Basic Research in Computer Science

Some Remarks on Regular Words

Stephen L. Bloom
Zoltán Ésik

BRICS Report Series **RS-02-39**

ISSN 0909-0878 **September 2002**

**See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:**

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/02/39/`

Some remarks on regular words

Stephen L. Bloom*
Department of Computer Science
Stevens Institute of Technology
Hoboken, NJ 07030

Zoltán Ésik†
Institute for Informatics
University of Szeged
Szeged, Hungary

August 25, 2002

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 "discrete 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.

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.

*Partially supported by NSF grant 0119916.
†Partially supported by BRICS, Aalborg, Denmark, and NSF grant 0119916.

In this paper, we assume all "alphabets" are initial subsets $\{a_1, \ldots, a_n\}$ of the countable set $\{a_1, a_2, \ldots\}$. 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 $\{1, 2, \ldots, n\}$, 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 $(L_u, \leq_u)$ is a countable linearly ordered set, and $u : L_u \to A$ 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 $f : L_u \to L_v$ such that for all $x, y \in L_u$,

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

We usually identify isomorphic words. We call the linearly ordered set $(L_u, \leq_u)$, or just L_u for short, the **underlying order** of the word u.

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

Definition 2.1 *Suppose that $(L, \leq)$ is a linear order, and for each $x \in L$, let $(K_x, \leq)$ be a linear order. The ordering $\sum_{x \in L} K_x$, obtained by **substitution** of K_x for $x \in L$, 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 $A = \{a_1, \ldots, a_n\}$, 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 $w = u(a_1/v_{a_1}, \ldots, a_n/v_{a_n})$, 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:*

$$w(x, y) := v_{u(y)}(x).$$

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 $A^* - \{\lambda\}$. 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,

$$x \quad = \quad ux.$$

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

$$x \quad = \quad xu.$$

Suppose that $(P, \leq_P)$ and $(Q, \leq_Q)$ are linear orders. The **sum** $P + Q$ is defined as follows: $P + Q$ is the disjoint union $P \cup Q$, ordered so that every element in P is less than each element in Q; otherwise, the elements are ordered as in P or Q, respectively. We define two special cases of the **product** of two linear orders. The linear order $P \otimes \omega$ is $\sum_{i \in \omega} P_i$, where $P_i = P$, for each $i \geq 0$. The linear order $P \otimes \omega^{op}$ is $\sum_{i \in \omega^{op}} P_i$, where $P_i = P$, for each $i < 0$.

Corollary 2.4 *Let* u, v *be words. The underlying order of* uv *is the sum of the order of* u *and the order of* v*; the underlying order of* u^ω *is* $L_u \otimes \omega$*, and the underlying order of* $u^{\omega^{op}}$ *is* $L_u \otimes \omega^{op}$*.* □

We need the following fact, a special case of a construction in [Heil80], proved using a "back-and-forth" argument.

Lemma 2.5 *For any nonempty finite set* $A = \{a_1, \ldots, a_n\}$ *there is, up to isomorphism, a unique word* $(P, \leq, \rho_n)$ *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 $\rho_n(z) = a_i$.

The underlying order of the word ρ_n is isomorphic to the rationals, $\mathbb{Q}$.

Now, we define the **shuffle** [Heil80] of the finite sequence $(u_1, \ldots, u_n)$ of words by:

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

Thus, in particular, we may write

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

The underlying order of the word $[u_1, \ldots, u_n]^\eta$ can be described in general as follows. For linearly ordered sets $P_1, \ldots, P_n$, let S be the subset of

$$\bigcup_{i=1}^{n} \{i\} \times P_i \times \mathbb{Q}$$

defined by:

1. For each $q \in \mathbb{Q}$ there is a unique $i \in [n]$ such that for all $x \in P_i$, $(i, x, q) \in S$. Moreover, if $(j, y, q) \in S$, then $j = i$, and $y \in P_i$.
2. if $q < q'$ in $\mathbb{Q}$, then for each $i \in [n]$ there is some q'' in $\mathbb{Q}$ such that $q < q'' < q'$ and for all $x \in P_i$, $(i, x, q'') \in S$.

The set S is ordered by:

$$(i, x, q) \leq (j, y, q') \quad \iff \quad q < q' \text{ or } (q = q' \text{ and } x \leq y \text{ in } P_i).$$

We denote $(S, \leq)$ by $[P_1, \ldots, P_n]^\eta$. We call it the **shuffle** of the linear orders $P_1, \ldots, P_n$.

Proposition 2.6 *If L_i is the underlying order of the word u_i, $i \in [n]$, then the underlying order of the word $[u_1, \ldots, u_n]^\eta$ is $[L_1, \ldots, L_n]^\eta$.* □

Definition 2.7 *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*

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

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:*

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

Note that a regular word is not empty. Note also 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 nonleaf 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 of equations that determine the discrete 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

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

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 2.8 *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. □

Proposition 2.9 *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 $P \mapsto P \otimes \omega$ and $P \mapsto P \otimes \omega^{op}$.* □

Definition 2.10 *We call a linear order* ***regular*** *if it is isomorphic to an order in the class* **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) 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 nonempty words in $(A \cup \{x_1, \ldots, x_n\})^*$, 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.

Proposition 2.11 *[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.* □

We will need a slight extension of this result.

Proposition 2.12 *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, locally finite, tree t whose interior nodes have one or two successors. We need to find a regular complete binary tree whose frontier is isomorphic to u. Now since t has only a finite number of subtrees, there are only finitely many subtrees, say $t_1, \ldots, t_N$, which are not leaves, and, say, $e_1, \ldots, e_k$ which are leaves. If $N = 0$, t is itself a leaf, the frontier of t is a letter, and the corresponding C consists of just the empty string, a complete prefix code. When $N > 0$, by choosing a variable x_i for the tree t_i, $i = 1, \ldots, N$, we obtain a system of N equations, as follows. If t_i is a tree whose root has both a left and right successor, say the roots of the trees t_l and t_r respectively, we indicate this fact by writing:

$$t_i \quad = \quad 0.t_l + 1.t_r.$$

In the case t_i has just a left successor, the root of the tree t_l, we write

$$t_i \quad = \quad 0.t_l.$$

Similarly, for just a right successor. When one or both of the trees t_l, t_r is a leaf, we write the label of the leaf instead.

Thus, we get N equations, each one of which has one of the following three forms:

$$\begin{aligned} x_i &= 0.z_j + 1.z_k \\ x_i &= 0.z_j \\ x_i &= 1.z_j, \end{aligned} \tag{2}$$

where the z's range over $\{x_1, \ldots, x_N\} \cup A$. For example, if the root of the tree t_i has two successors, the equation for x_i has the first form, where z_j

is in A if z_j is (the label of) a leaf and the left subtree of t_i is z_j, and z_j is x_r if the left subtree of t_i is the tree t_r, etc. If the root of t_i has only one successor, then the equation for x_i has one of the last two forms.

From the collection of all subtrees $\{t_1, \ldots, t_N, e_1, \ldots, e_k\}$ of t we construct a directed graph $G = (V, E)$ whose vertices are the subtrees. There is an edge $u \to v$ if the root of u has the root of the subtree v as either a right or left successor. We say the *degree of the subtree* t_i is the number of successors of the root of t_i. If $u \in V$ is a leaf, u has degree 0. Note: since t is locally finite, for each vertex $u \in V$ there is some path in G to a vertex of degree 0. Let $\sim$ be the least equivalence on the vertices such that $u \sim v$ if there is a path $u \rightsquigarrow v$ in G such that each vertex on the path has degree 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 $v = v_0, v_1, \ldots, v_s$ 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 $v_i \in [v]$. Also, if v and v' have degree 0 or 2 and $v \sim v'$, then $v = v'$.

We now modify the system of equations above, assuming that t_1 is the given binary tree. Depending on the degree of t_i, we do one of three things. If the root of t_i has degree 2, we keep the original equation (2) for x_i. Otherwise, if a vertex e_j of degree 0 belongs to $[t_i]$, replace x_i on the right side of every equation by the label, say a_j of this leaf. If t_i has degree 1 and if t_k has degree 2 and $t_k \in [t_i]$, delete the original equation for x_i and replace x_i on the right side of every equation by x_k. We have to make some more adjustments if $i = 1$, the index of the original tree. If there is a leaf in $[t_1]$, the system can be replaced by $x_1 = a$, if a is the label of the leaf. If there is a tree of degree 2 in $[t_1]$, say t_k, replace the equation for x_1 by $x_1 = s$, where s is the right side of the equation for x_k. In the new system, every equation has the form (2) or the form $x_1 = a$, for some $a \in A$. This system is a description for a regular complete binary tree whose frontier is isomorphic to the original word. □

3 Language theoretic characterizations

We now turn to language theoretic characterizations of the regular words, and the discretely ordered regular words, described below. We observe (in Proposition 3.3) 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 3.4) 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 3.11, 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. 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 $B_n = \{b_1, \ldots, b_n\}$ is an n-element alphabet, ordered by $b_1 < b_2 < \ldots < b_n$. Then the strings on B_n are also linearly ordered by the lexicographic order:*

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

Proposition 3.2 *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 $(P, \leq_\ell)$.*

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

$$(11 + 0)^*10. \tag{3}$$

(See [Thom86].) And 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. And 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 2.12 can be extended to give a different proof of this result.) □

Proposition 3.3 *Suppose that C is a (complete) prefix code and $\leq_\ell$ is the lexicographic order. Suppose that C is partitioned into $R_1, \ldots, R_n$, where each set R_i, $i \in [n]$, is regular. Then there is a regular word $u(R_1, \ldots, R_n) = (C, \leq_\ell, u)$, 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 $R_1, \ldots, R_n$ of disjoint regular subsets of $\{0,1\}^*$ such that $R_1 \cup \ldots \cup R_n$ is a complete prefix code and w is isomorphic to $u(R_1, \ldots, R_n)$.*

Proof. Both directions follow immediately from the fact that, up to isomorphism, regular words are frontiers of (complete) locally finite, regular trees. See Propositions 2.11 and 2.12. 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. □

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

Proposition 3.4 *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 3.3.

Define the set $\hat{L}$ as the set of words obtained by putting -1 on the right of each word in L:

$$\hat{L} \quad := \quad \{x(-1) : x \in L\}.$$

Then $\hat{L}$ is a prefix free, 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 $(L, \leq_\ell) \to (\hat{L}, \leq_\ell)$. Now define the function $w : \hat{L} \to A$ as follows.

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

Thus, the words w and u are isomorphic.

Now we let $\varphi : \{-1, 0, 1\}^+ \to \{0, 1\}^+$ 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 $\hat{L}$ is prefix free, it follows that $\varphi(\hat{L})$ is a prefix code and $\varphi : (\hat{L}, \leq_\ell) \to (\varphi(\hat{L}), \leq_\ell)$ is an order isomorphism. Thus, the word v is isomorphic to u, where

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

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 2.12, u is isomorphic to a regular word whose underlying order is a regular, complete prefix code. □

Remark 3.5 *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 $b_1, \ldots, b_k$, with $b_1 < \ldots < b_k$, 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 terminolgy. 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, $\delta : Q \times \{0, 1\} \to Q$ 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 $\delta : Q \times \{0, 1\}^* \to Q$ 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 3.6 *For an n-element alphabet A, an **A-automaton***

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

is a DFA with n final states, labeled $f_1, \ldots, f_n$, 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 $(L, \leq_\ell, \mu_M)$ 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 $\mathcal{L}(M)$ is a complete prefix code and μ_M is isomorphic to u.

Example. Let A be the k letter alphabet $\{a_1, \ldots, a_k\}$. We define an A-automaton determining the word ρ_k (see Lemma 2.5).

The (non sink) states are divided into three groups: $c_0, \ldots, c_{k-1}$, $d_0, \ldots, d_{k-1}$ and the final states $f_1, \ldots, f_k$. 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, $i \in [n]$.

Lemma 3.7 *For a string $x \in \{0,1\}^*$, and integer $0 \leq i < n$, $\delta(c_0, x) = c_i$ iff $x \in B_i$.*

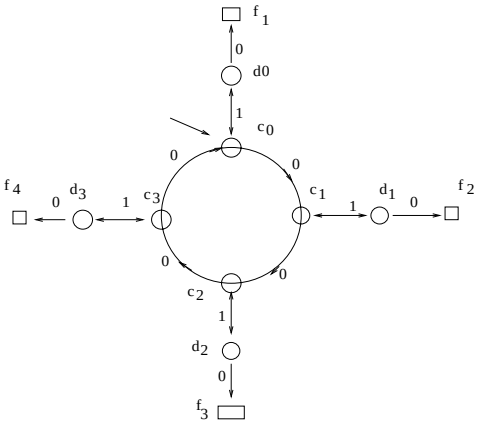

Figure 1: Most of the automaton for ρ_4

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

Lemma 3.8 *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, $(C, \leq_\ell)$ is a dense linear order, with no first or last (see (3) above). We show that for any two strings $x <_\ell y$ 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 $k > 0$, the word $z_k = (u11)0^k(10)$ belongs to $B10$ and $x < z_k < y$. For any $0 \leq i < n$, we can find an appropriate k such that $\delta(c_0, u110^k) = c_i$, so that $\delta(c_0, z_k) = f_i$.

The second case is $x = u0v(10)$ and $y = u1w(10)$, where $u0v$ and $u1w$ belong to B. Then, for any $k > 0$, if $z_k = u1w0^k10$, then

$$x < z_k < y,$$

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

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.

Lemma 3.9 *Let M_0 be a B-automaton on the k-letter alphabet B. Let $M_1, \ldots, M_k$ be A-automata, Then there is an A-automaton*

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

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

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

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

Proof. Let $\langle M_1, \ldots, M_k \rangle$ 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 $\langle M_1, \ldots, M_k \rangle$ has k initial states. For $i \in [k]$, the i-th initial state is the initial state of M_i. Then, let $M = M_0 \cdot \langle M_1, \ldots, M_k \rangle$ be obtained from the disjoint union of the states in M_0 and $\langle M_1, \ldots, M_k \rangle$ by identifying the sink states and the i-th final state of M_0 with the initial state of M_i. Then 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. □

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 3.10 *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*

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

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

Proposition 3.11 *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,*

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

Proof. By Lemma 3.9, we need only show how to find A-automata for the basic words $a, ab, a^\omega, a^{\omega^{op}}$ and ρ_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 $a = a_i \in \{a_1, \ldots, a_n\}$, let $M(a_i)$ have a sink and final states $f_1, \ldots, f_n$, 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

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

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

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

$$\begin{aligned}
|M(a_i)| &= n \\
|M(uv)| &= 1 + |M(u)| + |M(v)| - n \\
|M(u^\omega)| &= 1 + |M(u)| \\
|M(u^{\omega^{op}})| &= 1 + |M(u)| \\
|M([u_1, \ldots, u_k]^\eta)| &= 3k + \sum_{i=1}^{k} |M(u_i)| - n(k-1) - k \\
&= 2k - n(k-1) + \sum_{i=1}^{k} |M(u_i)|, \qquad (4)
\end{aligned}$$

by Lemma 3.10.

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

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

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 $w = [u_1, \ldots, u_k]^\eta$, where u_i are regular expressions on the n-letter alphabet, by (4)

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

since $n \geq 1$,

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

by induction,

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

This completes the proof. □

4 Discrete regular words

A linear order $(L, \leq)$ is **discrete** if there is no order embedding of $\mathbb{Q}$ into L. (In [BruyCar, BruyCar2], a discrete linear order is called "scattered".) A word $(L_u, \leq_u, u)$ is discrete when its underlying linear order is. Note that any word isomorphic to a discrete word is discrete. In order to avoid the term "not discrete", we call a linear order $(L, \leq)$ **quasi dense** if there is an order embedding $\mathbb{Q} \to L$. For example, all of the linear orders $[P_1, \ldots, P_k]^\eta$ are quasi dense. In this section, we consider the question of when a regular word is discrete, and give several characterizations of such words. In the next section, we give polynomial time algorithm to decide when the regular word determined by a given A-automaton is discrete.

Since every shuffle is quasi-dense, we have

Proposition 4.1 *A regular word on the alphabet A is discrete 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 discrete regular orders are those isomorphic to the orders in the least class of linear orders which contains the singleton* **1**, *closed under sum and the operations $P \mapsto P \otimes \omega$ and $P \mapsto P \otimes \omega^{op}$.*

(Courcelle [Cour78] characterized the discrete regular words as the solutions to a "quasi rational" system of fixpoint equations.)

Which are the regular (complete) prefix codes C such that $(C, \leq_\ell)$ is discretely ordered?

We call a DFA M a **monotone** automaton 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 automaton 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 4.2 *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 discrete 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 $(P, \leq_\ell)$.

Lemma 4.4 *Suppose that $P \subseteq \{0,1\}^*$. Then*

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

Proof. We prove only the first statement. Suppose that $u, v \in 1^*0P$ and $u \leq_\ell v$. We may write $u = 1^n 0u'$, $v = 1^m 0v'$, for some strings $u', v' \in P$. Then either $n < m$ or $n = m$ and $u' \leq_\ell v'$ in P. This is exactly the definition of $P \otimes \omega$. □

Proposition 4.5 *Let L be a discrete regular linear order. Then L is isomorphic to a monotone subset of $\{0,1\}^*$ which is a complete prefix code.*

Proof. Clearly, $\mathbf{1}$ has the required property, and we will show that it is preserved by the operations $P \mapsto P \otimes \omega$, $P \mapsto P \otimes \omega^{op}$ and $P, Q \mapsto (P + Q)$.

The set consisting of the empty string is monotone, and a complete prefix code. Ordered lexicographically, this set is isomorphic to $\mathbf{1}$.

Now assume that P is a monotone, complete prefix code. Then, by Lemma 4.4, 1^*0P, ordered lexicographically, is isomorphic to $P \otimes \omega$. If M is a monotone DFA accepting P, then a monotone automaton M' accepting 1^*0P is obtained by adding a new initial state to the states of M, say q_0', and defining the transition function to be the extension of the transition function of M for which

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

By interchanging the 0's and 1's, we get a monotone DFA accepting 0^*1P, which, when ordered lexicographically, is isomorphic to $P \otimes \omega^{op}$. Last, if P, Q are monotone, complete prefix codes, so is $0P \cup 1Q$, which, when ordered lexicographically is isomorphic to $P + Q$, by Lemma 4.4. We show

how one may obtain a monotone DFA accepting $0P \cup 1Q$ from monotone DFA's, say M_P, M_Q, which accept P and Q, having initial states q_0^P and q_0^Q, respectively. Add a new initial state q_0' to the disjoint union of the states of M_P, M_Q, and define a new transition function extending those of M_P and M_Q as follows:

$$\begin{aligned} \delta(q_0', 0) &:= q_0^P \\ \delta(q_0', 1) &:= q_0^Q. \quad \square \end{aligned}$$

There is a slightly stronger converse.

Proposition 4.6 *Suppose that C is a monotone prefix code (not necessarily complete). Then C, ordered lexicographically, is a discrete regular linear order.*

Proof. Let M be the minimal DFA accepting C. We assume the states of M are $\{q_1, q_2, \ldots, q_n\}$, 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 discrete 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 discrete 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 discrete regular linear order, by

induction, and the linear order determined by C is isomorphic to $L \otimes \omega$. Similarly, if the initial state has a self loop on the letter 0, C will determine a linear order isomorphic to $L \otimes \omega^{op}$. The proof is complete. □

Corollary 4.7 *A linear order is discrete regular linear order iff it is isomorphic to the lexicographic ordering of a monotone, complete prefix code.* □

Corollary 4.8 *A word w on the alphabet A is regular and discrete 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 4.8 (and elsewhere) is crucial. A regular complete prefix code C may not be monotone, but nonetheless $(C, \leq_\ell)$ may be discrete. Consider the set $C = (01)^*(00 + 1)$, for example. The linear order $(C, \leq_\ell)$ is isomorphic to $\omega + \omega^{op}$, and C is not monotone.

We summarize the above facts.

Theorem 4.9 *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 discrete (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 $(L_u, \leq_\ell)$ is discrete and a regular (complete) prefix code.*
5. *w is isomorphic to a word $u(R_1, \ldots, R_n)$, where the sets R_i are regular, pairwise disjoint, and $\bigcup_{i \in [n]} R_i$ is a monotone (complete) prefix code.*

□

From Theorem 4.9 and Cantor's normal form theorem [Sier58], we obtain the following Corollary.

Corollary 4.10 *An ordinal α is in **RL** iff $\alpha < \omega^\omega$.* □

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.* □

Thus, every A-automaton accepts a prefix code.

Corollary 5.2 *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 $(P, \leq_\ell)$ is well ordered.*

Proof outline. First, in polynomial time, find the minimal automaton $\overline{M}$ 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, 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. □

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 $(C, \leq_\ell)$ is discrete.

Recall Definition 2.1.

Lemma 5.3 *If $(L, \leq)$ is a discrete linear order, and for each $x \in L$, $(K_x, \leq)$ is a discrete linear order, then $\sum_{x \in L} K_x$ is also discrete. If $(L, \leq)$ is a discrete linear order, and $C \subseteq L$, then C with the inherited order is also a discrete linear order.*

Proof. We prove only the first statement. Suppose that $\varphi : \mathbb{Q} \to \sum_{x \in L} K_x$ 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. □

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 5.4 *Suppose that $C \subseteq \{0,1\}^*$ and $(C, \leq_\ell)$ is quasi dense. Then either $(0^{-1}C, \leq_\ell)$ or $(1^{-1}C, \leq_\ell)$ is quasi dense.*

Proof. Let $f : \mathbb{Q} \to C$ 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. □

Lemma 5.5 *Suppose that $C \subseteq \{0,1\}^*$ is quasi dense. Then there is a string u such that*

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

are quasi dense.

Proof. Suppose not. Then 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 word v_n differs from u_n only in the last letter.
4. For each $n \geq 1$, $v_n(v_n^{-1}C)$ is discrete.
5. $C \subseteq \bigcup_{n \geq 1} v_n(v_n^{-1}C) \cup \{u_0, u_1, \ldots\}$.

Define $u_0 = \lambda$. Now assume that u_n has been defined such that $u_n^{-1}C$ is quasi dense. Thus, by assumption, exactly one of $(u_n0)^{-1}C$ and $(u_n1)^{-1}C$ is quasi dense. We define only u_{n+1}, since v_{n+1} is then determined.

$$u_{n+1} := \begin{cases} u_n0 & \text{if } (u_n0)^{-1}C \text{ is quasi-dense} \\ u_n1 & \text{otherwise.} \end{cases}$$

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 discrete, contradicting the assumption.

The words 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 lexicographivally, 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 discrete linear order.

Thus, $\bigcup_n v_n(v_n^{-1}C) \cup \{u_0, u_1, \ldots\}$ is isomorphic to the poset obtained from substituting either a singleton linear order or a discrete linear order $v_n(v_n^{-1}C)$ in the discrete order L. By Lemma 5.3, C is discrete. □

Lemma 5.6 *Suppose that C is a regular subset of $\{0,1\}^*$. If $(C, \leq_\ell)$ 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 5.4. If $(xi)^{-1}C \in G$ there is an edge $x^{-1}C \to (xi)^{-1}C$, $i \in \{0,1\}$. 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 5.5. Let $\overline{G}$ be the set of strong components of G is partially ordered by: $S \leq S'$ 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 slightly stronger converse.

Lemma 5.7 *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 $(C, \leq_\ell)$ is quasi dense.

Proof. Let $y \in r^{-1}C$, and define $\varphi : \{0,1\}^+ \to C$ as the semigroup morphism determined by the following conditions:

$$\begin{aligned} 0 &\mapsto r0sy \\ 1 &\mapsto r1ty. \end{aligned}$$

Then, for example,

$$0110 \;\mapsto\; r0syr1tyr1tyr0sy.$$

Thus, the restriction of φ to a copy of $\mathbb{Q}$, say to $(11+0)^*10$, is an order embedding of $\mathbb{Q}$ into $(C, \leq_\ell)$, proving C is quasi dense. □

We have proved

Proposition 5.8 *Let C be a regular subset of $\{0,1\}^*$. Then $(C, \leq_\ell)$ is quasi dense iff there are strings r, s, t such that*

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

Corollary 5.9 *There is a polynomial time algorithm to decide, given a DFA which accepts a subset C of $\{0,1\}^*$, whether $(C, \leq_\ell)$ is quasi dense.*

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

$$q = \delta(q, 0s) \;=\; \delta(q, 1t). \tag{5}$$

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

Corollary 5.10 *There is a polynomial time algorithm to decide, given an A-automaton M, whether μ_M is discrete.* □

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.

[Sier58] W. Sierpinski. *Cardinal and Ordinal Numbers.* Warsaw: PWN, 1958.

[Thom86] W. Thomas. On frontiers of regular trees. *Theoretical Informatics and Applications*, vol. 20, 1986, 371–381.

Recent BRICS Report Series Publications

RS-02-39 **Stephen L. Bloom and Zoltán Ésik.** ***Some Remarks on Regular Words*****. September 2002. 27 pp.**

RS-02-38 **Daniele Varacca.** ***The Powerdomain of Indexed Valuations*****. September 2002. 54 pp. Short version appears in Plotkin, editor, *Seventeenth Annual IEEE Symposium on Logic in Computer Science*, LICS '02 Proceedings, 2002, pages 299–308.**

RS-02-37 **Mads Sig Ager, Olivier Danvy, and Mayer Goldberg.** ***A Symmetric Approach to Compilation and Decompilation*****. August 2002. To appear in Neil Jones's Festschrift.**

RS-02-36 **Daniel Damian and Olivier Danvy.** ***CPS Transformation of Flow Information, Part II: Administrative Reductions*****. August 2002. 9 pp. To appear in the *Journal of Functional Programming*. This report supersedes the earlier BRICS report RS-01-40.**

RS-02-35 **Patricia Bouyer.** ***Timed Automata May Cause Some Troubles*****. August 2002. 44 pp.**

RS-02-34 **Morten Rhiger.** ***A Foundation for Embedded Languages*****. August 2002. 29 pp.**

RS-02-33 **Vincent Balat and Olivier Danvy.** ***Memoization in Type-Directed Partial Evaluation*****. July 2002. 18 pp. To appear in Batory and Consel, editors, *ACM SIGPLAN/SIGSOFT Conference on Generative Programming and Component Engineering*, GPCE '02 Proceedings, LNCS, 2002.**

RS-02-32 **Mads Sig Ager, Olivier Danvy, and Henning Korsholm Rohde.** ***On Obtaining Knuth, Morris, and Pratt's String Matcher by Partial Evaluation*****. July 2002. 43 pp. To appear in Chin, editor, *ACM SIGPLAN ASIAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, ASIA-PEPM '02 Proceedings, 2002.**

RS-02-31 **Ulrich Kohlenbach and Paulo B. Oliva.** ***Proof Mining: A Systematic Way of Analysing Proofs in Mathematics*****. June 2002. 47 pp.**

RS-02-30 **Olivier Danvy and Ulrik P. Schultz.** ***Lambda-Lifting in Quadratic Time*****. June 2002.**