

An ω -Automata Approach to the Representation of Bilevel Images

Yih-Kai Lin and Hsu-Chun Yen

Abstract—We use ω -automata (i.e., automata over infinite words) as a device for representing bilevel images. A major advantage of our approach, as opposed to using the conventional finite automata, lies in that ω -automata are capable of representing image objects of zero size, such as lines and points. To demonstrate the feasibility of our approach, we also show how a number of image processing operations, including *shift, flip, rotation, complement, boundary, difference, union, intersection, and size*, can be effectively carried out in the framework of ω -automata. In particular, the size of an image represented by an ω -automaton is measured based on the theory of Markov chains. In comparison with other automata-based image representation schemes reported in the literature, our approach is capable of supporting a richer set of operations, which can be performed on the automata directly and easily.

Index Terms— ω -automata, image representation.

I. 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 digital image compression (see, e.g., [1]–[4]). 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 bilevel (i.e., black-and-white) images. The model of *weighted finite automata* [1] has been proposed to encode and process gray-tone images. The idea of using finite automata to encode bilevel images is the following. By recursively subdividing an image area into four quadrants, a subimage can be addressed by a string x_1, \dots, 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 subdividing 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 way that a string x is in L if and only if the corresponding subimage addressed by x is “black.” To support the applicability of such a strategy, [3] has shown that various image processing applications can be effectively performed in the framework of *weighted finite transducers*. Weighted finite transducers act as language transducers by accepting input languages (interpreting the original images) then generating target languages (interpreting the target images). Since weighted finite automata and weighted finite transducers deal with words of finite length, an infinite resolution image is interpreted as the “limit” of images defined by weighted finite automata of increasing resolution.

To deal with images of infinite resolution, an alternative approach is to use ω -finite transducers [5] to process bilevel images. An ω -finite transducer defined in [5] is a device which maps a deterministic Büchi recognizable language into another deterministic Büchi recognizable language. In a language accepted by a conventional finite automaton, a string is of finite length, meaning that the corresponding

image area has nonzero 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 two-dimensional (2-D) 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 some zero-sized images (such as lines) as well as images with scattered “noises” of zero size. In a recent paper [6], the question of which images can be encoded as finite automata with infinite resolution has been investigated. One practical advantage of producing zero size objects is to present line-like images against finite automata which are designed for the representation of silhouette-like images. In [5], image operations of “zooming” and “placement” are implemented by the “quotient” and “concatenation” operations, respectively, while the affine transformation is carried out through the so-called *generalized sequential machines*.

Motivated by the work of [5] (also [4]), in this paper we focus on those bilevel images representable by a special type of ω -automata called *Büchi automata* [7], a model that has been extensively studied in the literature (see, e.g., [7] and [8]). 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 size, complement, and boundary which were not studied in [5]) are carried out in a unified framework based upon ω -automata. As we shall see later that in comparison with [5], our approach is capable of supporting a richer set of operations, and such operations are performed on the automata directly (rather than using transducers).

Among the new results is the measurement of *sizes* of infinite resolution images represented by deterministic Büchi automata. For 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. Although the computation of integrals proposed in [3] suits for counting the size of an image represented by a weighted finite automaton, in this paper, we propose a new way of measuring the size of an image based on the theory of Markov chains. More will be said about the comparison between these two approaches in Section IV. We also show the relationship between image representation schemes based on finite automata and Büchi automata. As opposed to the transducer-based images operations, each of our image operations is performed on the automata directly.

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

II. 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, over Σ , respectively. An ω -word over Σ is an infinite string written in the form $\omega_1\omega_2\dots$ where $\omega_i \in \Sigma$. Let \mathbb{R} denote the set of real numbers, \mathbb{Q} the set of rational numbers and \mathbb{Z} the set of integers. Let \mathbb{R}^k denote the set of vectors of k real numbers, \mathbb{Q}^k the set of vectors of k rational numbers and \mathbb{Z}^k the set of vectors of k integers.

Although we mainly focus on 2-D digital images and numbers in base 2, our results can be easily extended to higher dimensions and

Manuscript received August 17, 2000; revised September 9, 2001. A preliminary version of this paper was presented at Computing: The Australian Theory Symposium 2000.

The authors are with the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: klin@cobra.ee.ntu.edu.tw; yen@cc.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/TSMCB.2003.811123

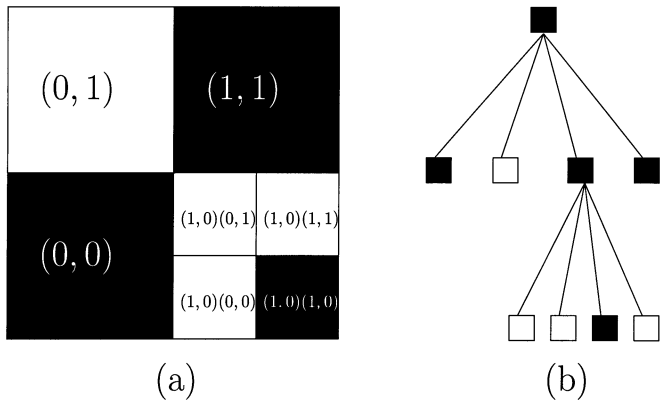


Fig. 1. Quadtree representation of an image. (a) Addresses of some subsquares and (b) the corresponding quadtree.

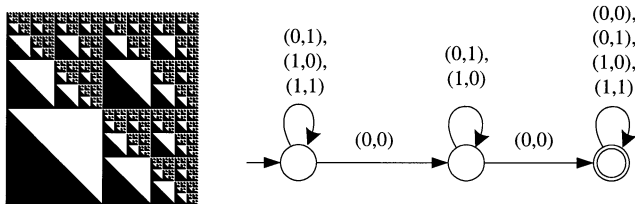


Fig. 2. Diminishing triangles and the corresponding FA.

other bases. In this paper, the locations of image pixels are referred to by their quadtree addresses in the following way. A quadtree representation of an image is based on the successive subdivision of the given image into four quadrants. The four quadrants form the subtrees of the root representing the original image. The formal definition of quadtrees can be found in [9]. 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 Fig. 1 illustrates the addresses of several subsquares and 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 [1]–[4]. The encoding is done in 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.” Fig. 2 demonstrates a picture of diminishing triangles (given in [4]) together with the corresponding finite automaton. Fig. 3 illustrates the finite automaton “approximating” a line, and clearly, by using the multi-resolution concept, 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 of a finite automaton not only accepts strings of finite length, and each string represents a square of nonzero size. To overcome this shortcoming, in this paper we use ω -finite automata (which accepts strings of infinite length) to encode bi-level images. Related results can be found in [5].

Without loss of generality, 2-D square images are assumed to be normalized in the sense that the ranges of 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 $p = (x = x_12^{-1} + x_22^{-2} + x_32^{-3} + \cdots, y = y_12^{-1} + y_22^{-2} + y_32^{-3} + \cdots) \in [0, 1]^2$, the corresponding quadtree address is $(x_1, y_1)(x_2, y_2)(x_3, y_3) \dots$, where

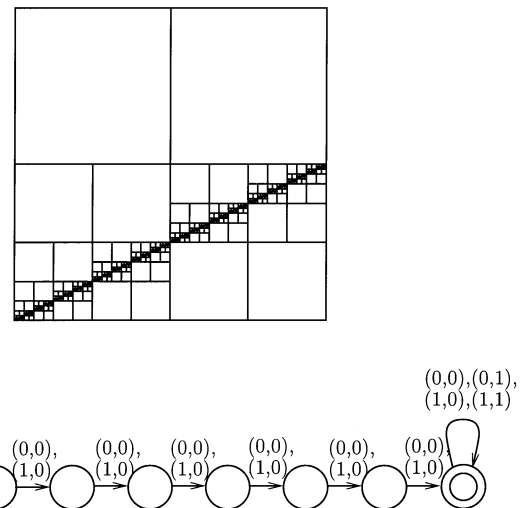
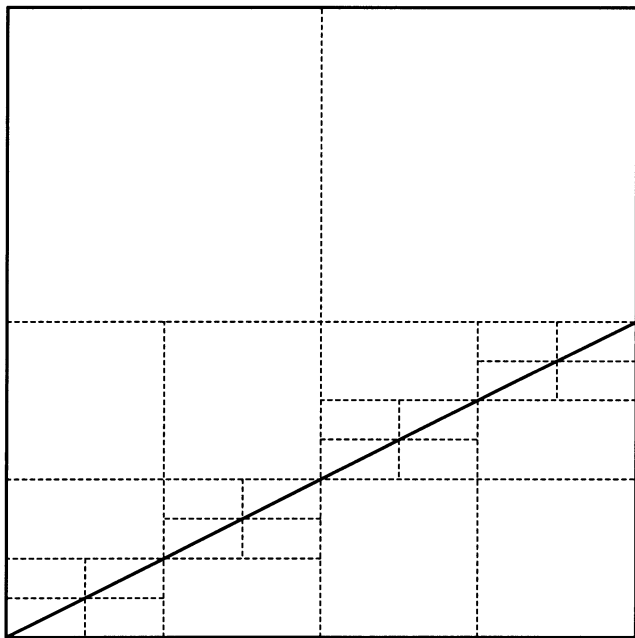


Fig. 3. Approximation of a slope line and the corresponding FA.

$(x_i, y_i) \in \Sigma^2$. For example, point $(1/3, 1/3)$ can be mapped into “ $(0, 0)(1, 1)(0, 0)(1, 1)(0, 0)(1, 1) \dots$ ” 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*, $\delta (\subseteq S \times \Sigma \times S)$ defines the *transition relation*, and $F (\subseteq S)$ is the set of *final states*. Note that Büchi automata are a special type of ω -automata. A *deterministic Büchi automaton* is a Büchi automaton whose transition relation is restricted to a function $\delta: S \times \Sigma \rightarrow S$. 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 $\sigma = \omega_1\omega_2 \cdots$ be an ω -word over Σ . A *run* of B on $\sigma = \omega_1\omega_2 \cdots$ is an infinite sequence of states $r = r_0r_1 \cdots$ such that $r_0 = s_0$ and $(r_i, \omega_{i+1}, r_{i+1}) \in \delta$, 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 . Let $L(B) = \{\sigma \in \Sigma^\omega | B \text{ accepts } \sigma\}$. 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 $F = \{F_1, \dots, F_k\}$ (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 [8]. 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 states* are highlighted by double circles. 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')

Fig. 4. Line $y = (1/2)x$.

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.)

III. BILEVEL IMAGES AND BÜCHI AUTOMATA

In this section, we focus on those bilevel images which can be characterized 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 feeling for how Büchi automata can be used to “compress” images, consider an example illustrated by Figs. 4 and 5, in which a line $y = (1/2)x$ (which contains infinite points) is “encoded” in a succinct fashion by a Büchi automaton B (which consists of only two states). Notice that the point $\begin{bmatrix} x=001101110101010\dots \\ y=000110111010101\dots \end{bmatrix}$ is in $L(B)$ (i.e., on the line), but $\begin{bmatrix} x=1\dots \\ y=1\dots \end{bmatrix}$ 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 $x = 2y$).

In view of the above example, a natural question to ask is whether a more general form of lines can be encoded by Büchi automata. A recent result [10] answers the above in the affirmative. More precisely, we have the following.

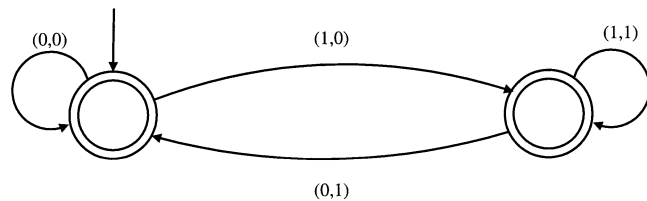
Theorem 1 [10]: For arbitrary $\vec{a} \in \mathbb{Z}^2$ and $b \in \mathbb{Q}$, the set of strings \vec{x} (over $\Sigma = \{0, 1\}^2$) satisfying $\vec{a} \cdot \vec{x} = b$ is recognizable by a Büchi automaton where “ \cdot ” is the inner product.

What Theorem 3.1 says is that given an equation $\vec{a} \cdot \vec{x} = b$, there exists a Büchi automaton recognizing the set of strings encoding \vec{x} . The interested reader is referred to [10] for details of the proof.

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

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 [7] and [8]).

Theorem 2: The class of languages recognized by Büchi automata is closed under union, intersection, difference, and complement.

Fig. 5. Büchi automaton B representing the line $y = (1/2)x$.

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: 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 [11]). A state in the constructed automaton B is a triple (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 feeling for how the construction functions, consider Fig. 6 in which the sum of 010111... and 011111... is performed. Here, the real numbers 0.010111... and 0.011111... are represented by the automata shown in Fig. 6(a). The corresponding fragment of the constructed automaton is depicted in Fig. 6(b). Take the transition $(c, w, 1) \xrightarrow{0} (d, x, 1)$ for example [see Fig. 6(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 $L = L(B') + L(B'')$, where $L = (\{0, 1\}, S, \delta, s_0, F)$, where

- 1) $s_0 = (s'_0, s''_0, 0)$;
- 2) $S = \{(r_1, r_2, r_3) \mid r_1 \in S', r_2 \in S'' \text{ and } r_3 \in \{0, 1\}\}$;
- 3) 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 $r_3 = (a' + a'' + r'_3) \bmod 2$ and a is the remainder of $(a' + a'' + r'_3)$ divided by 2;
- 4) $F = \{F' \times S'' \times \{0, 1\}, S' \times F'' \times \{0, 1\}\}$.

The correctness of the construction is 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 the following.

Corollary 1: The sum of two Büchi recognizable images remains Büchi recognizable.

Corollary 2: Given a Büchi automaton $B = (\{0, 1\}^2, S, \delta, s_0, F)$ and $\vec{x} \in \mathbb{Q}^2$, we can construct a Büchi automaton $B_{+\vec{x}}$ 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 1. ■

As we shall see later, the shift operation is carried out based upon the result of Corollary 2. The reader is referred to [5] how the theory of *affine transformation* is used for implementing the shift operation.

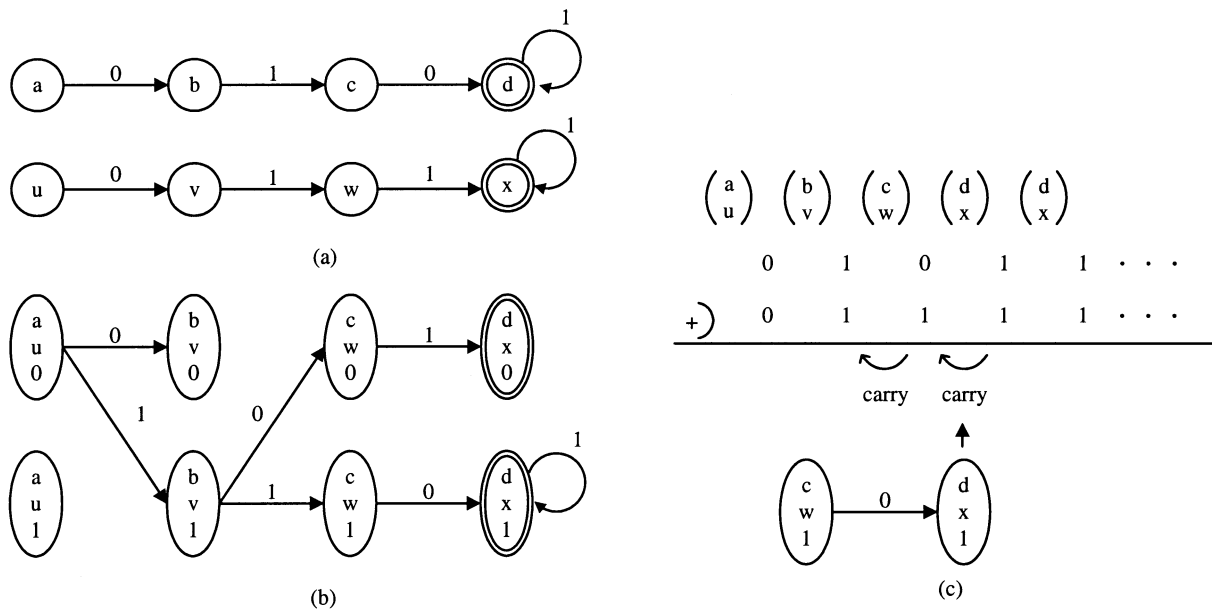


Fig. 6. Addition of two real numbers. (a) Automata representing the two numbers, (b) fragment of the constructed automaton, and (c) the concept of carry-in.

In some image processing applications, the ability to calculate certain geometric properties, such as *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. For 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 $\sigma = s_1, \dots, s_k$, denoted by $\Pr(\sigma)$, is $P(s_1, s_2) \times \dots \times P(s_{k-1}, s_k)$. Such a probability measure can be extended to the set $ACCEPT(M) = \{\sigma | \sigma \text{ is an infinite computation from } s_1 \text{ which visits some state in } F \text{ infinitely many times}\}$ 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 [12] 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 Fig. 7, in which an image [Fig. 7(a)] is represented by a deterministic Büchi automaton [Fig. 7(b)]. Now a Markov chain, as illustrated in Fig. 7(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 $a \xrightarrow{1/4} b$ in Fig. 7(c) has probability $1/4$, for the associated transition $a \xrightarrow{(0,0)} b$ in Fig. 7(b) refers to the lower left-hand subimage, 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., $S = \{s_1, s_2, \dots, s_m\}$. The one-step transition probability is organized into a one-step transition matrix

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

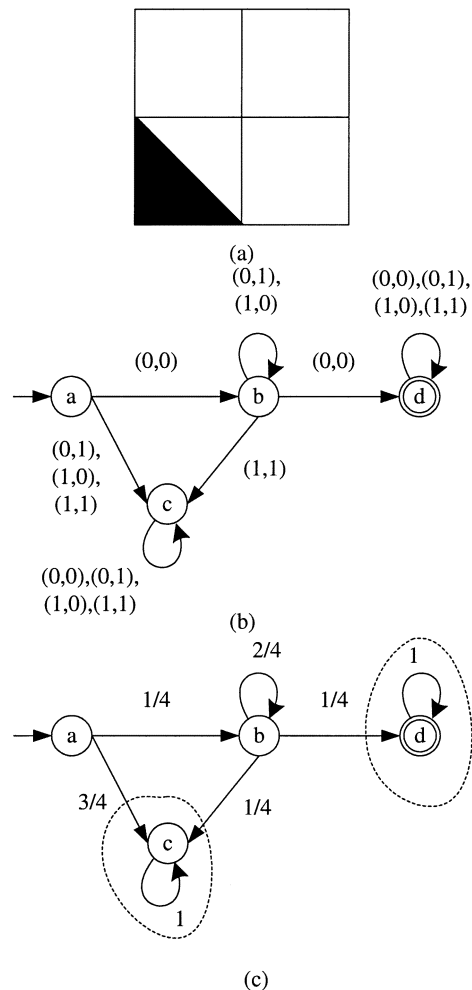


Fig. 7. How to compute “size?” (a) Triangle, (b) ω -FA, and (c) Markov chain.

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 Fig. 7 for example. In Fig. 7(c), states c and d are recurrent, while the rest [(a) and (b)] are transient, i.e., a and b are not recurrent.

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

Lemma 1: 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 $r \in S'$, $\Pr, \{\sigma \mid \text{run } \sigma = r r_1 r_2 \dots, \text{ encounters some state in } F \text{ infinitely often}\} = 1$. (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 D s, a $D' \subseteq S'$ with $D' \cap F = \emptyset$ and $\Pr(C_{D'}) > 0$ 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 1, 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 $f_{i,j}^{(*)} = \lim_{n \rightarrow \infty} f_{i,j}^{(n)}$. Using a known result concerning Markov chain [12], $\mathbf{F}_{\mathcal{T}\mathcal{R}}^* = (f_{i,j}^{(*)})_{i \in \mathcal{T}, j \in \mathcal{R}}$ can be computed as follows:

$$\mathbf{F}_{\mathcal{T}\mathcal{R}}^* = (\mathbf{I} - \mathbf{P}_{\mathcal{T}\mathcal{T}})^{-1} \mathbf{P}_{\mathcal{T}\mathcal{R}}$$

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

Using Lemma 1, we have the following.

Theorem 4: Given a Markov chain $M = (S, P, s_1, F)$

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

where $\{\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_k\}$ 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 bilevel images) and Markov chains, the size of a deterministic Büchi recognizable image can be effectively measured.

For example, consider the image shown in Fig. 7(a) for which the transition matrix of the corresponding Markov chain is the following:

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

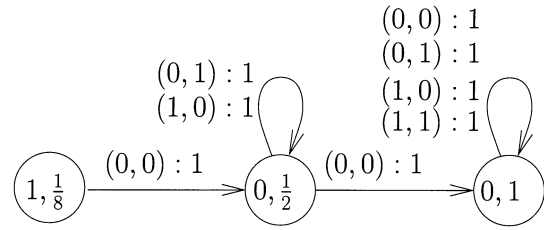


Fig. 8. Weighted finite automaton encoding the image shown in Fig. 7(a).

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:

$$\begin{aligned} \mathbf{F}_{\mathcal{T}\mathcal{R}_2} &= \begin{bmatrix} f_{a,d}^{(*)} \\ f_{b,d}^{(*)} \end{bmatrix} \\ &= (\mathbf{I} - \mathbf{P}_{\mathcal{T}\mathcal{T}})^{-1} \mathbf{P}_{\mathcal{T}\mathcal{R}_2} \\ &= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & \frac{1}{4} \\ 0 & \frac{1}{2} \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ \frac{1}{4} \end{bmatrix} \\ &= \left(\begin{bmatrix} 1 & -\frac{1}{4} \\ 0 & \frac{1}{2} \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ \frac{1}{4} \end{bmatrix} \\ &= \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{1}{4} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{8} \\ \frac{1}{2} \end{bmatrix}. \end{aligned}$$

Thus, the size of the black area in Fig. 7(a) equals $f_{a,d}^{(*)} = 1/8$.

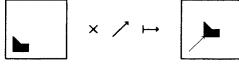
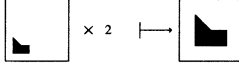
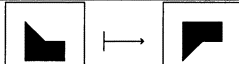
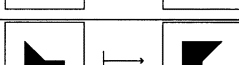
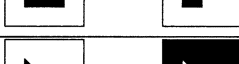
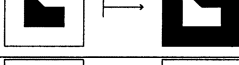
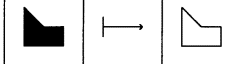



To show the differences between our model and weighted finite automata [1], [2], we encode the image shown in Fig. 7(a) into a weighted finite automaton and then compute the “integral” using weighted finite transducers. The integral of a bilevel image coincides with the size of the image.

We now briefly describe the preliminaries of weighted finite automata and weighted finite transducers (see [1], [3] for more details.) A function $f: \Sigma^* \mapsto \mathbb{R}$ is used to assign a real number, interpreted as a gray level, for each pixel in difference resolution addressed by a string in Σ^* . Function f is called *average preserving* if $\sum_{x \in \Sigma^*} f(wx) = |\Sigma| \cdot f(w)$ for all $w \in \Sigma^*$. Intuitively, function f defines serial images at different resolutions when it is average preserving. A weighted finite automaton (WFA) A can be used to define such a function f , i.e., $f(x_1, x_2, \dots, x_k) = I \cdot W_{x_1} \cdot W_{x_2} \cdots W_{x_k} \cdot F$, where $I \in \mathbb{R}^{1 \times m}$ is the initial distribution, $F \in \mathbb{R}^{m \times 1}$ is the final distribution, and $W_{x_i} \in \mathbb{R}^{m \times m}$ for all $1 \leq i \leq k$, $x_i \in \Sigma$ are weight matrices. Intuitively, function $f(w)$ is the sum of the weights of all paths labeled w . The weight of a path is multiplication of the initial distribution on the first state, weights of the transitions on the path, and the final distribution on the final state. A weight finite transducer (WFT) is a finite automaton with input alphabet Σ_1 , output alphabet Σ_2 , initial distribution $I \in \mathbb{R}^{1 \times n}$, final distribution $F \in \mathbb{R}^{n \times 1}$, and weight matrixes $W_{a,b} \in \mathbb{R}^{n \times n}$ for all $a \in \Sigma_1$ and $b \in \Sigma_2$. Given a WFT, the weighted relation $f_T: \Sigma_1^* \times \Sigma_2^* \mapsto \mathbb{R}$ is defined by $f_T(a_1 a_2 \cdots a_k, b_1 b_2 \cdots b_k) = I \cdot W_{a_1, a_b} \cdot W_{a_2, b_2} \cdots W_{a_k, b_k} \cdot F$, where for $1 \leq i \leq k$, $a_i \in \Sigma_1, b_i \in \Sigma_2$. Note that the above defines an ε -free transducer i.e., $W_{\varepsilon, \varepsilon}, W_{a, \varepsilon}$, and $W_{\varepsilon, b}$ are zero matrixes for all $a \in \Sigma_1, b \in \Sigma_2$. Given a multiresolution function $f: \Sigma_1^* \mapsto \mathbb{R}$ and a weighted relation $f_T: \Sigma_1^* \times \Sigma_2^* \mapsto \mathbb{R}$, a function $g: \Sigma_2^* \mapsto \mathbb{R}$ is defined by $g(t) = \sum_{s \in \Sigma_1^*} f(s) f_T(s, t)$ for all $t \in \Sigma_2^*$.

Following the above definitions we encode the image shown in Fig. 7(a) into a WFA shown in Fig. 8.

TABLE I

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)

name	notation	definition	meaning
shift	sh	$sh : Image \times \mathbb{Q}^2 \mapsto Image$	
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	$bn : Image \times \mathbb{Q} \mapsto Image$	
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	$size : Image \mapsto \mathbb{R}$	

In Fig. 8, the edge from states i to j is labeled by x : $(W_x)_{i,j}$. The pair I_i, F_i labeled in state i is the initial and the final distributions, respectively, for state i . For example, string $(0, 0)(0, 1)$ corresponds to the $1/4 \times 1/4$ pixel whose gray-level is $1/2 = 1 \times 1 \times 1/2$. In [3], a WFT computing the integral $\int_0^x \int_0^y f(x, y) dx dy$ is purposed. For bilevel images, $\int_0^1 \int_0^1 f(x, y) dx dy$ has the same meaning as “size.” In fact, the final distribution at the left-most state $1/8$ (corresponding to the gray level of the whole image) is exactly the “size” of the image. Here, we illustrate the computation of the “size” by $\int_0^1 \int_0^1 f(x, y) dx dy$. For simplicity, the resolution is set to 2^3 , i.e., the unit square is divided into $2^3 \times 2^3$ subsquares. From the WFA shown in Fig. 8, we have $f((0, 0)(0, 0)(i, j)) = 1$ for $i, j \in \{0, 1\}$; $f((0, 0)(i, j)(0, 0)) = 1$ for $i, j \in \{0, 1\}$ and $i \oplus j = 1$; $f((0, 0)(i, j)(k, l)) = 1/2$ for $i, j, k, l \in \{0, 1\}$, $i \oplus j = 1$, and $k \oplus l = 1$ where “ \oplus ” denotes “XOR.” By computing $\int_0^x \int_0^y f(x, y) dx dy$, the image size is

$$\begin{aligned}
& g((1, 1)(1, 1)(1, 1)) \\
&= \sum_{s \in \Sigma^3} f(s) \cdot f_T(s, (1, 1)(1, 1)(1, 1)) \\
&= \sum_{i, j \in \{0, 1\}} f((0, 0)(0, 0)(i, j)) \\
&\quad \cdot f_T((0, 0)(0, 0)(i, j), (1, 1)(1, 1)(1, 1)) \\
&+ \sum_{\substack{i, j \in \{0, 1\} \\ i \oplus j = 1}} f((0, 0)(i, j)(0, 0)) \\
&\quad \cdot f_T((0, 0)(i, j)(0, 0), (1, 1)(1, 1)(1, 1))
\end{aligned}$$

$$\begin{aligned}
& + \sum_{\substack{i, j, k, l \in \{0, 1\} \\ i \oplus j = 1, k \oplus l = 1}} f((0, 0)(i, j)(k, l)) \\
&\quad \cdot f_T((0, 0)(i, j)(k, l), (1, 1)(1, 1)(1, 1)) = \frac{1}{8}.
\end{aligned}$$

Intuitively, $g((1, 1)(1, 1)(1, 1))$ is the sum of the gray levels of ten pixels. Thus, the accuracy of the gray level of a pixel, computing from the original image, dominates the “size” computed by applying integral (represented as a WFT) on WFA. For example, if the final distribution of the left-most (middle) state is $1/6$ ($2/3$), then $g((1, 1)(1, 1)(1, 1)) = 1/6$.

IV. IMPLEMENTATIONS OF IMAGE OPERATIONS

In this section, we show how each of the image processing operations defined in Table I can be performed in the framework of Büchi automata.

Theorem 5: The set of operations listed in Table I can be implemented effectively using an automata approach. (Notice that the *size* operation applies to images representable by deterministic Büchi automata only.)

Proof: • *shift:* $image \times \mathbb{Q}^2 \rightarrow image$

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

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

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

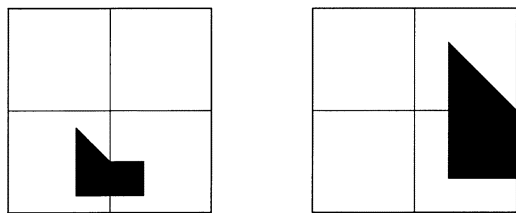


Fig. 9. Procedure of resize.

x and y axes). Suppose $s_0 \xrightarrow{r} s_r$, $r \in \{0, 1\}^2$ (i.e., s_r , $r \in \{0, 1\}^2$, represent the four immediate successors of s_0). It is easy 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 Fig. 9. Then, it is easy to see that $\text{resize}(B, 2) = B_{(0,0)}$. (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 $B_{1/2} = (\{0, 1\}^2, S \cup \{s'_0\}, \delta', s'_0, F)$ such that $\delta' = \delta \cup \{s'_0 \xrightarrow{(0,0)} s_0\}$. (That is, $B_{1/2}$ is obtained from B by adding a new initial state s'_0 together with transition $s'_0 \xrightarrow{(0,0)} s_0$.) $B_{1/2}$ clearly encodes $\text{resize}(B, 1/2)$.

- *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 $\text{flip}(M)$ along the x -axis. Flipping along the y -axis is done similarly.

- *rotate: image $\times \{90^\circ, 180^\circ, 270^\circ\} \rightarrow$ image*

It suffices to consider rotating the input image 90° in the clockwise fashion. Like the flipping case, 90° 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 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 $\times \mathbb{Q} \rightarrow$ 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 $b \in \mathbb{Q}$. Given an image B' , we extend B' by constructing a Büchi automaton B that recognizes $\text{ext}(B) = L(B') + L(F)$ (guaranteed by Corollary 2). See Fig. 10(top row). By repeatedly rotating the image and then performing the above shift operation, the boundary of B' can be computed as $\text{diff}(B, \text{un}(\text{un}(\text{un}(\text{ext}(B), \text{ext}(\text{flip}(\text{ro}(\text{ro}(B))))), \text{ext}(\text{flip}(B))), \text{ext}(\text{ro}(\text{ro}(B))))))$. See Fig. 10 for a series of such operations and how the boundary of an image (with the desired thickness) is extracted.

- *size: image $\rightarrow \mathbb{R}$*

The size of a deterministic Büchi recognizable image can easily be computed as the consequence of Theorem 4. ■

For related results concerning affine transformation, the reader is referred to [5] and [6]. In particular, it was shown in [6] that the set of images encodable as ω -automata is closed under rational affine transformations, they are also listed below for the sake of completeness.

Two given images F and F' are said to be *similar* within error bound ϵ if $\text{area}(\text{diff}(F, F')) \leq \epsilon$. 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 6: Given a finite automaton A , an ω -automaton B can be constructed such that $\text{Img}(A) \sim_0 \text{Img}(B)$, where $\text{Img}(A)$ and

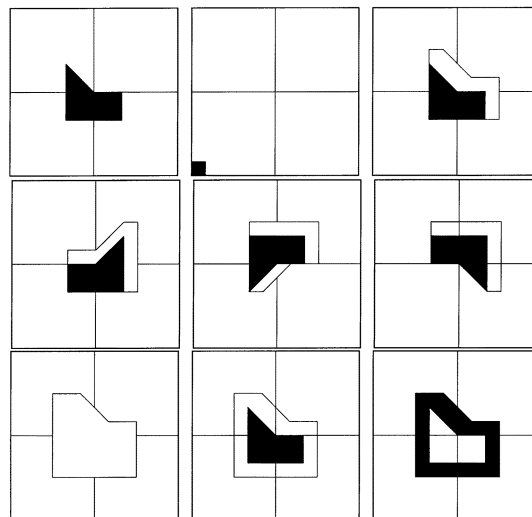


Fig. 10. Computing the boundary of an image.

$\text{Img}(B)$ 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 -automaton to the desired depth specified by the error bound, the following result is easy to obtain. Due to space limitations, the details are omitted.

Theorem 7: Given an ω -automaton A and an error bound ϵ , a finite automaton B can be constructed such that $\text{Img}(A) \sim_\epsilon \text{Img}(B)$, where $\text{Img}(A)$ and $\text{Img}(B)$ are the images represented by A and B , respectively.

V. CONCLUSION

We have proposed a novel approach, based on the theory of ω -automata, to represent bi-level images. The usefulness of this approach has further been demonstrated by showing a rich set of image operations to be performed easily in the framework of our ω -automata. In particular, we have shown that by applying the theory of Markov chains, the size of an image is easily computable. As far as computing image size is concerned, our approach provides an alternative to the “integral” operation reported in the literature for weighted finite automata. In our future research, it is of interest to see how color images can be dealt with in the framework of ω -automata.

ACKNOWLEDGMENT

The authors thank the anonymous referees for their comments which greatly improved the contents as well as the presentation of this paper. They also thank T. Izumi for discussions and comments.

REFERENCES

- [1] K. Culik and J. Kari, “Image compression using weighted finite automata,” *Comput. Graph.*, vol. 17, no. 3, pp. 305–313, 1993.
- [2] K. Culik and J. Karhumäki, “Finite automata computing real functions,” *SIAM J. Comput.*, vol. 23, no. 4, pp. 789–814, 1994.
- [3] K. Culik and J. Kari, “Finite-state transformations of images,” *Comput. Graph.*, vol. 34, pp. 151–166, 1997.
- [4] K. Culik and V. Valenta, “Finite automata based compression of bi-level and simple color images,” *Comput. Graph.*, vol. 21, pp. 61–68, 1997.
- [5] K. Culik and S. Dube, “Rational and affine expressions for image description,” *Discrete Appl. Mathematics*, vol. 41, pp. 85–120, 1993.

- [6] H. Jürgensen and L. Staiger, "Finite automata encoding geometric figures," in *Workshop on Implementing Automata WIA 99 Pre-Proceedings Potsdam 1999*, 1999, pp. XII-1–XII-8.
- [7] J. R. Büchi, "On a decision method in restricted second order arithmetic," in *Proceedings of the International Congress on Logic, Method, and Philosophy of Science*. Stanford, CA: Stanford Univ. Press, 1962, pp. 1–12.
- [8] W. Thomas, "Automata on infinite objects," in *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. Amsterdam, The Netherlands: Elsevier, 1990.
- [9] H. Samet, *Applications of Spatial Data Structures*. Reading, MA: Addison-Wesley, 1993.
- [10] B. Boigelot, S. Rassart, and P. Wolper, "On the expressiveness of real and integer arithmetic automata," in *Proceedings of the 25th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science*. New York: Springer-Verlag, 1998, vol. 1443, pp. 152–163.
- [11] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison-Wesley, 1979.
- [12] B. Nelson, *Stochastic Modeling: Analysis and Simulation*. New York: McGraw-Hill, 1995.
- [13] L. de Alfaro, "Formal verification of probabilistic systems," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, Dec. 1997.
- [14] H. Freeman, "Computer processing of line-drawing images," *ACM Comput. Surv.*, vol. 6, no. 1, pp. 57–97, 1974.

Optimal Generation Scheduling Based on AHP/ANP

James A. Momoh and Jizhong Zhu

Abstract—This paper proposes an application of the analytic hierarchy process (AHP) and analytic network process (ANP) for enhancing the selection of generating power units for appropriate price allocation in a competitive power environment. The scheme addresses adequate ranking, prioritizing, and scheduling of units before optimizing the pricing of generation units to meet a given demand. In the deregulated environment, the classical optimization techniques will be insufficient for the above-mentioned purpose. Hence, by incorporating the interaction of factors such as load demand, generating cost curve, bid/sale price, unit up/down cost, and the relative importance of different generation units, the scheme can be implemented to address the technical and nontechnical constraints in unit commitment problems. This information is easily augmented with the optimization scheme for an effective optimal decision. The scheme proposed is tested using the IEEE 39-bus test system.

Index Terms—Analytic hierarchy process, analytic network process, deregulated power industry, optimization, unit commitment.

I. INTRODUCTION

Since generators cannot instantly turn on and produce power, unit commitment must be planned in advance so that enough generation is always available to handle system demand with an adequate reserve margin in the event that generators or transmission lines go out or load demand increases. Unit commitment handles the unit generation schedule in a power system for minimizing the operating cost and satisfying the prevailing constraints such as load demand and system re-

serve requirements over a set of time periods [1]–[5]. Optimization techniques such as augmented Lagrangian relaxation, dynamic programming, and the branch-and-bound algorithm have been used to solve the classic unit commitment problem. In the deregulated power industry, the unit commitment problem is, in general, very complicated with a large number of possible decision variables such as load demand, generating cost curve, time-varying bid price of generation power, and the relative importance of different generation units. The classical optimization techniques are insufficient for handling these factors. Thus, in recent power system operation, it is necessary to modify the conventional unit commitment (UC) planning of thermal units with network constraints and the value of generator bidding as additional constraints. The classical UC problem is aimed at determining the startup and shutdown schedules of thermal units to meet forecasted demand over certain time periods (24 h to 1 week) and belongs to a class of combinatorial optimization problems. The methods [10], [11] that have been studied so far fall into roughly three types: heuristic search, mathematical programming MP, and hybrid methods.

Although these techniques are effective for problem posed, they do not handle network constraints and bidding issues. The paper proposes future UC requirements in a deregulated environment where network constraints, reliability, value of generation, and variational changes in demands and other costs may be factors.

The classical UC Lagrange method cannot solve the problem due to combinatorial explosion. Accordingly, as an initial approach to solve this complex problem, we attempt to find a method for solving UC considering network limitation and generation bids as a daily operational planning problem. This approach supports the decision making effectively of ranking units in terms of their values by using the analytic hierarchy process (AHP) and the analytic network process (ANP) techniques. The scheduled generation over time is studied as input into the optimal power flow (OPF) problem for optimal dispatch within the network and generation constraint.

This paper proposes application of the AHP [6], [8] and the ANP [7] for enhancing the selection of generating power units for appropriate price allocation in the deregulated power industry. It is different from the traditional unit commitment problem, which can be solved by directly using the optimization methods.

Explanation of Proposed Scheme

The basic concept of proposed optimal generation scheduling is as follows:

First, it is assumed that the ranking of generating units, and their priority as well as demand is known. As a result, the preferred generators for competitive scheduling and pricing will be known. Therefore, the number of generators whose fuel consumption constraints must be considered can be reduced considerably. This reduces the difficulties of unit commitment and optimal power flow. The proposed scheme addresses adequate ranking and prioritizing of units before optimizing the pricing of generation units to meet a given demand. By incorporating the interaction of factors, such as load demand, generating cost curve, bid/sale price, unit up/down cost, and the relative importance of different generation units, the scheme can be implemented to address the technical and nontechnical constraints in unit commitment problem. This information is easily augmented with the optimization scheme for effective optimal decisions. The scheme consists of the three following stages:

- 1) ranking of units in terms of their values by AHP/ANP;
- 2) checking the constraints by rule-based method;
- 3) solving optimization problem by interior point optimal power flow.

Manuscript received June 1, 1999; revised March 26, 2001. This work was supported by Electric Power Research Institute (EPRI), NSF RIMI, and CESaC. This paper was recommended by Associate Editor K. Pattipati.

J. A. Momoh is with the Department of Electrical Engineering, Center for Energy Systems and Control, Howard University, Washington, DC 20059 USA (e-mail: jmomoh@nsf.gov).

J. Zhu is with Alstom EAI, Bellevue, WA 98004 USA (e-mail: jizhong.zhu@alstom.esca.com).

Digital Object Identifier 10.1109/TSMCB.2003.811122