URL: http://www.elsevier.nl/locate/entcs/volume31.html 15 pages

An ω-Automata Approach to the Compression of Bi-Level Images

Yih-Kai Lin and Hsu-Chun Yen[1]

Dept. of Electrical Engineering, National Taiwan University
Taipei, Taiwan 106, R.O.C.
yen@cc.ee.ntu.edu.tw

Abstract

We use *ω–finite automata* as a device for compressing bi-level images. One of the advantages of our approach, as opposed to using the conventional finite automata, lies in that ω–finite automata are capable of representing image objects of zero size, such as lines and points. To demonstrate the feasibility of our strategy, we also show how a number of image processing operations, including *shift, flip, rotation, complement, boundary, difference, union, intersection*, and *size*, can effectively be carried out in the framework of ω-finite automata.

1 Introduction

As a modeling tool, *finite automata* have played an important role in various areas of computer science and the related disciplines. Aside from their usefulness in formal languages and complexity theory, it has recently been shown that finite automata can also play a constructive role in the compression of digital images. (See, e.g., [5–8].) By exploiting self-similarities within images, evidence has suggested that *finite automata*, serving as an image compression tool, are capable of significantly reducing the amount of memory needed to encode bi-level (i.e., black-and-white) images. More recently, a generalized model called *weighted finite automata* [5] has been proposed to encode and process gray-tone images. The idea of using finite automata to encode bi-level images is the following. By recursively subdividing an image area into four quadrants, a subimage can be addressed by a string $x_1, ..., x_n$, where each x_i belongs to an alphabet of four letters (representing the four quadrants), and n represents the granularity of the subdivision. The subdivision procedure continues until quadrants are either entirely black or entirely white. In this setting, an image can then be associated with a *language* L in such a

[1] To whom all correspondence should be sent.

way that a string x is in L iff the corresponding subimage addressed by x is 'black.' To support the applicability of such a strategy, it has been shown in [7] that various image processing applications can effectively be performed in the framework of finite automata.

An alternative approach is to use *ω-finite transducer* [4] to process bi-level images. In a language accepted by a conventional finite automaton, a string is of finite length, meaning that the corresponding image area has non-zero size. It seems that finite automata are not appropriate for representing image objects of zero size, such as 'points' and 'lines.' To overcome such a deficiency in image representation, in ω-automata (i.e., automata on infinite strings) a string of infinite length is used to capture the essence of a 'pixel.' The coordinates of a point in the plane can be treated as a 2 -dimensional vector of real numbers in Euclidean space. Since any real number can be represented by an infinite string over an appropriate alphabet, it becomes possible to use ω-automata to represent zero-sized images, as well as images with scattered 'noises' of zero size.

Motivated by the work of [4] (also [8]), in this paper we focus on those bi-level images representable by *Büchi automata* [3], a model that has been extensively studied in the literature (see, e.g., [3,13]). Taking advantage of several known results as well as results derived in this paper concerning *Büchi automata*, we are able to show how a variety of image processing operations (including several that were not studied in [4]) are carried out in a unified framework based upon ω-finite automata. Among the new results is the measurement of *sizes* of images represented by deterministic Büchi automata. To this end, we reduce the computation of image sizes to the *probabilistic reachability problem* in the theory of *Markov chains*, which, in turn, leads to an effective procedure for measuring the sizes of images encoded by deterministic Büchi automata. We also show the relationship between image representation schemes based on finite automata and Büchi automata.

The remainder of the paper is organized as follows. In Section 2, the definition of Büchi automata as well as their connection to bi-level images are introduced. Several basic results concerning Büchi automata are given in Section 3, using which a number of image processing operations can effectively be carried out in the framework of ω-automata (Section 4).

2 Preliminaries

Given an *alphabet* (i.e., a finite set of *symbols*) Σ, Σ^* and Σ^ω denote the sets of finite words and ω–words, respectively, over Σ. (An ω–word over Σ is an infinite string written in the form $\omega_1\omega_2\ldots$ where $\omega_i \in \Sigma$.) Let R (resp., Q and Z) denote the set of real numbers (resp, rational numbers and integers), and R^k (resp., Q^k and Z^k) the set of vectors of k real numbers (resp., rational numbers and integers).

Even though we mainly focus on two-dimensional digital images and num-

bers in base 2, our results can easily be extended to higher dimensions and other bases. In our setting, the locations of image pixels are referred to by their quadtree addresses in the following way. The address of a node in a quadtree is a string ω over the alphabet $\Sigma = \{0,1\}^2$ (i.e., $\{(0,0),(0,1),(1,0),(1,1)\}$). Given a square area, the empty string ε is chosen as the address of the whole square. The four quadrants of the square addressed by w have $w \cdot (0,0)$, $w \cdot (0,1)$, $w \cdot (1,0)$, and $w \cdot (1,1)$ as their addresses, where '$\cdot$' denotes the string concatenation operation. The example shown in Figure 1 illustrates the addresses of several subsquares and the corresponding quadtree.

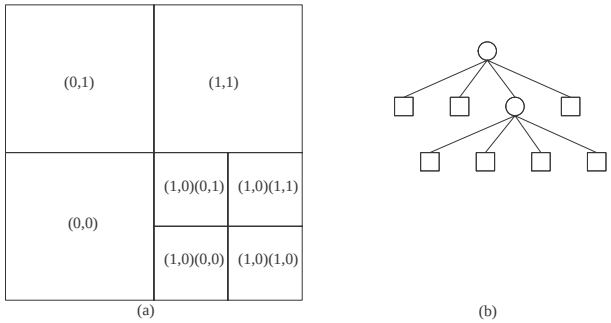


Fig. 1. (a) The addresses of some subsquares; (b) the corresponding quadtree.

The use of *finite automata* to encode (and compress) images is not new. Results along this line of research can be found in [5–8]. The encoding is done is such a way that the set $\{0,1\}^2$ is treated as the input alphabet of a finite automaton, and a string $x_1x_2\cdots x_k$ ($x_i \in \{0,1\}^2$) is accepted iff the corresponding square represented by $x_1x_2\cdots x_k$ (in terms of the quadtree address) is 'black.' Figure 2 demonstrates a picture of diminishing triangles (given in [8]) together with the corresponding finite automaton. Figure 3 illustrates the finite automaton 'approximating' a line, and clearly, more states are required in order to get an approximation of higher resolution. The inability to faithfully encode a line (which has zero size) is exactly the pitfall that a finite automaton only accepts strings of finite length, and each of such strings represents a square of non-zero size. To overcome such a shortcoming, in this paper we use *ω-finite automata* (which accept strings of infinite length) to encode bi-level images. Related results can be found in [4].

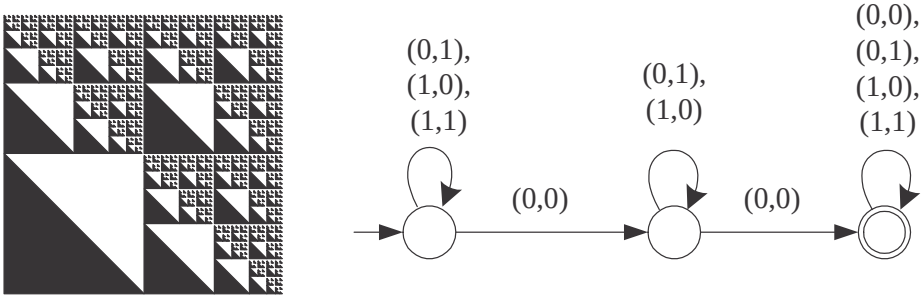


Fig. 2. The diminishing triangles and the corresponding FA.

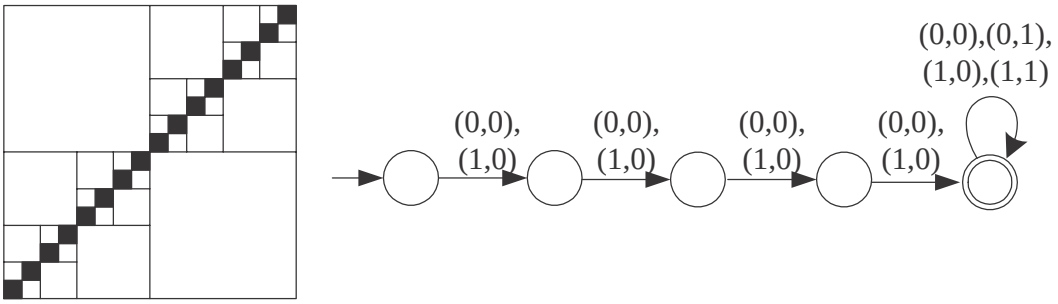


Fig. 3. The approximation of a slope line and the corresponding FA.

Without loss of generality, two-dimensional square images are assumed to be normalized in the sense that the ranges of the x and y coordinates are in $[0, 1]$. The transformation between the real coordinates and the associated quadtree addresses of a pixel is rather straightforward. Let $\Sigma = \{0, 1\}$. Given a point $p = (x = x_1 2^{-1} + x_2 2^{-2} + x_3 2^{-3} + \ldots, y = y_1 2^{-1} + y_2 2^{-2} + y_3 2^{-3} + \ldots) \in [0, 1]^2$, the corresponding quadtree address is $(x_1, y_1)(x_2, y_2)(x_3, y_3)\ldots$, where $(x_i, y_i) \in \Sigma^2$. For example, point $(\frac{1}{3}, \frac{1}{3})$ can be mapped into '$(0, 0)(1, 1)(0, 0)$ $(1, 1)(0, 0)(1, 1)\ldots$.' In this setting, images with infinite resolution can be represented as sets of ω-words over Σ^2. Throughout the rest of this paper, we are interested in those images which can be characterized by *Büchi automata*, which define the so-called ω regular languages. Since a real number might have two possible encodings (e.g., "0.1000000..." and "0.0111111..." both represent $1/2$ in binary), we restrict ourselves to languages which include either both the encodings of a real number or none of them.

A *nondeterministic Büchi automaton* is a 5–tuple $B{=}(\Sigma, S, \delta, s_0, F)$, where Σ is a finite set of *input symbols*, S is a finite set of *states*, s_0 $(\in S)$ is the *initial state*, δ $(\subseteq S \times \Sigma \times S)$ defines the *transition relation*, and F $(\subseteq S)$ is the set of *final states*. A *deterministic Büchi automaton* is a Büchi automaton whose transition relation is restricted to a function $\delta : S \times \Sigma \to S$. Notice that nondeterministic Büchi automata are strictly more powerful than their deterministic counterparts. Let $B = (\Sigma, S, \delta, s_0, F)$ be a Büchi automaton and $\sigma = \omega_1\omega_2\ldots$ be an ω–word over Σ. A *run* of B on $\sigma = \omega_1\omega_2\ldots$ is an infinite sequence of states $r = r_0 r_1 \ldots$ such that $r_0 = s_0$ and $(r_i, w_{i+1}, r_{i+1}) \in \delta$, for $i \geq 0$. A run is said to be *successful* (or *accepting*) if there exist infinitely many $i \geq 0$ such that $r_i \in F$. B accepts ω–word σ if B has a successful run on σ. The set $L(B) = \{\sigma \in \Sigma^\omega \mid B \textit{ accepts } \sigma\}$ If there is a Büchi automaton B such that $L = L(B)$, L is said to be Büchi *recognizable*. A *generalized Büchi automaton* is a 5-tuple $(\Sigma, S, \delta, s_0, F)$, where Σ, S, δ and s_0 are identical to that of a Büchi automaton, and $F = \{F_1, ..., F_k\}$ (for some k) such that $F_i \subseteq S, 1 \leq i \leq k$. A *run* is successful (accepting) if for each $1 \leq i \leq k$, some state in F_i appears infinitely often in the run. It is known that the classes of languages recognized by generalized Büchi automata and Büchi automata are identical [13]. Unless stated otherwise, Büchi automata are assumed to be nondeterministic throughout this paper.

In our subsequent discussion, Büchi automata are also viewed as directed graphs, in which nodes and edges represent states and transitions, respectively. The *initial state* is annotated by an incoming arrow, whereas the *final state* is highlighted by a double circle. Given two nodes u and v in a directed graph G, we write $u \hookrightarrow v$ to denote the existence of a path from u to v in G. A directed graph (V, E) is *strongly connected* if for every pair of nodes u and v, $u \hookrightarrow v$ and $v \hookrightarrow u$. A *strongly connected component* G' of a directed graph G is a strongly connected subgraph which is maximal (i.e., no other strongly connected subgraph in G properly contains G'). A subgraph (V', E') of a directed graph (V, E) is an *end component* if the subgraph is strongly connected and if $u \in V'$ and $(u, v) \in E$, then $v \in V'$ as well. (Notice that a node without any outgoing edges is considered an end component.)

3 Bi-level images and Büchi automata

In this section, we focus on those bi-level images that can be charactered by Büchi automata, as well as on how conventional operations in image processing can be carried out in the framework of formal languages. To give the reader a better feel for how Büchi automata can be used to 'compress' images, consider an example illustrated by Figures 4 and 5, in which a line $y = \frac{1}{2}x$ (which contains infinitely many points) is 'encoded' in a succinct fashion by a Büchi automaton B (which consists of only two states). Notice that the point $\begin{bmatrix} x=001101110101010\cdots \\ y=000110111010101\cdots \end{bmatrix}$ is in $L(B)$ (i.e., on the line), but $\begin{bmatrix} x=1\cdots \\ y=1\cdots \end{bmatrix}$ is not. In fact, it is easy to see that for a point to be on the line, the first component x equals the left shift of the second component y (because $x = 2y$).

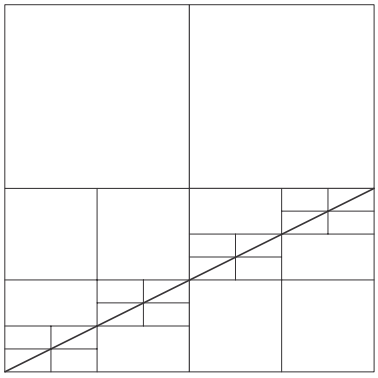


Fig. 4. A line $y = \frac{1}{2}x$,

In view of the above example, a natural question to ask is whether a more general form of lines can be encoded using Büchi automata. A recent result ([2]) answers the above in the affirmative. More precisely, we have

Theorem 3.1 *([2]) For arbitrary $\vec{a} \in Z^2$ and $b \in Q$, the set of strings $\vec{x}$ (over $\Sigma = \{0, 1\}^2$) satisfying $\vec{a} \cdot \vec{x} = b$ is recognizable by a Büchi automaton.*

The interested reader is referred to [2] for the details of the proof.

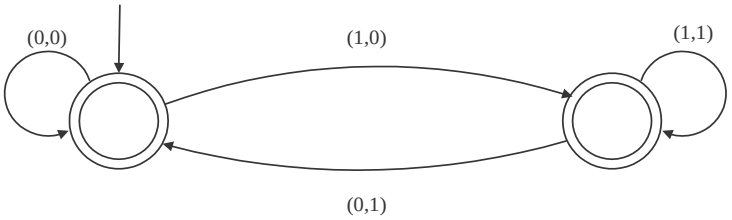


Fig. 5. A Büchi automaton B representing the line $y = \frac{1}{2}x$.

For an encoding scheme to be useful, it must be able to support various image processing operations effectively. To this end, our subsequent discussion will reveal the applicability of our ω-finite automata scheme to those basic image processing operations summarized in Figure 6.

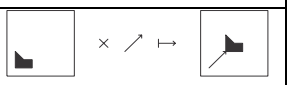
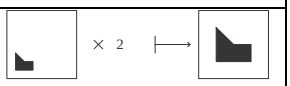
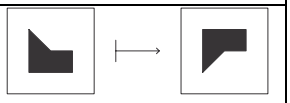
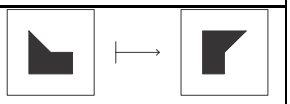
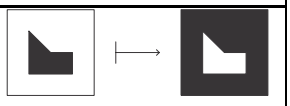
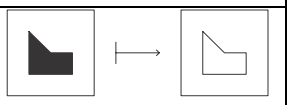
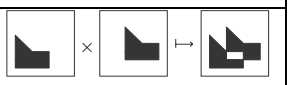
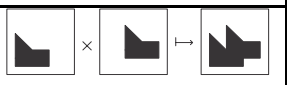
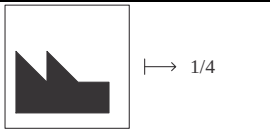
name	notation	definition	meaning
shift	sh	$sh : Image \times Q^2 \mapsto Image$	
resize	rs	$rs : Image \times 2^k \mapsto Image$	
flip	fl	$fl : Image \mapsto Image$	
rotation	ro	$ro : Image \mapsto Image$	
complement	comp	$comp : Image \mapsto Image$	
boundary	bn	$bn : Image \times Q \mapsto Image$	
difference	diff	$diff : Image \times Image \mapsto Image$	
union	un	$un : Image \times Image \mapsto Image$	
intersection	inter	$inter : Image \times Image \mapsto Image$	
size	size	$size : Image \mapsto R$	

Fig. 6. List of basic operations in image processing. (The rational number in the definition of *boundary* specifies the 'thickness' of the boundary. The size operation applies to images representable by deterministic Büchi automata only.)

In order to deal with image processing operations in the framework of ω-automata, we require a few results concerning the closure properties of the languages accepted by Büchi automata. The first one is a well known result. See [3] (also [13]).

Theorem 3.2 *The class of languages recognized by Büchi automata is closed under union, intersection, difference, and complement.*

Note, however, that the class of languages recognized by deterministic Büchi automata is not closed under complement.

Given two infinite strings ω and ω' over the alphabet $\{0,1\}^2$, we write $\omega+\omega'$ to denote the string encoding the sum of the numbers represented by ω and ω'. We also write $L+L'$ $(L, L' \subseteq \Sigma^\omega)$ to denote $\{\omega+\omega' \mid \omega \in L, \omega' \in L'\}$. In what follows, we show that the 'sum' of two Büchi recognizable images remains Büchi recognizable.

Theorem 3.3 *Given two* Büchi *automata* $B' = (\{0,1\}, S', \delta', s_0', F')$ *and* B'' $=(\{0,1\}, S'', \delta'', s_0'', F'')$*, a generalized Büchi automaton* B *can be constructed to recognize* $L = \{\omega'+\omega'' \mid \omega' \in L(B'), \omega'' \in L(B')\}$*, i.e.,* $L = L(B')+L(B'')$.

Proof: (Sketch) To a certain extent, our construction is a modification of the so-called 'product automata' approach which has many applications in automata theory (See [10]). A state in the constructed automaton B is a three-tuple (r_1, r_2, r_3), where r_1 and r_2 represent states of B' and B'', respectively, and $r_3 (\in \{0,1\})$ is a flag used for recording the 'carry bit' of the summation up to the position associated with state pair (r_1, r_2). A transition between two states simulates an addition of two digits. To give the reader a better feel for how the construction functions, consider Figure 7 in which the sum of 010111... and 011111... (i.e., the real numbers 0.010111... and 0.011111... respectively which represented by fragments of automata shown in Figures 7(a)) is performed. The corresponding fragment of the constructed automaton is depicted in Figure 7(b). Take the transition $(c, w, 1) \xrightarrow{0} (d, x, 1)$ for example (see Figure 7(c)). What it means is that if the summation up to state pair (d, x) has a carry, then adding the two bits associated with transitions $c \xrightarrow{0} d$ and $w \xrightarrow{1} x$, together with the carry-in bit, will result in a carry-out (to the left), while the resulting bit is 0.

Now we are in a position to describe the automaton B which accepts L. B=$(\{0,1\}, S, \delta, s_0, F)$, where

(i) $s_0 = (s_0', s_0'', 0)$,

(ii) $S = \{(r_1, r_2, r_3) \mid r_1 \in S',\ r_2 \in S''$ *and* $r_3 \in \{0,1\}\}$,

(iii) The transition relation δ is defined as follows: For each $(r_1, r_2, r_3) \in S$, $\delta'(r_1, a') \in S'$, $\delta''(r_2, a'') \in S''$, we have $(\delta'(r_1, a'), \delta''(r_2, a''), r_3') \in \delta((r_1, r_2, r_3), a)$, provided that $r_3 = (a' + a'' + r_3')$ *mod* 2 and a is the remainder of $(a' + a'' + r_3')$ divided by 2, and

(iv) $F = \{F' \times S'', S' \times F''\}$.

The correctness of the construction is reasonably straightforward. □

Even though the above theorem deals with the alphabet $\{0,1\}$, it is straightforward to generalize the result to the alphabet $\{0,1\}^2$. Hence we have

Corollary 3.4 *The* sum *of two Büchi recognizable images remains Büchi recognizable.*

Corollary 3.5 *Given a* Büchi *automaton* $B = (\{0,1\}^2, S, \delta, s_0, F)$ *and* $\vec{x} \in Q^2$*, we can construct a* Büchi *automaton* B_{+x} *to accept* $L = \{l + \omega \mid l \in L(B)\}$*, where* ω *encodes* $\vec{x}$*.*

Proof: Clearly, ω (which encodes $\vec{x}$) can be accepted by a Büchi automaton, our result then follows immediately from Corollary 3.4. □

As we shall see later, the shift operation is carried out based upon the result of Corollary 3.5. The reader is referred to [4] for a similar result based on the theory of *affine transformation.*

In some image processing applications, the ability to calculate certain geometric properties, such as *area* (i.e., *size*), is important. Given a deterministic Büchi automaton which represents an image, we now propose a method to calculate the *size* of the image area. To this end, we use the theory of *Markov chains* to capture the essence of image *sizes.*

A *Markov chain* M is a 4-tuple (S, P, s_1, F), where S is the set of states, s_1 $(\in S)$ is the initial state, $P : S \times S \rightarrow [0,1]$ defines the transition probability function satisfying the condition that for a given state, the sum of its outgoing probabilities equals one, and F is the set of accepting states. The *probability measure* of a sequence of states $\sigma = s_1, ..., s_k$, denoted by $Pr(\sigma)$, is $P(s_1, s_2) \times \cdots \times P(s_{k-1}, s_k)$. Such a probability measure can be extended to the set $ACCEPT(M)$=$\{\sigma \mid \sigma$ is an infinite computation from s_1 which visits some state in F infinitely many times$\}$ using the theory of Markov chains in a standard way. We define $Pr(M)$ = the probability measure of $ACCEPT(M)$. The reader is referred to [11] for more background on probability theory and Markov chains.

The idea of using Markov chains to capture the sizes of images (encoded by deterministic Büchi automata) is illustrated in Figure 8, in which an image (Figure 8(a)) is represented by a deterministic Büchi automaton (Figure 8(b)). Now a Markov chain, as illustrated in Figure 8(c), is constructed in such a way that the 'probability' along a transition of the Markov chain reflects the ratio of the image sizes before and after the transition is taken. For example, the transition $a \stackrel{\frac{1}{4}}{\rightarrow} b$ in Figure 8(c) has probability $\frac{1}{4}$, for the associated transition $a \stackrel{(0,0)}{\rightarrow} b$ in Figure 8(b) refers to the lower left-hand sub-image, whose size is one quarter of the original image. The transformation of a deterministic Büchi automaton to the corresponding Markov chain is straightforward, and the details are left to the reader. Now the *size* of an image is defined to be the accepting probability of the corresponding Markov chain which models the

image. (Recall that in this paper we only consider images in the unit square area $[0,1] \times [0,1]$.)

For ease of expression, an ordering is given to the set of states S, i.e., $S = \{s_1, s_2, ..., s_m\}$. The one–step transition probability is organized into a one–step transition matrix

$$\mathbf{P} \equiv \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mm} \end{bmatrix}$$

in which $p_{ij} = P(s_i, s_j), 1 \leq i, j \leq m$. Notice that for each row i, $\sum_{j=1}^{m} p_{ij} = 1$. A Markov chain is *irreducible* if none of its subsets of states also forms a Markov chain (i.e., none of its subgraphs is an end component). States that are members of an irreducible set are called *recurrent*, and the remaining states are *transient*. We let $\mathcal{T}$ be the set of transient states and $\mathcal{R}$ be the set of recurrent states. Take Figure 8 for example. In Figure 8(c), states c and d are recurrent, while the rest are transient.

To compute the accepting probability of a Markov chain, we require the following lemma.

Lemma 3.6 *Given a Markov chain $M = (S, s_1, P, F)$ and an end component $S'(\subseteq S)$ with $S' \cap F \neq \emptyset$, then for each $r \in S'$, $Pr\{\sigma \mid$ run $\sigma = rr_1r_2\cdots$, encounters some state in F infinitely often$\}$=1. (That is, the set of 'accepting' computations from r has probability 1.)*

Proof: (Sketch) Suppose, to the contrary, that there were an $r \in S'$ such that the probability measure of the set of accepting computations from r is less than 1. Given a $D \subseteq S'$, let C_D be the set of infinite computations from r such that along each of such computations, the set of states that appear infinitely many times is exactly D. Since there are only finitely many Ds, a $D' \subseteq S'$ with $D' \cap F = \emptyset$ and $Pr(C_{D'})$ (i.e., the probability measure of $C_{D'}$) > 0 must exist. Clearly, nodes in D' (together with their incident edges) must form a strongly connected component. We claim that D' is also an end component. If this is not the case, there must be an edge leaving D', and hence, the probability for the computation to stay in D' forever is zero (a known result which is relatively easy to show). As a result, D' must be identical to S' (otherwise, it is impossible for both D' and S' to be end components) – contradicting the assumption that $D' \cap F = \emptyset$. This completes the proof. □

With the help of Lemma 3.6, in order to find $Pr(M)$ for a given Markov chain M, it suffices to compute the probability of reaching those end components that contain some accepting states. Such a problem is known as the *probabilistic reachability problem* in the theory of Markov chains. Given two states i and j, let $f_{i,j}^{(n)}$ be the probability of reaching j from i in no more than

n steps. Let $f_{i,j}^{(*)} = \lim_{n\to\infty} f_{i,j}^{(n)}$. Using a known result concerning Markov chain [11], $\mathbf{F}^*_{\mathcal{TR}} = (f_{i,j}^{(*)})_{i\in\mathcal{T}, j\in\mathcal{R}}$ can be computed as follows:

$$\mathbf{F}^*_{\mathcal{TR}} = (\mathbf{I} - \mathbf{P}_{\mathcal{TT}})^{-1}\mathbf{P}_{\mathcal{TR}},$$

where $\mathbf{P}_{\mathcal{TT}}$ stands for the one–step transition probabilities between states in $\mathcal{T}$ and $\mathbf{P}_{\mathcal{TR}}$ represents the one–step transition probabilities from states in $\mathcal{T}$ to states in $\mathcal{R}$.

Using Lemma 3.6, we have

Theorem 3.7 *Given a Markov chain $M = (S, P, s_1, F)$,*

$$Pr(M) = \sum_{j\in\bigcup_{i=1}^{k}\mathcal{I}_i} f_{s_1,j}^{(*)} \quad ,$$

where $\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k\}$ is the set of all irreducible sets (i.e., end components) such that $\mathcal{I}_i \cap F \neq \emptyset$, for all $1 \leq i \leq k$.

With the above theorem, in conjunction with the close connection between deterministic Büchi automata (encoding bi-level images) and Markov chains, the size of a deterministic Büchi recognizable image can effectively be measured.

For example, consider the image shown in Figure 8(a) for which the transition matrix of the corresponding Markov chain is the following:

$$\mathbf{P} \equiv \begin{bmatrix} 0 & \frac{1}{4} & \frac{3}{4} & 0 \\ 0 & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In this example, $\mathcal{T} = \{a, b\}$, $\mathcal{R}_1 = \{c\}$, $\mathcal{R}_2 = \{d\}$. Since $\mathcal{R}_2$ is the only irreducible set which contains an accepting state, the size of the image equals $f_{a,d}^{(*)}$, which can be computed in the following way:

$$\begin{aligned} \mathbf{F}_{\mathcal{TR}_2} &= \begin{bmatrix} f_{a,d}^{(*)} \\ f_{b,d}^{(*)} \end{bmatrix} \\ &= (I - P_{\mathcal{TT}})^{-1} P_{\mathcal{TR}_2} \\ &= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & \frac{1}{4} \\ 0 & \frac{1}{2} \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ \frac{1}{4} \end{bmatrix} \\ &= \left(\begin{bmatrix} 1 & -\frac{1}{4} \\ 0 & \frac{1}{2} \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ \frac{1}{4} \end{bmatrix} \end{aligned}$$

$$= \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{1}{4} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{8} \\ \frac{1}{2} \end{bmatrix}$$

Thus, the size of the black area in Figure 8(a) equals $f^{(*)}_{a,d} = \frac{1}{8}$.

4 Implementations of image operations

In this section, we show how each of the image processing operations defined in Figure 6 can be performed in the framework of Büchi automata. Even though operations such as *shift, resize, flip, rotation* and *union*, were initially studied in [4], they are also listed below for the sake of completeness.)

Theorem 4.1 *The set of operations listed in Figure 6 can be implemented effectively. (Notice that the* size *operation applies to images representable by deterministic Büchi automata only.)*

Proof:

• *shift: image* $\times Q^2 \rightarrow$ *image*

Given an image (represented by a Büchi automaton B) and a vector $\vec{x} \in Q^2$, Corollary 3.5 shows the feasibility of shifting the image encoded by B by vector $\vec{x}$.

• *resize: image* $\times 2^k \rightarrow$ *image,* $k \in Z$

Let $B = (\{0,1\}^2, S, \delta, s_0, F)$ be an automaton accepting the input image. It suffices to show the cases for k=1 and -1. First consider $k = 1$ (i.e., enlarging the image by a ratio of 2 along both x and y axes). Suppose $s_0 \stackrel{r}{\rightarrow} s_r, r \in \{0,1\}^2$ (i.e., $s_r, r \in \{0,1\}^2$, represent the four immediate successors of s_0). It is not hard to observe that $B_{(0,0)} = (\{0,1\}^2, S, \delta, s_{(0,0)}, F)$ encodes an image which is the enlargement of the $(0,0)$-subimage by a ratio of 2×2. See Figure 9. Then it is reasonably easy to see that $resize(B, 2) = B_{(0,0)}$. (Notice that the enlargements of the remaining $(0,1), (1,0), (1,1)$-subimages are beyond the boundary of the $[0,1] \times [0,1]$ area.) Now consider $k = -1$ (i.e., shrinking the image by a ratio of 2 along both x and y axes). We define $B_{\frac{1}{2}} = (\{0,1\}^2, S \cup \{s'_0\}, \delta', s'_0, F)$ such that $\delta' = \delta \cup \{s'_0 \stackrel{(0,0)}{\rightarrow} s_0\}$. (That is, $B_{\frac{1}{2}}$ is obtained from B by adding a new initial state s'_0 together with transition $s'_0 \stackrel{(0,0)}{\rightarrow} s_0$.) $B_{\frac{1}{2}}$ clearly encodes $resize(B, \frac{1}{2})$.

• *flip: image* $\rightarrow$ *image*

By interchanging the symbol $(0,0)$ with symbol $(0,1)$ and symbol $(1,0)$ with symbol $(1,1)$ in the ω-automaton encoding the input image M, the resulting automaton encodes $flip(M)$ along the x-axis. Flipping along the y-axis is done similarly.

• *rotate: image* $\times\{90^o, 180^o, 270^o\} \rightarrow$ *image*
It suffices to consider rotating the input image 90^o in the clockwise fashion. Like the flipping case, 90^o rotation can be achieved by interchanging the four input symbols in a circular fashion $(0,0) \rightarrow (0,1) \rightarrow (1,1) \rightarrow (1,0) \rightarrow (0,0)$.
• *complement, difference, union, intersection*
According to Theorem 3.2, the class of languages accepted by Büchi automata is closed under complement, difference, union, intersection, implying the feasibility of the associated image processing operations.
• *boundary: image* $\times Q \rightarrow$ *image*
(Recall that the rational number in the operation specifies the 'thickness' of the boundary.) Let F be a Büchi automaton accepting the image $[0,b] \times [0,b]$, for a desired thickness $b \in Q$ Given an image B', we extend B' by constructing a Büchi automaton B that recognizes $ext(B) = L(B') + L(F)$ (guaranteed by Corollary 3.5). See Figure 10(a)-(c). By repeatedly rotating the image and then performing the above shift operation, the boundary of B' can be computed as $diff(B, un(un(un(ext(B), ext(flip(ro(ro(B))))), ext(flip(B))), ext(ro(ro(B)))))$. See Figure 10 for a series of such operations and how the boundary of an image (with the desired thickness) is extracted.
• *size: image* $\rightarrow R$
The size of a deterministic Büchi recognizable image can easily be computed as the consequence of Theorem 3.7. □

Two given images F and F' are said to be *similar* within error bound ϵ if *area(diff(F,F'))* $\leq \epsilon$. In this case, we write $F \sim_\epsilon F'$. In what follows, we show the connection between the image compression approaches based upon the conventional finite automata and the ω- finite automata.

Theorem 4.2 *Given a finite automaton A, an ω-finite automaton B can be constructed such that $Img(A) \sim_0 Img(B)$, where $Imag(A)$ and $Img(B)$ are the images represented by A and B, respectively. (That is, the two images are identical.)*

Proof: Let $A = (\Sigma, S', \delta', s_0, F)$. B is constructed as $(\Sigma, S, \delta, s_0, F)$, where $\delta = \delta' \cup \{(s,a,s) | s \in F\}$ (i.e., for each of the final state s, a self loop is attached for every input symbol a). The correctness of the construction is rather obvious. □

By unwinding the computation of an ω-finite automaton to the desired depth specified by the error bound, the following result is relatively easy to obtain. Due to space limitations, the details are omitted.

Theorem 4.3 *Given an ω-finite automaton A and an error bound ϵ, a finite automaton B can be constructed such that $Img(A) \sim_\epsilon Img(B)$, where $Imag(A)$ and $Img(B)$ are the images represented by A and B, respectively.*

References

[1] Luca de Alfaro, *Formal Verification of Probabilistic Systems*, Ph.D.Dissertation, Department of Computer Science Stanford University, December 1997.

[2] B. Boigelot, S. Rassart and P. wolper, On the Expressiveness of Real and Integer Arithmetic Automata, *Proc. 25th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science, Vol. 1443,* Springer-Verlag, pp. 152-163, 1998.

[3] J. R. Büchi, On a Decision Method in Restricted Second Order Arithmetic, *Proc. of the International Congress on Logic, Method, and Philosophy of Science,* pp. 1–12, Stanford University Press, Stanford, CA, USA, 1962.

[4] K. Culik and S. Dube, Rational and Affine Expressions for Image Description, *Discrete Applied Mathematics,* Vol. 41, pp. 85-120, 1993.

[5] K. Culik and J. Kari, Image Compression Using Weighted Finite Automata, *Computers and Graphics,* Vol. 17, No. 3, pp. 305–313, 1993.

[6] K. Culik and J. Karhumaki, Finite Automata Computing Real Functions, *SIAM J. on Computing,* Vol. 23, No. 4, pp. 789-814, 1994.

[7] K. Culik and J. Kari, Finite–State Transformations of Images, *Computer and Graphics.* Vol. 34, pp. 151-166, 1997.

[8] K. Culik and V. Valenta, Finite Automata Based Compression of Bi-Level and Simple Color Images, *Computer and Graphics,* Vol. 21, pp. 61-68, 1997.

[9] H. Freeman, Computer Processing of Line–Drawing Images, *ACM Computing Surveys,* Vol. 6, No. 1, pp. 57-97, 1974.

[10] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[11] B. Nelson, *Stochastic Modeling: Analysis and Simulation,* McGraw-Hill, 1995.

[12] H. Samet, *Applications of Spatial Data Structures,* Addison-Wesley, 1993.

[13] W. Thomas, Automata on Infinite Objects, in *Handbook of Theoretical Computer Science*, Edited by J. van Leeuwen, Elsevier Science Publisher B.V., 1990.

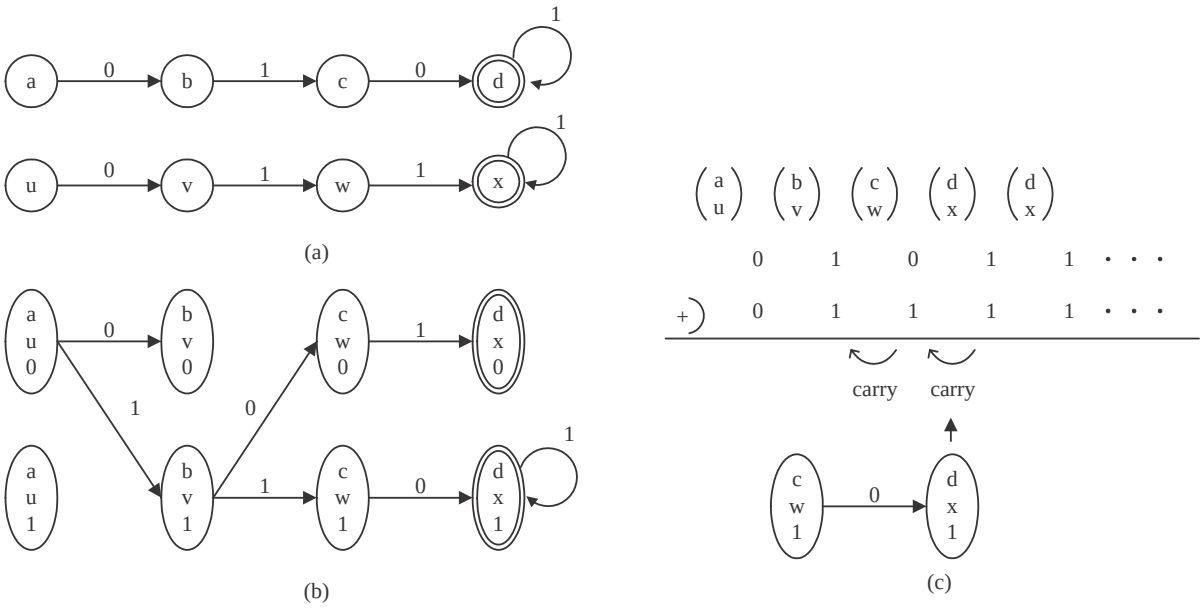


Fig. 7. The addition of two real numbers.

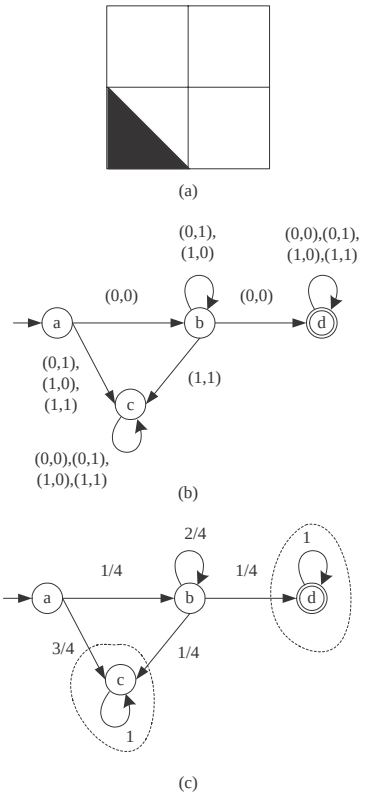
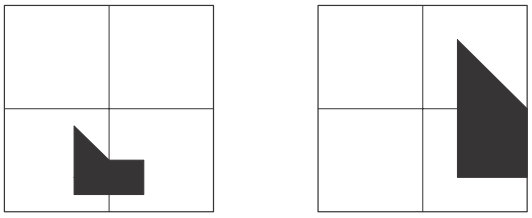
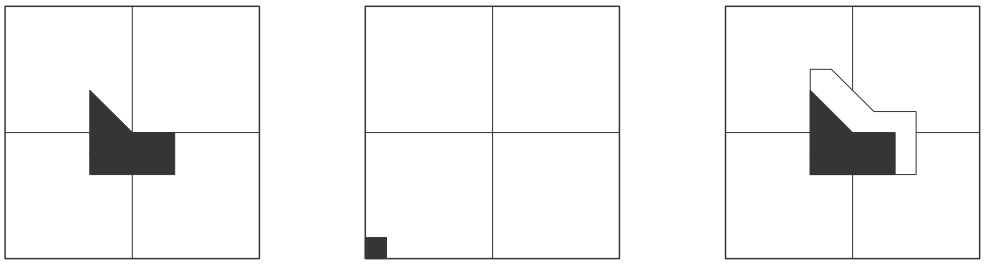


Fig. 8. (a) triangle, (b) ω–FA, (c) Markov chain.

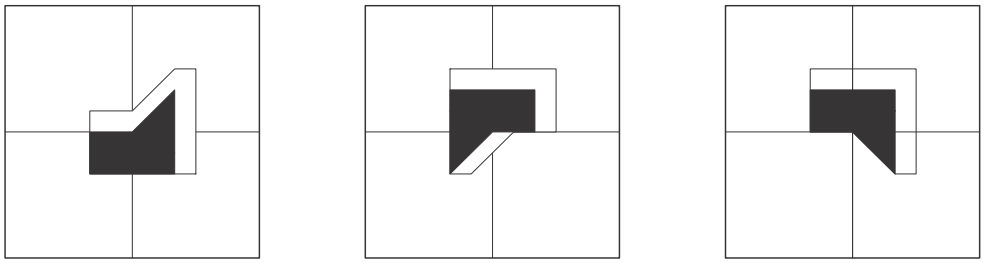


(a) (b)

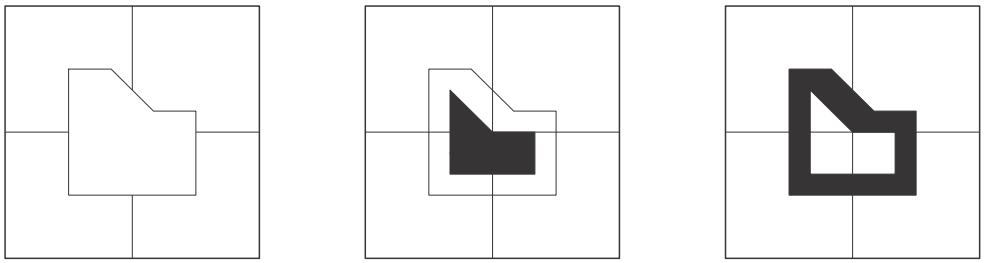
Fig. 9. The procedure of resize.



(a)image B (b)fixed image F (c)ext (B)



(d)ext(flip(ro(ro(B)))) (e)ext(flip(B)) (f)ext(ro(ro(B)))



(g)union of (b), (c), (d), (e) and (f) (h)diff((g), (a)) (i)boundary

Fig. 10. Computing the boundary of an image.