows:

$$\begin{aligned} \delta(c_i, 0) &= c_{i+1}, & 0 \leq i < k-1 \\ \delta(c_{k-1}, 0) &= c_0 \\ \delta(c_i, 1) &= d_i, & 0 \leq i < k \\ \delta(d_i, 1) &= c_i, & 0 \leq i < k \\ \delta(d_i, 0) &= f_{i+1}, & 0 \leq i < k. \end{aligned}$$

For a string x , let $|x|_0$ denote the number of 0's in x . Now let $B = (0 + 11)^*$, and for $0 \leq i < n$, let B_i be defined by:

$$x \in B_i \iff x \in B \text{ and } |x|_0 \equiv i \pmod{n}.$$

Thus B is the disjoint union of the sets B_i , $i \in [n]$.

Claim 1: For a string $x \in \{0, 1\}^*$, and integer $0 \leq i < n$, $\delta(c_0, x) = c_i$ iff $x \in B_i$.

Figure 1. Most of the automaton for ρ_4

Proof. Indeed, $\delta(c_i, 11) = c_i$, so that only the number of 0's determines the resulting state.

Claim 2: For any string $x \in \{0, 1\}^*$ and integer $0 \leq i < n$, $\delta(c_0, x) = f_i$ iff $x = y10$ for some $y \in B_i$.

Now let $C = B10$. Then, (C, \leq_ℓ) is a dense linear order, with no first or last (see (2) above).

Claim 3: For $x <_\ell y$ in C , and any $0 \leq i < n$ there is some z in C with $x <_\ell z <_\ell y$ and $\delta(c_0, z) = f_i$.

There are two cases. First, suppose $x = u(10)$ and $y = u(11)^r(10)$, where $u \in B$ and $r > 0$. Then, for any $j > 0$, the word $z_j = (u11)0^j(10)$ belongs to $B10$ and $x < z_j < y$. For any $0 \leq i < n$, we can find an appropriate j such that $\delta(c_0, u110^j) = c_i$, so that $\delta(c_0, z_j) = f_i$.

The second case is $x = u0v(10)$ and $y = u1w(10)$, where $u0v$ and $u1w$ belong to B . Then, for any $j > 0$, if $z_j = u1w0^j10$, then

$$x < z_j < y,$$

and $u1w0^j \in B10$. For any choice of $i \in [n]$ there is an appropriate value of j such that $\delta(c_0, z_j) = f_i$.

This completes the proof that the automaton determines ρ_k . □

We now give a direct construction which, applied to a regular expression w , produces an A -automaton

M such that $u = \mu_M$ is isomorphic to the word denoted by w , and such that the underlying linear order of u is the lexicographic order on a complete prefix code.

We make a preliminary observation. The easy proof is only sketched.

Lemma 3.1. Let M_0 be a B -automaton on the k -letter alphabet B . Let M_1, \dots, M_k be A -automata. Then there is an A -automaton

$$M = M_0 \cdot \langle M_1, \dots, M_k \rangle$$

which accepts the strings yz such that y labels a path from the initial state of M_0 to some final state f_j of M_0 , and z labels a path from the initial state of M_j to a final state of M_j . The word determined by M is

$$v_0(a_1/v_1, \dots, a_k/v_k),$$

where $v_j = \mu_{M_j}$, $j = 0, 1, \dots, k$. Further, if the languages $\mathcal{L}(M_j)$, for $j = 0, 1, \dots, n$, are all complete prefix codes, so is $\mathcal{L}(M)$.

Proof. Let $\langle M_1, \dots, M_k \rangle$ be obtained from the disjoint union of the states of the n automata M_i , $i \in [k]$, by identifying only the corresponding final states in each, so now state f_1 in M_1 becomes the same state as f_1 in M_2 , etc. We also identify all sink states. The automaton $\langle M_1, \dots, M_k \rangle$ has k initial states. For $i \in [k]$, the i -th initial state is the initial state of M_i . Then, let $M = M_0 \cdot \langle M_1, \dots, M_k \rangle$ be obtained from the disjoint union of the states in M_0 and $\langle M_1, \dots, M_k \rangle$ by identifying the sink states and the i -th final state of M_0 with the initial state of M_i . Then, since no nonempty word labels a path starting from any final state in M_i , $i = 1, \dots, k$, a string x labels a path in M from the initial state of M_0 to the final state f_i iff $x = yz$ where y labels a path in M_0 from the initial state of M_0 to some accessible final state, say f_j in M_0 , and z labels a path in M_j from the initial state of M_j to the final state f_i in M_j . We omit the remaining routine verification. \square

Before stating the next Proposition, we define the **size**, $|M|$, of an A -automaton M as the number of **non sink states** in M . The following fact is immediate by construction.

Lemma 3.2. If M_0 is a B automaton and M_i , $i \in [k]$ are A -automata, where B has k letters and M_i , for $i \in [k]$ has n letters, then

$$|M_0 \cdot \langle M_1, \dots, M_k \rangle| = \sum_{j=0}^k |M_j| - n(k-1) - k.$$

Proof. We subtract $n(k-1)$ since we are identifying the corresponding final states of each of the k automata M_1, \dots, M_k ; and we subtract k since we identify the exit state f_i of M_0 with the initial state of M_i , $i \in [k]$. \square

Proposition 3.5. There is an algorithm which, given a regular expression w on the n -letter alphabet A , produces an A -automaton $M(w)$ such that the set of strings x accepted by $M(w)$ is a complete prefix code C_w , and the word denoted by w is isomorphic to $\mu_{M(w)}$. Further,

$$|M(w)| \leq (n+1)|w|.$$

Proof. By Lemma 3.1, we need only show how to find A -automata for the basic words $a, ab, a^\omega, a^{\omega^{op}}$ and $\rho_k, k \geq 1$. We have already given the automaton for ρ_k on a k -letter alphabet. It has size $3k$.

For a letter $a = a_i \in \{a_1, \dots, a_n\}$, let $M(a_i)$ have a sink and final states f_1, \dots, f_n , with f_i as the initial state. The set of strings accepted by $M(a_i)$ is the set consisting of the empty string, and it is labeled a_i . Thus, $|M(a_i)| = n$.

Now, if $a = a_1$, and $b = a_2$, an $\{a_1, a_2\}$ -automaton for ab has an initial state q_0 aside from the final and sink states. The transition function is:

$$\begin{aligned}\delta(q_0, 0) &= f_1 \\ \delta(q_0, 1) &= f_2.\end{aligned}$$

Its domain is the complete prefix code $\{0, 1\}$.

We construct an $\{a\}$ automaton for a^ω . There is one state aside from the final state f_1 and the sink, namely the initial state. The transition function is defined by:

$$\begin{aligned}\delta(q_0, 1) &= q_0 \\ \delta(q_0, 0) &= f_1.\end{aligned}$$

Then the strings accepted by this automaton are those in 1^*0 , and $(1^*0, \leq_\ell)$ is isomorphic to ω , via the map

$$1^n 0 \mapsto n.$$

There is a similar construction for $a^{\omega^{op}}$.

As for the sizes of the resulting A -automata, our construction gives:

$$\begin{aligned}|M(a_i)| &= n \\ |M(uv)| &= 1 + |M(u)| + |M(v)| - n \\ |M(u^\omega)| &= 1 + |M(u)| \\ |M(u^{\omega^{op}})| &= 1 + |M(u)| \\ |M([u_1, \dots, u_k]^\eta)| &= 3k + \sum_{i=1}^k |M(u_i)| - n(k-1) - k \\ &= 2k - n(k-1) + \sum_{i=1}^k |M(u_i)|,\end{aligned}\tag{3}$$

by Lemma 3.2.

Now, we prove by induction on the regular expression w , that $|M(w)| \leq (n+1)|w|$. In fact, it is easier to prove

$$|M(w)| \leq (n+1)|w| - 1.$$

When $w = a_i \in A$, $|M(a_i)| = n = (n + 1)|a_i| - 1$. If $w = uv$, then

$$\begin{aligned} |M(uv)| &= 1 + |M(u)| + |M(v)| - n \\ &\leq 1 + (n + 1)(|u| + |v|) - 2 - n \end{aligned}$$

by the induction hypothesis,

$$\begin{aligned} &< (n + 1)(|u| + |v| + 1) - 1 \\ &= (n + 1)|uv| - 1. \end{aligned}$$

When $w = u^\omega$,

$$\begin{aligned} |M(u^\omega)| &= 1 + |M(u)| \\ &\leq 1 + (n + 1)|u| - 1 \\ &\leq (n + 1)(|u| + 1) - 1 \\ &= (n + 1)|u^\omega| - 1. \end{aligned}$$

Similarly, if $w = u^{\omega^{op}}$. Last, if $w = [u_1, \dots, u_k]^\eta$, where u_i are regular expressions on the n -letter alphabet, by (3)

$$\begin{aligned} |M([u_1, \dots, u_k]^\eta)| &= 2k - n(k - 1) + \sum_{i=1}^k |M(u_i)| \\ &\leq 2k - (k - 1) + \sum_{i=1}^k |M(u_i)|, \end{aligned}$$

since $n \geq 1$,

$$\leq k + 1 + \sum_{i=1}^k ((n + 1)|u_i| - 1),$$

by induction,

$$\begin{aligned} &= 1 + (n + 1) \sum_{i=1}^k |u_i| \\ &\leq (n + 1)(1 + \sum_{i=1}^k |u_i|) - 1 \\ &= (n + 1)|[u_1, \dots, u_k]^\eta| - 1. \end{aligned}$$

This completes the proof. □

4. Scattered regular words

In this section, we give several characterizations of the scattered regular words. In the next section, we give polynomial time algorithms to decide when the regular word determined by a given A -automaton is scattered, and if not, whether it is dense.

Proposition 4.1. A linear order (L, \leq) is quasi dense iff there is an order embedding $(\{0, 1\}^*, \leq_\ell) \rightarrow (L, \leq)$.

Proof. By Proposition 3.1. □

Since every nonempty shuffle is quasi dense, we have

Proposition 4.2. A regular word on the alphabet A is scattered iff it belongs to the least class of words which contains the singletons $a \in A$, and is closed under product, $u, v \mapsto uv$, and the operations $u \mapsto u^\omega$ and $u \mapsto u^{\omega^{op}}$. The scattered regular orders are those isomorphic to the orders in the least class of linear orders which contains the singleton $\mathbf{1}$, closed under sum and the operations $P \mapsto P \times \omega$ and $P \mapsto P \times \omega^{op}$.

Courcelle [Cour78] characterized the scattered regular words as the solutions to a “quasi rational” system of fixpoint equations.

Since the underlying order of any regular word is isomorphic to the lexicographic order on a prefix code, the following question suggests itself:

Which are the regular (complete) prefix codes C such that (C, \leq_ℓ) is scattered?

We call a DFA M a **monotone** DFA if there is a partial order \leq on the state set Q such that the initial state is minimum, and if $q' = \delta(q, a)$, then $q \leq q'$. Thus, the only loops possible in a monotone DFA are self loops. We call a subset of $\{0, 1\}^*$ **monotone** if it is a regular set accepted by some monotone DFA. (Thus, by definition, any monotone set is regular.)

For a detailed study and various characterizations of monotone automata, their underlying semiautomata, and their languages, see the books [Pin86, Eil76] and the paper [Brzo80]. In the last cited paper, monotone automata are called “partially ordered automata”. This paper contains, among other results, a proof of the fact that an automaton is “partially ordered” iff its transition monoid is R-trivial.

Lemma 4.1. A subset of $\{0, 1\}^*$ is monotone iff its minimal automaton is monotone.

Proof. Indeed, any morphic image of a monotone DFA is also monotone. □

Proposition 4.3. [Brzo80] An accessible DFA $M = (Q, q_0, \delta, F)$ on the input alphabet X is monotone iff the binary relation \sqsubseteq on Q is antisymmetric, where $q \sqsubseteq q'$ iff there is some string $u \in X^*$ such that $\delta(q, u) = q'$. Thus, an accessible DFA is not monotone iff it has at least two states $q \neq q'$ such that $\delta(q, x) = q'$ and $\delta(q', y) = q$ for some strings, $x, y \in X^*$. □

We will show that any scattered regular linear order is isomorphic to a monotone subset of $\{0, 1\}^*$ which is a complete prefix code. We break the proof into several easy parts. We sometimes identify a subset P of $\{0, 1\}^*$ with the linearly ordered set (P, \leq_ℓ) . First, an easy observation.

Lemma 4.2. Suppose that $P \subseteq \{0, 1\}^*$. Then

1. $(1^*0P, \leq_\ell)$ is isomorphic to $P \times \omega$.
2. $(0^*1P, \leq_\ell)$ is isomorphic to $P \times \omega^{op}$.
3. Suppose $P, Q \subseteq \{0, 1\}^*$. Then $(0P \cup 1Q, \leq_\ell)$ is isomorphic to $(P, \leq_\ell) + (Q, \leq_\ell)$.

Proposition 4.4. Let L be a scattered regular linear order. Then L is isomorphic to a monotone subset of $\{0, 1\}^*$ which is a complete prefix code.

Proof sketch. Clearly, $\mathbf{1}$ has the required property, and one shows that it is preserved by the operations $P \mapsto P \times \omega$, $P \mapsto P \times \omega^{op}$ and $P, Q \mapsto (P + Q)$.

For example, assume that P is a monotone, complete prefix code. Then, by Lemma 4.2, 1^*0P , ordered lexicographically, is isomorphic to $P \times \omega$. If M is a monotone DFA accepting P , then a monotone DFA M' accepting 1^*0P is quite similar to the A -automaton $M_0 \cdot \langle M \rangle$, (recall the construction in Lemma 3.1) where M_0 is the automaton above accepting ω . By interchanging the 0's and 1's, we get a monotone DFA accepting 0^*1P , which, when ordered lexicographically, is isomorphic to $P \times \omega^{op}$. \square

There is a slightly stronger converse.

Proposition 4.5. Suppose that C is a monotone prefix code (not necessarily complete). Then C , ordered lexicographically, is a scattered regular linear order.

Proof. Let M be the minimal DFA accepting C . We assume the states of M are $\{q_1, q_2, \dots, q_n\}$, and assume the initial state is q_1 . We use induction on n . If $n = 1$, then M accepts either no strings or all strings. Neither is a prefix code. If $n = 2$, C must consist of just the empty string, and the initial state is the only final state. Otherwise, C cannot be a prefix code. The codes accepted by a 3 state minimal, monotone DFA are:

$$\{0\}, \{1\}, \{0, 1\}, 0^*1, 1^*0.$$

The corresponding orders are isomorphic to $\mathbf{1}, \mathbf{1}, \mathbf{1} + \mathbf{1}, \omega^{op}, \omega$, respectively. The proof is completed by the following observations: if C is a prefix code and $C = 0L_1 \cup 1L_2$, then at least one of L_1, L_2 is nonempty, and if nonempty, both L_1 and L_2 are prefix codes. If $C = 0^*1L$, then L is a (complete) prefix code; similarly if $C = 1^*0L$. Now assume $n > 3$. Either the initial state has a self loop, or not. If not, then $C = 0L_1 \cup 1L_2$, where L_1 is the behavior of the state $\delta(q_1, 0)$ and L_2 is the behavior of the state $\delta(q_1, 1)$. Hence, by induction, if both L_1, L_2 are nonempty, both are scattered regular linear orders, and C determines a linear order isomorphic to their sum. If L_1 is empty, say, then C determines a linear

order isomorphic to the one determined by L_2 , which is a scattered regular linear order, by induction. If both L_1, L_2 are empty, C is not a code.

If the initial state has a self loop on the letter 1, say, then $C = 1^*0L$, where L is the behavior of the state $\delta(q_1, 0)$. If L is empty, so is C , so C cannot be a code. Otherwise, L determines an scattered regular linear order, by induction, and the linear order determined by C is isomorphic to $L \times \omega$. Similarly, if the initial state has a self loop on the letter 0, C will determine a linear order isomorphic to $L \times \omega^{op}$. The proof is complete. \square

Corollary 4.1. A linear order is scattered regular linear order iff it is isomorphic to the lexicographic ordering of a monotone, (complete) prefix code. \square

Corollary 4.2. A word w on the alphabet A is regular and scattered iff there is a monotone A -automaton $M = (Q, q_0, \delta, F)$ accepting a (complete) prefix code such that w is isomorphic to μ_M .

The word ‘‘isomorphic’’ in Corollary 4.2 (and elsewhere) is crucial. A regular complete prefix code C may not be monotone, but nonetheless (C, \leq_ℓ) may be scattered. Consider the set $C = (01)^*(00 + 1)$, for example. The linear order (C, \leq_ℓ) is isomorphic to $\omega + \omega^{op}$, and C is not monotone.

We summarize the above facts.

Theorem 4.1. For a word (L_w, \leq_w, w) on the alphabet A , the following are equivalent.

1. w belongs to the least class of words containing the single letters $a \in A$, closed under product, $u, v \mapsto uv$, and the operations $u \mapsto u^\omega$ and $u \mapsto u^{\omega^{op}}$.
2. w is regular and L_w is a scattered (regular) linear order.
3. w is isomorphic to a regular word u , where L_u is a monotone, (complete) prefix code.
4. w is isomorphic to a regular word u , where (L_u, \leq_ℓ) is scattered and a regular (complete) prefix code.
5. w is isomorphic to a word $u(R_1, \dots, R_n)$, where the sets R_i are regular, pairwise disjoint, and $\bigcup_{i \in [n]} R_i$ is a monotone (complete) prefix code.

\square

We recall Cantor’s normal form theorem ([Ro82], page 61): any ordinal α may be written uniquely as

$$\alpha = \omega^{\gamma_1} \times n_1 + \dots + \omega^{\gamma_k} \times n_k,$$

for some $\gamma_1 > \dots > \gamma_k$ and $0 < n_i < \omega$, all $i \in [k]$.

Recall the definition of the class **RL**, Definition 2.7. From Theorem 4.1 and Cantor’s normal form theorem we obtain the following Corollary.

Corollary 4.3. An ordinal α is in **RL** iff $\alpha < \omega^\omega$. \square

5. Some algorithms

The DFA's that accept prefix codes are easily characterized. The proof of the following fact is an easy exercise.

Proposition 5.1. An accessible DFA $M = (Q, q_0, \delta, F)$ accepts a prefix code iff for each final state q , there is no nonempty string $x \in \{0, 1\}^*$ such that $\delta(q, x)$ is also final. An accessible DFA $M = (Q, q_0, \delta, F)$ accepts a complete prefix code iff it accepts a prefix code and for each coaccessible state q , either q is final or both $\delta(q, 0)$ and $\delta(q, 1)$ are coaccessible. \square

Thus, every A -automaton accepts a prefix code.

Corollary 5.1. There is a polynomial time algorithm to determine, given a DFA M on the alphabet $\{0, 1\}$,

1. whether M accepts a monotone, (complete) prefix code, say P , and
2. if it does, whether (P, \leq_ℓ) is well ordered.

Proof outline. First, in polynomial time, find the minimal DFA \overline{M} equivalent to M , and then check whether \overline{M} has nontrivial cycles. Then, check whether \overline{M} satisfies the conditions in Proposition 5.1. If it does, it is not hard to see that the lexicographic order on the language accepted by M is well-ordered iff there is no path in \overline{M} from the initial state to a final state which contains a loop labeled 0. \square

Remark 5.1. One referee pointed out that there is a simpler characterization, which applies to all trim DFA. (A DFA is trim if all states are accessible and there is at most one state which is not coaccessible.) A trim DFA M with input alphabet $\{0, 1\}$ accepts a subset of $\{0, 1\}^*$ which is well-ordered by \leq_ℓ iff for any state p , if $\delta(p, 0) = q$, and p, q are coaccessible, then p, q are not in the same strong component.

For a complete prefix code, the question of determining when (C, \leq_ℓ) is dense is not difficult.

Suppose that $C \subseteq \{0, 1\}^*$ is a complete prefix code. An ordered pair of distinct strings u, v in C is **adjacent** if $u <_\ell v$ and there is no string $w \in C$ with $u <_\ell w <_\ell v$. The linear order (C, \leq_ℓ) is dense iff C does not have an adjacent pair of words, i.e., whenever $u <_\ell v$ in C there is some $w \in C$ with $u <_\ell w <_\ell v$.

Proposition 5.2. A complete prefix code C contains two adjacent strings iff there is a string u and nonnegative integers n, p such that $u01^n$ and $u10^p$ belong to C .

Proof. First assume that for some $u, u01^n, u10^p$ belong to C . If $v \in C$ and $u01^n <_\ell v <_\ell u10^p$, then $v = uw$, for some w and $01^n <_\ell w <_\ell 10^p$. Clearly, w cannot be empty. If the first letter in w is 0, then $w = 0w'$ and $1^n <_\ell w'$. But this implies that w' has 1^n as a prefix, which is impossible. Similarly, the first letter of w cannot be 1.

Now assume that $x <_\ell y$ are adjacent strings in C . Let u be the longest common prefix of x, y , so that $x = u0x'$ and $y = u1y'$, for some strings x', y' . Now if x' can be written as x_10x_2 for some strings x_1, x_2 , then since C is complete, there is a string $v = x_11z \in C$, for some z and $x <_\ell v <_\ell y$, contradicting the assumption that x, y are adjacent. Thus, $x' = 1^n$, for some $n \geq 0$. Similarly, $y' = 0^p$, for some $p \geq 0$. \square

Corollary 5.2. Suppose that C is a regular complete prefix code accepted by the DFA M . Then C has two adjacent strings iff there is an accessible state p in M such that the the behavior of $\delta(p, 0)$ contains a string in 1^* , and the behavior of $\delta(p, 1)$ contains a string in 0^* . \square

Corollary 5.3. There is an $O(n^2)$ algorithm to determine, given an A -automaton M which accepts a complete prefix code, whether μ_M is dense. (Here n is the number of states in M .)

Proof. By definition, μ_M is not dense iff the underlying linear order of μ_M contains two adjacent strings. For each accessible non-final state p of M , check in linear time whether there is any path from $\delta(p, 0)$ to a final state all of whose edges are labeled 1; then in linear time, see if there is any path from $\delta(p, 1)$ to a final state all of whose edges are labeled 0. \square

The more interesting question, for a complete prefix code C , is how to determine whether (C, \leq_ℓ) is scattered. We will show that there is a polynomial time algorithm to determine, given a DFA M that accepts the language $C \subseteq \{0, 1\}^*$, whether the linear order (C, \leq_ℓ) is scattered.

Recall Definition 2.1.

Lemma 5.1. If (L, \leq) is a scattered linear order, and for each $x \in L$, (K_x, \leq) is a scattered linear order, then $\sum_{x \in L} K_x$ is also scattered. If (L, \leq) is a scattered linear order, and $C \subseteq L$, then C with the inherited order is also a scattered linear order.

Proof. We prove only the first statement. Suppose that $\varphi : \mathbb{Q} \rightarrow \sum_{x \in L} K_x$ is an order embedding. Then, for each $x \in L$, unless $\varphi^{-1}(K_x)$ is empty or has exactly one point, K_x is quasi dense. But then φ is in fact an order embedding of \mathbb{Q} into L , which is impossible. \square

For any $C \subseteq \{0, 1\}^*$, and any string $u \in \{0, 1\}^*$, the **left quotient of C by u** , written $u^{-1}C$, is the set $\{v \in \{0, 1\}^* : uv \in C\}$. Note that if C is a (complete) prefix code, and if a left quotient is nonempty, it is also a (complete) prefix code.

Lemma 5.2. Suppose that $C \subseteq \{0, 1\}^*$ and (C, \leq_ℓ) is quasi dense. Then either $0^{-1}C$ or $1^{-1}C$ is quasi dense (with the lexicographic ordering).

Proof. Let $f : \mathbb{Q} \rightarrow C$ be an order embedding. Let \mathbb{Q}_0 be the set of rationals q such that $f(q) \in 0(0^{-1}C)$, and let \mathbb{Q}_1 be $\mathbb{Q} - \mathbb{Q}_0$. Then \mathbb{Q}_0 is closed downward, and \mathbb{Q}_1 is closed upward. Thus, if \mathbb{Q}_0 is nonempty, $0^{-1}C$ is quasi dense. Similarly, if \mathbb{Q}_1 is nonempty $1^{-1}C$ is quasi dense. \square

Lemma 5.3. Suppose that $C \subseteq \{0, 1\}^*$ is quasi dense. Then there is a string u such that

$$u^{-1}C, (u0)^{-1}C \text{ and } (u1)^{-1}C \tag{4}$$

are quasi dense.

Proof. The proof is by contradiction. Suppose That C is quasi dense but there is no string u such that all three sets (4) are quasi dense. In this case, we will construct strings u_n, v_n with the following properties.

1. The length of both u_n and v_n is n .
2. $u_n^{-1}C$ is quasi dense.
3. The string v_n differs from u_n only in the last letter.
4. For each $n \geq 1$, $v_n(v_n^{-1}C)$ is scattered.
5. $C \subseteq \bigcup_{n \geq 1} v_n(v_n^{-1}C) \cup \{u_0, u_1, \dots\}$.

Define $u_0 = \lambda$. Now assume that u_n has been defined such that $u_n^{-1}C$ is quasi dense. Thus, by Lemma 5.2 and our assumption, exactly one of $(u_n 0)^{-1}C$ and $(u_n 1)^{-1}C$ is quasi dense. We define only u_{n+1} , since v_{n+1} is then determined.

$$u_{n+1} := \begin{cases} u_n 0 & \text{if } (u_n 0)^{-1}C \text{ is quasi dense} \\ u_n 1 & \text{otherwise.} \end{cases}$$

We prove our claims. It is clear that the first four items hold. As for the last, any finite string x in C is either one of the u_i or is in $v_n(v_n^{-1}C)$, for some $n \geq 1$. Indeed, suppose that $|x| = n$. If $n = 0$, then $x = u_0$. Otherwise, if x is not the string u_n , then, there is a first letter where x differs from u_n , say, the i -th. Then $x \in v_i(v_i^{-1}C)$.

We will now show that these properties imply that C is scattered, contradicting the assumption.

The strings v_n fall into two groups: those for which $v_n = u_{n-1}0$, the “zero-group”, and those for which $v_n = u_{n-1}1$, the “one-group”. Note that if v_n is in the zero group, then $u_{n+1} = u_n 1$.

When ordered lexicographically, the strings in the zero-group form a finite chain or an omega-chain, the strings in the zero-group together with the strings $u_n, n \geq 0$, form an omega-chain, and the strings in the one-group form a finite chain or an ω^{op} -chain. Thus, the strings the strings u_n and $v_m, n \geq 0, m \geq 1$, form an linearly ordered set isomorphic to $\omega + k$, for some $k \geq 0$, or to $\omega + \omega^{op}$, a scattered linear order. Thus, $\bigcup_n v_n(v_n^{-1}C) \cup \{u_0, u_1, \dots\}$ is isomorphic to the poset obtained from substituting either a singleton linear order or a scattered linear order $v_n(v_n^{-1}C)$ in the scattered order L . By Lemma 5.1, C is scattered. \square

Lemma 5.4. Suppose that C is a regular subset of $\{0, 1\}^*$. If (C, \leq_ℓ) is quasi dense, then there are strings r, s, t in $\{0, 1\}^*$ such that

$$r^{-1}C = (r0s)^{-1}C = (r1t)^{-1}C \neq \emptyset.$$

Proof. Let G be the edge-labeled, directed graph whose vertices are those left quotients $x^{-1}C$ of C such that $(x^{-1}C, \leq_\ell)$ is quasi dense. Thus $C = \lambda^{-1}C$ is a vertex in G , and if $x^{-1}C$ belongs to G , then at least one of $(x0)^{-1}C, (x1)^{-1}C$ belongs to G , by Lemma 5.2. If $(xi)^{-1}C \in G$ there is an edge $x^{-1}C \rightarrow (xi)^{-1}C, i \in \{0, 1\}$. So each vertex of G has outdegree at least one. Also, for each

vertex $x^{-1}C$ in G , there is some string u such that $(xu)^{-1}C$, $(xu0)^{-1}C$ and $(xu1)^{-1}C$ belong to G , by Lemma 5.3. Let \overline{G} be the set of strong components of G . \overline{G} is partially ordered by the relation $S \leq S'$ if there is a path from some vertex of S to some vertex of S' . Since C is regular, both G and \overline{G} are finite. Hence \overline{G} has a maximal element, say S . If $x^{-1}C \in S$, then for any string v , if $(xv)^{-1}C$ belongs to G , then $(xv)^{-1}C$ belongs to S , since S is maximal. Thus, choosing the string u such that $(xu)^{-1}C$, $(xu0)^{-1}C$ and $(xu1)^{-1}C$ all belong to G , since S is a strong component, there are strings s, t such that $(xu0s)^{-1}C = (xu)^{-1}C$, and $(xu1t)^{-1}C = (xu)^{-1}C$. Hence, letting $r = xu$, we have found r, s, t such that

$$r^{-1}C = (r0s)^{-1}C = (r1t)^{-1}C.$$

Since each is quasi dense, each is nonempty. □

There is a stronger converse.

Lemma 5.5. Suppose that C is a not necessarily regular subset of $\{0, 1\}^*$ and r, s, t are strings such that

$$r^{-1}C = (r0s)^{-1}C = (r1t)^{-1}C \neq \emptyset.$$

Then (C, \leq_ℓ) is quasi dense.

Proof. Note that for any string z ,

$$rz \in C \iff r(0s)z \in C \iff r(1t)z \in C.$$

Let $y \in r^{-1}C$, and define $\varphi_0 : \{0, 1\}^+ \rightarrow C$ as the semigroup morphism determined by the conditions:

$$\begin{aligned} 0 &\mapsto 0s \\ 1 &\mapsto 1t. \end{aligned}$$

Then, for example,

$$\varphi_0(0110) = 0s1t1t0s.$$

Note that if $u \leq_\ell v$ in $\{0, 1\}^*$, then $\varphi_0(u) \leq_\ell \varphi_0(v)$. Now define $\varphi(z) = r\varphi_0(z)y$. Thus, for every string z , $\varphi(z) \in C$, and φ is an order embedding of $(\{0, 1\}^*, \leq_\ell)$ into (C, \leq_ℓ) , proving C is quasi dense, by Proposition 4.1. □

We have proved

Proposition 5.3. Let C be a regular subset of $\{0, 1\}^*$. Then (C, \leq_ℓ) is quasi dense iff there are strings r, s, t such that

$$r^{-1}C = (r0s)^{-1}C = (r1t)^{-1}C \neq \emptyset.$$

Corollary 5.4. There is a polynomial time algorithm to decide, given a DFA which accepts a subset C of $\{0, 1\}^*$, whether (C, \leq_ℓ) is quasi dense.

Proof. We need only check for each coaccessible state q in the minimal DFA for C , whether there are strings s, t such that

$$q = \delta(q, 0s) = \delta(q, 1t). \quad (5)$$

Any efficient shortest path algorithm will solve this problem in polynomial time. \square

Remark 5.2. Note that Proposition 5.3 may be phrased as follows. A regular language C is quasi dense iff there is a coaccessible state p in the minimal DFA for C such that $p, \delta(p, 0)$ and $\delta(p, 1)$ are all in the same strong component. One of the referees noted that a trim DFA M accepts a quasi dense language iff there are states p, q, q' with $\delta(p, 0) = q$ and $\delta(p, 1) = q'$ and all three states are in the same strong component. In fact, this result follows from Proposition 5.3 above, since if M is any DFA accepting C , there is surjective morphism from M to the minimal DFA for C , and the states in any strong component of M map to states in a strong component of the minimal machine.

Corollary 5.5. There is a polynomial time algorithm to decide, given an A -automaton M , whether μ_M is scattered. \square

6. Open Problems

The regular words on an alphabet A may be described as the least class of words containing the singletons $a \in A$, closed under the operations of product, omega and omega-op powers, and the shuffle operations. It would be nice to have a complete axiomatization of these operations. Thomas [Thom86] has shown, using methods and results of formal logic, that the equational theory of this algebra is decidable. However, the methods applied in [Thom86] do not provide an elementary upper bound, not even for the equational theory of product, omega power and omega-op power. It would be interesting to find upper and lower bounds.

7. Acknowledgement

We would like to thank the referees for several helpful comments on an earlier draft. Their suggestions have improved both the presentation and several results.

References

- [BlCho01] S.L. Bloom and C. Choffrut. Long words: the theory of concatenation and ω -power. *Theoretical Computer Science*, vol. 259, (1-2) 2001, 533-548.
- [BruyCar] V. Bruyère and O. Carton. Automata on linear orderings. Proceedings *Mathematical Foundations of Computer Science*, 2001, Lecture Notes in Computer Science, vol. 2136.
- [BruyCar2] V. Bruyère and O. Carton. Hierarchy among automata on linear orderings. To appear TCS 2002 (IFIP conference at Montreal at the end of August).

- [Brzo80] J.A. Brzozowski and F.E. Fich. Languages of R-trivial monoids. *J. Comp. Systems Sci*, 20, 1980, 32–49.
- [Car] O. Carton. Accessibility on automata on scattered linear orderings. To appear.
- [Cour78] B. Courcelle. Frontiers of infinite trees. *RAIRO Informatique*, vol. 12, 1978, 319–337.
- [Cour83] B. Courcelle. Fundamental properties of infinite trees. *Theoretical Computer Science*, vol. 25, 1983, 95–169.
- [Eil76] S. Eilenberg. *Automata, Languages and Machines*, vol. B. Academic Press, New York, 1976.
- [Heil80] S. Heilbrunner. An algorithm for the solution of fixed-point equations for infinite words. *RAIRO Informatique*, vol. 14, no. 2, 1980, 131–141.
- [Pin86] J.-E. Pin. *Varieties of Formal Languages*. Plenum Publishing Corp., New York, 1986.
- [Ro82] J.B. Rosenstein. *Linear Orderings*. Academic Press, New York, 1982.
- [Thom86] W. Thomas. On frontiers of regular trees. *Theoretical Informatics and Applications*, vol. 20, 1986, 371–381.