Incremental Evaluation of BDD-Represented Set Operators

HERALDO M. F. MADEIRA[1]*AND
JUNIOR BARRERA[2]

[1] Departamento de Informática - UFPR
PO Box 19081
81531-990 Curitiba - PR - Brazil
heraldo@inf.ufpr.br

[2] Instituto de Matemática e Estatística - USP
PO Box 66.281
05315-970 São Paulo - SP - Brazil
jb@ime.usp.br

Abstract. In Mathematical Morphology set operators are described by a formal language, whose vocabulary is composed of dilations, erosions, complementation, union and intersection. They are called morphological operators when expressed in this form. Translation invariant and locally defined set operators are called W-operators. Recently, decision diagrams have been used as an alternative representation for some 2-D and 3-D discrete W-operators. This paper shows that the reduced and ordered binary decision diagram (ROBDD) is a non-ambiguous scheme for representing W-operators and presents a method to compute the ROBDD of any W-operator from a corresponding morphological operator. This procedure of computing decision diagrams can be applied to the automatic proof of equivalence between morphological operators, since the W-operator they represent are equal if and only if they have the same ROBDD.

1 Introduction

Mathematical Morphology (MM) is a general framework for studying mappings over complete lattices [19] [12]. In particular, operators (or mappings) between binary images are of special interest in MM. A key aspect of MM is the description of such operators by means of a formal language, whose vocabulary is composed of the operations of intersection, union and complementation and of dilations and erosions. This language is called Morphological Language (ML) [4]. The sentences of ML are called *morphological operators*.

An important class of image operators is that of W-operators, *i.e.* the binary image operators that share the properties of translation invariance and local definition. W-operators are extensively used in morphological image processing, and this family of operators is the focus of this paper.

Many of the existing software for morphological image processing evaluate a set operator by directly parsing morphological phrases and successively applying the special-purpose algorithms for the elementary operations and operators. Besides the ML-based operator description, several structures for representing W-operators have been studied.

The interval-based representation used in [6] has been proved very useful in automatic learning of operators for image analysis [7]. The set of training input/output image pairs used in these systems can also be seen as a, rather incomplete, representation of the desired operator.

A graph-based representation for binary functions, called Binary Decision Diagram (BDD), was first used in MM in a special algorithm to compute the thinning operator [18]. This paper extends the use of BDD as a representation scheme for the whole class of W-operators.

The motivation of this work comes from the search for non-ambiguous (*i.e.* complete) and compact representations for a large class of operators. It is desired also that this representation lead to efficient algorithms for morphological image processing and that it satisfactorily solve the issue of verification whether two representations are equivalent.

In fact, the BDD-based representation is efficient:

*Acknowledges support from UFPR and CAPES.

the application time of a BDD-represented operator is, in the worst case, proportional to the size of the operator's window and to the size of the input set. The class of BDDs studied (reduced and ordered BDD) provides a trivial algorithm for determining the equivalence between morphological operators. The algorithms to convert a sentence of the ML to this new form of representation are presented in this work.

Following this Introduction, Section 2 recalls the basic concepts of binary MM. Section 3 presents the morphological language. Section 4 introduces the BDD as a representation scheme for Boolean functions. Section 5 presents the algorithms for basic manipulation of this graph-based representation. These algorithms are used in Section 6, where we state the uniqueness of this representation and show a method to derive the graph of any binary morphological operator. Examples of the graph construction process is given in Section 7, and implementation issues and results are found in 8. In the last section we give some conclusions and point directions for future works.

2 Windowed Set Operators

Let E be a nonempty set. $\mathcal{P}(E)$ denotes the power-set of E and $\subseteq$ is the usual set inclusion relation. The pair $(\mathcal{P}(E), \subseteq)$ is a *complete Boolean lattice*[8]. A set operator is any mapping defined from $\mathcal{P}(E)$ into itself. The set Ψ of all set operators $\psi : \mathcal{P}(E) \to \mathcal{P}(E)$ inherits the complete lattice structure of $(\mathcal{P}(E), \subseteq)$ by setting $\psi_1 \leq \psi_2 \Leftrightarrow \psi_1(X) \subseteq \psi_2(X), \forall \psi_1, \psi_2 \in \Psi, \forall X \in \mathcal{P}(E)$. Let $X, Y \in \mathcal{P}(E)$. $X \cup Y$, $X \cap Y$ and $X \setminus Y$ are the usual set operations of union, intersection and difference, and X^c is the usual set complementation.

Let $(E, +)$ be an *Abelian group* with zero element $o \in E$, called *origin*. Let $h \in E$ and $X, Y \subseteq E$. The set X_h defined by $X_h = \{x + h : x \in X\}$ is the *translation* of X by h. The set $X^t = \{-x : x \in X\}$ is the *transpose* of X. The set operations $X \oplus Y = \cup_{y \in Y}(X_y)$ and $X \ominus Y = \cap_{y \in Y}(X_{-y})$ are the *Minkowski addition* and *subtraction*.

A set operator $\psi \in \Psi$ is called *translation invariant (t.i.)* if and only if, $\forall h \in E, \forall X \in \mathcal{P}(E), \psi(X_h) = \psi(X)_h$.

Let W be a finite subset of E. A set operator ψ is called *locally defined (l.d.) within a window* W if and only if, $\forall h \in E, \forall X \in \mathcal{P}(E), h \in \psi(X) \Leftrightarrow h \in \psi(X \cap W_h)$.

Let Ψ_W denote the collection of all t.i. operators that are also l.d. within W. The elements of Ψ_W are called *W-operators*. The pair $(\Psi_W, \leq)$ constitutes a sub-lattice of the lattice $(\Psi, \leq)$. Furthermore, it is isomorphic to the complete Boolean lattice $(\mathcal{P}(\mathcal{P}(W)), \subseteq)$, since the mapping $\mathcal{K}_W : \Psi_W \to \mathcal{P}(\mathcal{P}(W))$ defined by $\mathcal{K}_W(\psi) = \{X \in \mathcal{P}(W) : o \in \psi(X)\}, \forall \psi \in \Psi_W$, is bijective and preserves the partial order. The set $\mathcal{K}_W(\psi)$ is the *kernel* of ψ.

3 Morphological Language

In this session we recall the main concepts of Mathematical Morphology from the viewpoint of a formal language.

Let $\psi_1, \psi_2 \in (\Psi, \leq)$. The *supremum* $\psi_1 \vee \psi_2$ and *infimum* $\psi_1 \wedge \psi_2$ operators verify $(\psi_1 \vee \psi_2)(X) = \psi_1(X) \cup \psi_2(X)$ and $(\psi_1 \wedge \psi_2)(X) = \psi_1(X) \cap \psi_2(X)$, $X \in \mathcal{P}(E)$. They can be generalized as $(\vee_{i \in I} \psi_i)(X) = \cup_{i \in I} \psi_i(X)$ and $(\wedge_{i \in I} \psi_i)(X) = \cap_{i \in I} \psi_i(X)$, where I is a set of indices. The *composition* operator $\psi_2 \psi_1$ is given by $\psi_2 \psi_1(X) = \psi_2(\psi_1(X))$, $X \in \mathcal{P}(E)$.

The set operators ι and ν defined by $\iota(X) = X$ and $\nu(X) = X^c$, $\forall X \in \mathcal{P}(E)$, are called, respectively, the *identity* and the *negation operators*. ι and ν are l.d. within the window $\{o\}$. The *dual* of an operator ψ, denoted by ψ^*, is defined by $\psi^*(X) = \psi(X^c)^c$, $X \in \mathcal{P}(E)$. Note that $\psi^* = \nu \psi \nu$.

For any $h \in E$, the set operator τ_h defined by $\tau_h(X) = X_h$, $\forall X \in \mathcal{P}(E)$, is called the *translation operator* by h. τ_h is l.d. in $\{-h\}$. For a t.i. operator ψ, $\tau_h \psi = \psi \tau_h$.

Let $B \in \mathcal{P}(W)$. The t.i. set operators δ_B and ε_B defined by $\delta_B(X) = X \oplus B$ and $\varepsilon_B(X) = X \ominus B$, $\forall X \in \mathcal{P}(E)$, are the *dilation* and *erosion* by the *structural element* B [6]. These set operators are l.d. in B^t and B, respectively. One can also write $\delta_B = \vee_{b \in B} \tau_b$ and $\varepsilon_B = \wedge_{b \in B} \tau_{-b}$.

Proposition 3.1 *If ψ, ψ_1 and ψ_2 are set operators locally defined within windows W, W_1 and W_2, respectively, then they have the following properties:*

1. *ψ is l.d. in any window $W' \supseteq W$;*
2. *$\psi_1 \wedge \psi_2$ and $\psi_1 \vee \psi_2$ are l.d in $W_1 \cup W_2$;*
3. *$\forall h \in E, \tau_h \psi$ is l.d. in W_{-h};*
4. *$\forall B \subseteq W, \delta_B \psi$ and $\varepsilon_B \psi$ are l.d. in $W \oplus B^t$ and $W \oplus B$, respectively;*
5. *$\psi_2 \psi_1$ is l.d. in $W_1 \oplus W_2$;*
6. *$\iota\psi$, $\psi\iota$, $\nu\psi$, $\psi\nu$ and ψ^* are l.d. in W.*

This proposition is demonstrated in [6].

Morphological operators, that is, sentences of the Morphological Language, are built as strings of elementary operators $(\varepsilon_{B_i}, \delta_{B_j}, \iota, \nu)$ bound by the operations $\vee$, $\wedge$ and composition.

As an example, let $A, B \in \mathcal{P}(W)$ such that $A \subseteq B$. The collection $[A, B] = \{X : A \subseteq X \subseteq B\}$ is

called an *interval*. The t.i. *sup-generating operator* $\lambda^W_{A,B}$, defined by $\lambda^W_{A,B}(X) = \{x \in E : (X_{-x}) \cap W \in [A,B]\}, X \in \mathcal{P}(E)$, can be described as $\lambda^W_{A,B} = \varepsilon_A \wedge \nu\delta_{\overline{B}^t}$, where $\overline{B} = W \backslash B$. Note that $\lambda^W_{A,B}$ is l.d. in $A \cup \overline{B}$, and, hence, in W.

The set $\mathcal{B}_W(\psi)$ of all maximal intervals in the kernel of a W-operator ψ is called *basis* of ψ [2]. See Section 2 for kernel definition.

Any W-operator ψ can be given by their canonical sup-decompositions:

$$\psi(X) = \cup\{\lambda^W_{A,B}(X) : [A,B] \subseteq \mathcal{K}_W(\psi)\} \quad [X \in \mathcal{P}(E)]$$

and

$$\psi(X) = \cup\{\lambda^W_{A,B}(X) : [A,B] \in \mathcal{B}_W(\psi)\} \quad [X \in \mathcal{P}(E)].$$

As a conclusion, ML is complete, in the sense that any W-operator can be represented by canonical forms, which are valid sentences of ML [3]. It is also expressive, since many useful operators can be described with short sentences.

4 Binary Decision Diagrams

A variety of representation methods for Boolean functions have been developed. In the classical ones, such as truth tables and canonical sum-of-products form, the representation of a function of n variables has size $O(2^n)$. Other approaches, such as the set of prime implicants (or equivalently the set of maximal intervals) or a subset of irredundant ones, yield to compact representation for many functions, but simple operations such as complementation may result in a representation of intractable size. Here we describe a graph-based representation method that is very useful for a large class of Boolean functions, many of them not tractable by other schemes.

4.1 Boolean Functions as DAGs

The representation of a Boolean function by a decision-based structure was introduced by Lee[15] and further popularized by Akers [1] under the name of binary decision diagram (BDD). Algorithms for BDD manipulation are described in [10]. Efficient implementation directions are found in [13] and [9]. Applications of BDDs in Digital Image Processing have been recently developed [20], [18], [17].

A Binary Decision Diagram of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a rooted, directed acyclic graph (DAG) with two types of nodes: terminal and nonterminal. Each terminal node v has as attribute a value $value(v) \in \{0,1\}$. The nonterminal nodes v have two children nodes, $low(v)$ and $high(v)$. Each nonterminal v is labeled with an input variable index $index(v) \in \{1,2,\ldots,n\}$.

For a given assignment to the input variable vector $\mathbf{x} = (x_1,\ldots,x_n)$, the value of the function is determined by traversing the graph from the root to a terminal node: at each nonterminal node v with $index(v) = i$, if $x_i = 0$, then the arc to $low(v)$ is followed. Otherwise ($x_i = 1$), the arc to $high(v)$ is followed. The value of the function is given by the value of the terminal node.

A node v in a BDD represents a Boolean function f_v such that: (a) if v is a terminal node with $value(v) = 0$, then $f_v = 0$; (b) if v is a terminal node with $value(v) = 1$, then $f_v = 1$; (c) if v is a nonterminal node and $index(v) = i$, then $f_v = \overline{x_i} \cdot f_{low(v)} + x_i \cdot f_{high(v)}$. The mathematical background of the BDD construction is the well known *Shannon Expansion* of a Boolean function: $f = x_i \cdot f|_{x_i} + \overline{x_i} \cdot f|_{\overline{x_i}}$. The restrictions $f|_{x_i} = f(x_1,\ldots,x_{i-1}, 1, x_{i+1},\ldots,x_n)$ and $f|_{\overline{x_i}} = f(x_1,\ldots,x_{i-1}, 0, x_{i+1},\ldots,x_n)$ are the *cofactors* of f with respect to the literals x_i and $\overline{x_i}$.

4.2 Reduced and Ordered BDDs

An *ordered BDD* (OBDD) is a BDD such that any path in the graph from the root to a terminal node visits the variables in ascending order of their indices, *i.e.*, $index(v) < index(low(v))$ whenever v and $low(v)$ are nonterminal, and $index(v) < index(high(v))$ whenever v and $high(v)$ are nonterminal. Since a variable appears at most once in each path, a function is evaluated in time $O(n)$ in an OBDD.

If an OBDD contains no pair of nodes $\{u,v\}$ such that the graph rooted by u and v are isomorphic, and if it contains no node v such that $low(v) = high(v)$, it is called a *reduced OBDD* (ROBDD). An OBDD of N vertices can be transformed in an equivalent ROBDD in time $O(N \cdot \log N)$ by the REDUCE algorithm presented in [10].

The following theorem states the canonicity of the ROBDD representation.

Theorem 4.1 *For any Boolean function f, there is a unique ROBDD representing f. Furthermore, any other OBDD representing f contains more vertices.*

The proof of this theorem is presented in [10].

Besides allowing function evaluation in linear time, ROBDDs can be used in the efficient computing of function satisfiability, tautology and equivalence.

4.3 The Lattice of ROBDDs

Let $\mathcal{F}_n$ denote the set of all functions $\{0,1\}^n \to \{0,1\}$ and let Φ_W denote the set of all functions $\mathcal{P}(W) \to$

$\{0,1\}$, such that $W = \{w_1, \ldots, w_n\}$. We establish a one-to-one correspondence between elements of $\mathcal{F}_n$ and Φ_W by making $f(x_1, \ldots, x_n) = \varphi(\{w_i \in W : x_i = 1\})$. Let $\leq$ denote the usual order in $\{0,1\}$. The partially ordered sets $(\mathcal{F}_n, \leq)$ and $(\Phi_W, \leq)$ are complete Boolean lattices isomorphic to $(\mathcal{P}(\mathcal{P}(W)), \subseteq)$. This is observed from the bijective mapping $K : \Phi_W \rightarrow \mathcal{P}(\mathcal{P}(W))$ given by $K(\varphi) = \{X : \varphi(X) = 1\}$.

Let $\mathbf{G}_W$ denote the set of all ROBDDs representing functions in $\mathcal{F}_n$. From the previous theorem, ROBDDs are unique and non-ambiguous representations of Boolean functions. Thus, there is a bijective mapping between $\mathbf{G}_W$ and $\mathcal{F}_n$, and, hence, between $\mathbf{G}_W$ and Φ_W. The ROBDD of a binary function $\varphi \in \Phi_W$ is denoted by $G(\varphi)$ and is simply called "graph" in the sequence. The pair $(\mathbf{G}_W, \sqsubseteq)$ is a complete Boolean lattice isomorphic to the lattice $(\Phi_W, \leq)$, where the partial order $\sqsubseteq$ between two graphs $G(\varphi_1)$ and $G(\varphi_2)$ in $\mathbf{G}_W$ is defined by $G(\varphi_1) \sqsubseteq G(\varphi_2) \Leftrightarrow \varphi_1 \leq \varphi_2$, $\varphi_1, \varphi_2 \in \Phi_W$.

5 Operations on BDDs

5.1 Algorithms for Logical Operations between Graphs

Let $\varphi, \varphi_1, \varphi_2 \in \Phi_W$ and let $G, G_1, G_2 \in \mathbf{G}_W$ be their corresponding graphs. The graph complement, infimum and supremum, respectively denoted by $\bar{\cdot}$, $\sqcap$ and $\sqcup$, are computed by the algorithm ApplyOperation presented in [10]. This algorithm takes as input two graphs with N_1 and N_2 nodes and a logical operation (AND, OR, XOR) as parameters, and return the resulting graph in time $O(N_1 \cdot N_2)$. The algorithm is based in the following property of Boolean functions: $f_1 \otimes f_2 = \overline{x_i} \cdot (f_1|_{\overline{x_i}} \otimes f_2|_{\overline{x_i}}) + x_i \cdot (f_1|_{x_i} \otimes f_2|_{x_i})$, where $\otimes$ denotes one of the sixteen logical function of two variables [11]. Thus,

$$\begin{array}{lll} G_1 \sqcap G_2 = G(\varphi_1 \cdot \varphi_2) & = \text{ApplyOp.}(G_1, G_2, AND); \\ G_1 \sqcup G_2 = G(\varphi_1 + \varphi_2) & = \text{ApplyOp.}(G_1, G_2, OR); \\ \overline{G} \qquad = G(\overline{\varphi}) & = \text{ApplyOp.}(G, 1, XOR). \end{array}$$

The graph of the complement of $\varphi \in \Phi_W$ can be alternatively computed by simply exchanging the values of the two terminal nodes.

The *graph dual* of G in $\mathbf{G}_W$ is denoted G^*, and is defined by $G^*(\varphi(X)) = G(\varphi^*(X)) = G(\overline{\varphi}(X^c))$. It can be computed from G by swapping $low(v)$ with $high(v)$ in each vertex of G and swapping the terminal nodes too.

5.2 Algorithm for the Translation of a Graph

Let $h \in E$ and $\varphi \in \Phi_W$. The *translation* of φ by h, denoted by φ_h is defined by $\varphi_h(Y) = \varphi(Y_h), \forall Y \in \mathcal{P}(W_{-h})$. Thus, $\varphi_h \in \Phi_{W_{-h}}$ and the Boolean expressions of φ_h can be obtained from those of φ by a changing of variables. The *input translation* by h of the graph $G(\varphi) \in \mathbf{G}_W$ is denoted by $G(\varphi) + h$ and defined as $G(\varphi) + h = G(\varphi_h)$. Note that $G(\varphi) + h \in \mathbf{G}_{W_{-h}}$.

For a trivial implementation of $G + h$, $W \cup W_{-h}$ must be consistently ordered, otherwise a reordering of the BDD may be mandatory. From this point, we assume that $E = Z^2$ and that a all translations of interest of W fit in a rectangle $R \subseteq E$. The usual lexicographical order for the elements of R yields to a preservation of the order of the elements of W in any translation. Thus, the translated graph $G + h$ is isomorphic to G up to vertex relabeling.

6 Translating Morphological Operators into Decision Diagrams

In this section we see how to incrementally compute the ROBDD of a W-operator described by a morphological expression.

We have seen (Section 2) that $(\Psi_W, \leq)$ is isomorphic to $(\mathcal{P}(\mathcal{P}(W)), \subseteq)$, and also (Section 4) that $(\Phi_W, \leq)$ is isomorphic to $(\mathbf{G}_W, \sqsubseteq)$ and to $(\mathcal{P}(\mathcal{P}(W)), \subseteq)$. This demonstrates, by transitivity of isomorphisms, that there is a one-to-one correspondence between elements of $\mathbf{G}_W$ and Ψ_W. This shows that a graph non-ambiguously and uniquely represents a W-operator.

We denote by $\mathcal{G}_W(\psi)$ the corresponding graph of a W-operator ψ. The caligraphic $\mathcal{G}()$ is to stress that the argument is an operator, while $G()$ has as argument a function. The subscript W is used in both notations to emphasize the window in which the graph is defined.

The mapping $\varphi : \Psi_W \rightarrow \Phi_W$ given by $\varphi(\psi)(X) = 1 \Leftrightarrow o \in \psi(X), \forall \psi \in \Psi_W$ establishes the *characteristic function* of the W-operator ψ. Note that the kernel $\mathcal{K}(\psi)$ is the *satisfying set* of $\varphi(\psi)$, and the basis $\mathcal{B}(\psi)$ is the set of *prime implicants* of $\varphi(\psi)$.

The following propositions establish the algorithms that perform the basic operations on graphs of morphological operators.

Proposition 6.1 *If ψ, ψ_1 and ψ_2 are W-operators, respectively within the windows W, W_1 and W_2 and with graphs $\mathcal{G}_W(\psi)$, $\mathcal{G}_{W_1}(\psi_1)$ and $\mathcal{G}_{W_2}(\psi_2)$, then*

$$\begin{array}{rcl} \mathcal{G}_W(\nu\psi) & = & \overline{\mathcal{G}}_W(\psi) \\ \mathcal{G}_{W_1 \cup W_2}(\psi_1 \wedge \psi_2) & = & \mathcal{G}_{W_1 \cup W_2}(\psi_1) \sqcap \mathcal{G}_{W_1 \cup W_2}(\psi_2) \\ \mathcal{G}_{W_1 \cup W_2}(\psi_1 \vee \psi_2) & = & \mathcal{G}_{W_1 \cup W_2}(\psi_1) \sqcup \mathcal{G}_{W_1 \cup W_2}(\psi_2) \\ \mathcal{G}_W(\psi^*) & = & \mathcal{G}^*_W(\psi). \end{array}$$

Proposition 6.2 *If ψ is a W-operator with graph $\mathcal{G}_W(\psi)$ and, $\forall h \in E$, τ_h is the translation operator*

by the vector h, then

$$\mathcal{G}_{W_{-h}}(\tau_h\psi) = \mathcal{G}_W(\psi) + h.$$

Proposition 6.3 *If ψ is a W-operator with graph $\mathcal{G}_W(\psi)$ and δ_B is the dilation by B, then*

$$\mathcal{G}_{W\oplus B'}(\delta_B\psi) = \sqcup_{b\in B}(\mathcal{G}_{W\oplus B'}(\psi) + b).$$

Proposition 6.4 *If ψ is a W-operator with graph $\mathcal{G}_W(\psi)$ and ε_B is the erosion by B, then*

$$\mathcal{G}_{W\oplus B}(\varepsilon_B\psi) = \sqcap_{b\in B}(\mathcal{G}_{W\oplus B}(\psi) + (-b)).$$

Proposition 6.5 *If ψ is a W-operator with graph $\mathcal{G}_W(\psi)$, $\lambda^{W'}_{A,B}$ is the sup-generating operator, and $\overline{B} = W'\backslash B$, then*

$$\mathcal{G}_{W\oplus W'}(\lambda^{W'}_{A,B}\psi) = \mathcal{G}_{W\oplus W'}(\varepsilon_A\psi) \sqcap \overline{\mathcal{G}}_{W\oplus W'}(\delta_{\overline{B}'}\psi).$$

Definition 6.1 *Given a set of indices I, we say that a collection of intervals $\{[A_i, B_i] : i \in I\}$* exactly covers *a W-operator ψ if $\psi = \vee_{i\in I}\lambda^W_{A_i,B_i}$.*

For example, the basis of a W-operator is an exact cover of it. And so is the set $\{[A, A] : A \in \mathcal{K}_W(\psi)\}$. The set of intervals representing the positive paths (those that yield to 1) of a graph $\mathcal{G}_W(\psi)$ is a *disjoint* exact cover of ψ, because it exactly covers ψ and any two of its intervals have empty intersection.

Proposition 6.6 *If ψ_1 and ψ_2 are t.i. operators l.d. in W_1 and W_2, respectively, with graphs $\mathcal{G}_{W_1}(\psi_1)$ and $\mathcal{G}_{W_2}(\psi_2)$, and $\{[A_i, B_i] : i \in I\}$ is a set of intervals that exactly covers ψ_2, then*

$$\mathcal{G}_{W_1\oplus W_2}(\psi_2\psi_1) = \sqcup_{i\in I}\mathcal{G}_{W_1\oplus W_2}(\lambda^{W_2}_{A_i,B_i}\psi_1).$$

The proofs of these propositions are based on the isomorphisms mentioned in the beginning of this section, and are presented in [16].

Figure 1 illustrates the building blocks used to pictorially describe a morphological operator (a) and

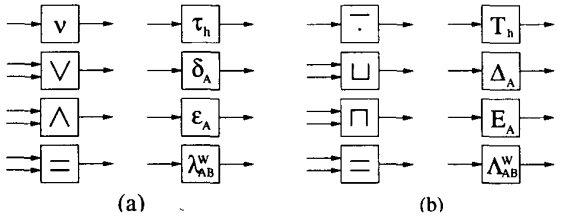

Figure 1: Basic operations on operators and corresponding operations on graphs.

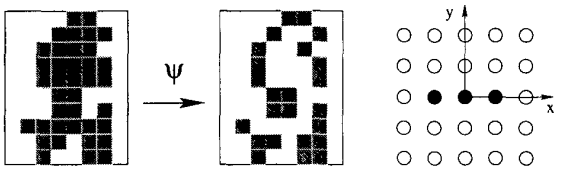

Figure 2: Vertical border detector and its window.

$X \subseteq W$	$\varphi(\psi)(X)$	$X \subseteq W$	$\varphi(\psi)(X)$
$\emptyset$	0	$\{w_1\}$	0
$\{w_3\}$	0	$\{w_1, w_3\}$	0
$\{w_2\}$	1	$\{w_1, w_2\}$	1
$\{w_2, w_3\}$	1	$\{w_1, w_2, w_3\}$	0

Table 1: Characteristic function of a W-operator.

their corresponding blocks used to incrementally build its graph (b). The correspondence between sets (a) and (b) is explained in the propositions above. The blocks with the symbol "=" compares two input morphological operators (and graphs), as explained in the sequence, and result a boolean value. In the examples below we will use such graphical construction.

6.1 Automatic Proof of Equivalence

An important task in MM is to determine whether two descriptions correspond to the same operator. Proving the equivalence of two morphological operators ψ_1 and ψ_2 involves manipulation of their expressions using well known set-theoretic properties, and it is often a difficult task. On the other hand, if we know the graphs G_1 and G_2 of two W-operators, then the proof of their equivalence is trivial: since the graph representation of a W-operator is unique, we simply compare if both graph descriptions are equal. Alternatively, we could verify it by calling ApplyOperation (G_1, G_2, XOR), and test if the resulting graph is the trivial leaf "0".

7 Examples

In this section we will see how to compute the ROBDD representation of a W-operator. Several ways of representing a W-operator exist. For example, an operator ψ locally defined in $W = \{w_1, w_2, w_3\} = \{(-1, 0), (0, 0), (1, 0)\}$ that detects the vertical borders of Figure 2, can be represented by:

- the characteristic function $\varphi(\psi)$ of Table 1;
- the Hasse diagram of Figure 3(a);
- the canonical sup-decomposition $\psi = \lambda^W_{\{w_2\},\{w_1,w_2\}} \vee \lambda^W_{\{w_2\},\{w_2,w_3\}}$;

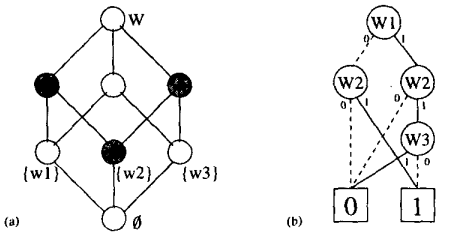

Figure 3: (a) Hasse diagram and (b) graph of the operator.

- the morphological expression by means of erosion and dilations $\psi = \varepsilon_{\{w_2\}} \wedge \nu\delta_{\{w_1\}} \vee \varepsilon_{\{w_2\}} \wedge \nu\delta_{\{w_3\}}$
- the block diagram of Figure 4(a);
- the Boolean expressions $F(\psi)(x_1, x_2, x_3) = x_2 \cdot \overline{x_1 x_3} = \overline{x}_1 x_2 + x_2 \overline{x}_3 = x_2(\overline{x}_1 + \overline{x}_3)$;
- the kernel $\mathcal{K}_W(\psi) = \{\{w_2\}, \{w_2, w_3\}, \{w_1, w_2\}\}$;
- the basis $\mathcal{B}_W(\psi) = \{[\{w_2\}, \{w_1, w_2\}], [\{w_2\}, \{w_2, w_3\}]\}$;
- the graph (ROBDD) $\mathcal{G}_W(\psi)$ of Figure 3(b).

It is possible to compute the graph of any W-operator, whenever its description in the morphological language is known. For the incremental computation of the graph of a W-operator ψ described by a morphological expression, we start with the graph of the identity operator and successively apply the algorithms that correspond to the propositions in the preceding section, according to the parsing of the sentence that describes ψ. Each step is initiated by the modification of the window by applying Proposition 3.1.

7.1 Anti-Extensive Vertical Border Detector

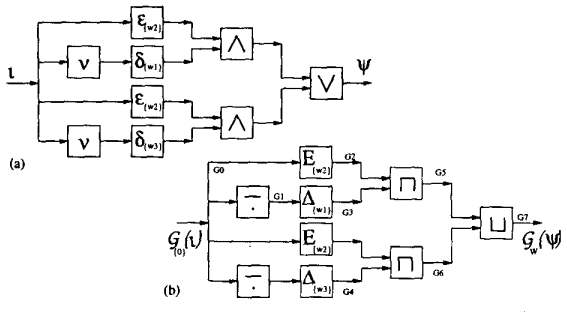

Figure 4: Construction of the morphological expression (a) and of the graph (b) of the vertical border detector.

In order to obtain the graph of the anti-extensive vertical border detector just seen, we set the configuration of Figure 4(b) and apply the corresponding algorithms. At each step we obtain the graphs shown in Figure 5.

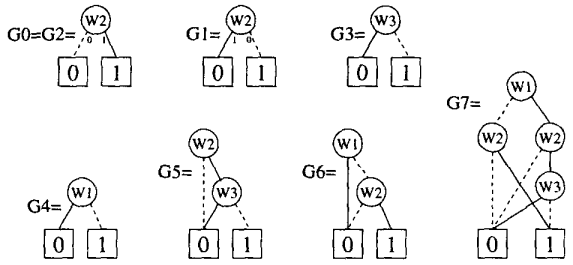

Figure 5: Incrementally computed graph of the border detector (see labels in Figure 4-b).

7.2 Composition of two Median Filters

This example illustrates the composition operation between two operator graphs. When two graphs are known, but we have no access to the morphological expression that generated them, then we can use the algorithm of Prop. 6.6 to calculate the graph of the composition of the two represented operators.

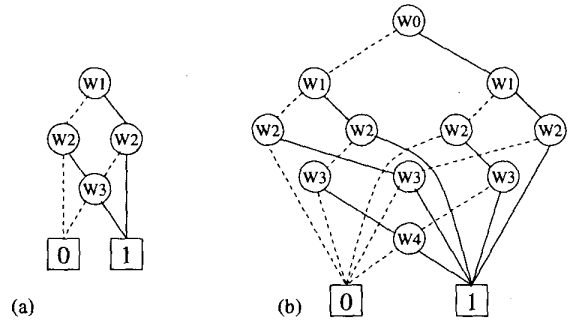

Figure 6: (a) 1x3 median filter and (b) composition of two of them.

Figure 6 shows the graph of the median filter of window $W = \{w_1, w_2, w_3\} = \{(-1,0),(0,0),(1,0)\}$, and the resulting composition between two of them. The latter has window $W = \{w_0, w_1, w_2, w_3, w_4\} = \{(-2,0), (-1,0), (0,0), (1,0), (2,0)\}$.

8 Implementation

We implemented the algorithms of section 6 to compute the BDD of image operators. The low-level BDD algorithms followed the ideas found in [9], *i.e.* the ITE

operation, instead of the already mentioned APPLY-OPERATION() of [10]. In this level, the graphs of all operators being processed share their nodes in a unique table (global storage). However, at a higher level (local storage), each operator has its own ROBDD, together with its window stored in the usual 2-D scan order.

The implementation of the translation operation is not straightforward in the ITE paradigm: a relabeling of nodes can't be done, because the label is part of the hash key (see [9]). In order to implement the translation operator still by means of an order preserving change of variables, the nodes in the unique table don't store the actual indices of the operator's function variables. Instead, they store a reference into an unordered table of all window elements in use. When extracting a graph from the unique table, or importing a new graph into it, the appropriate conversion between local and global indices is made. Thus, the translation of a graph is done by extracting it from the unique table, and then loading it again with the window appropriately displaced.

Compositions by erosion and dilations are computed by calling the translation several times. The parameters for the sup-generating operators used in the computation of the composition of two operators is directly obtained from each positive path of one of the input graphs. Sometimes the task of representation conversion from morphological expressions is slow, because the number of generated graph nodes may grow exponentially, depending on the operator.

The application of a BDD over an image could be done by a general program that traversed the graph structure, but in our system we implemented a code generator: each BDD is converted to a program in the C language, which takes the image and the window as inputs. The program is then compiled, optimized and stored in a dynamic library. The advantage of this approach is obvious: each node is an if-then-else structure which is directly executed, leading to faster programs. The implemented image structure uses one pixel per byte.

8.1 Experimental Results

The programs for incremental construction of BDDs and the one that applies them over images were developed on the Khoros 2.2 environment [14]. For comparison reasons we used the MMach Toolbox [5]. Several runs were made for the created BDD-based operators, and compared against the equivalent operators built from the specifically designed ones of the mentioned Toolbox.

Timing results were collected on a 450 MHz personal computer running the Linux operating system,

Image Name	Size (pixels)	Thin_4_homoth		Dil_diamond	
		MMach	BDD	MMach	BDD
Beetle	4096	63.2s	19.3s	33.4s	4.8s
NoisyBall	4096	62.1s	15.4s	35.4s	19.4s
UltraSnd	4600	69.6s	23.3s	39.3s	3.7s

Table 2: Comparison of running times for the operators.

Figure 7: Diamond-shaped structuring element.

and shown in Table 2. Two previously calculated operators, named "Thin_4_homoth" and "Dil_diamond", were applied over three images of around 2000 × 2000 pixels. They are explained in the sequence.

The first operator tested was the four-homothetic thinning as defined in [17]. Each complete turn is an operator that depends on a window of 82 elements. The task in the MMach test consisted on calling the thinning operator a single time, with parameter of 80 steps (each step rotates the structuring element by 45^o), while the equivalent task in the BDD test (our implementation) consisted in 10 consecutive applications of the BDD of a single turn. This BDD has 2899 nodes. The measured times show that the BDD implementation is at least tree times faster.

The second operator consists of a dilation by the diamond of Figure 7. The operator window size has 84 elements. The task in both runs consisted of 10 consecutive applications of the operator. As would be expected, the BDD program is too sensible to the input image contents: when it has large areas of zero value, short paths in the BDD are rare. Nevertheless, the implementation with BDD is much faster than in the other approach.

9 Conclusion

In this work we saw that the Reduced Ordered Binary Decision Diagram is a good alternative for the representation of translation-invariant and locally defined set operators. Its efficient application time makes it a good candidate as the core representation scheme of non-linear signal and image processors.

The main contribution of this work was the development of a well defined procedure to convert any expression of the morphological language for a given W-operator in its correspondent ROBDD. We saw that the ROBDD is a non-ambiguous (complete) and unique

representation scheme for W-operators, The uniqueness of the ROBDD representation allow a simple solution to the problem of checking the equivalence between morphological operators.

The main drawback of the graph-based representation is that its size (number of nodes) can grow exponentially with the number of variables (window size) in some cases. A challenging task is to classify the Boolean functions in terms of such behavior, in order to determine the conditions for which one description (morphological expression, basis, graph, etc.) is better than the other.

Currently we are working on an implementation of a BDD-based morphological machine and extending this approach to multi-level morphology.

References

[1] S. B. Akers. Binary Decision Diagrams. *IEEE Transactions on Computers*, C-27(6):509–516, June 1978.

[2] G. J. F. Banon and J. Barrera. Minimal Representations for Translation-Invariant Set Mappings by Mathematical Morphology. *SIAM J. Appl. Math.*, 51(6):1782–1798, December 1991.

[3] G. J. F. Banon and J. Barrera. Bases da Morfologia Matemática para Análise de Imagens Binárias. IX Escola de Computação, Pernambuco, July 1994.

[4] J. Barrera and G. J. F. Banon. Expressiveness of the Morphological Language. In *Image Algebra and Morphological Image Processing III*, volume 1769 of *Proc. of SPIE*, pages 264–274, San Diego, California, 1992.

[5] J. Barrera, G. J. F. Banon, R. A. Lotufo, and R. Hirata Jr. MMach: a Mathematical Morphology Toolbox for the Khoros System. *Electronic Imaging*, 7(1):174–210, 1998.

[6] J. Barrera and G. P. Salas. Set Operations on Closed Intervals and Their Applications to the Automatic Programming of Morphological Machines. *Electronic Imaging*, 5(3):335–352, July 1996.

[7] J. Barrera, R. Terada, R. Hirata Jr, and N. S. T. Hirata. Automatic Programming of Morphological Machines by PAC Learning. *Fundamenta Informaticae*, 41(1-2):229–258, January 2000.

[8] G. Birkhoff. *Lattice Theory*. American Mathematical Society, Providence, Rhode Island, 3rd edition, 1967.

[9] K. S. Brace, R. L. Rudell, and R. E. Bryant. Efficient Implementation of a BDD Package. In *Proceedings of 27th ACM/IEEE Design Automation conference*, pages 40–45, 1990.

[10] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[11] G. de Michelli. *Synthesis and Optimization of Digital Circuits*. Mc Graw-Hill, 1994.

[12] H. J. A. M. Heijmans. *Morphological Image Operators*. Academic Press, Boston, 1994.

[13] K. Karplus. Representing Boolean Functions with If-Then-Else DAGs. Technical Report UCSC-CRL-88-28, UCSC, November 1988.

[14] K. Konstantinides and J. Rasure. The KHOROS Software Development Environment for Image and Signal Processing. *IEEE Transactions on Image Processing*, 3(3):243–252, 1994.

[15] C. Y. Lee. Representation of Switching Circuits by Binary-Decision Programs. *Bell Syst. Tech. Journal*, 38:985–999, 1959.

[16] H. M. F. Madeira and J. Barrera. Decision Diagrams as the Core of Morphological Machines. *Manuscript in preparation*, 2000.

[17] H. M. F. Madeira, J. Barrera, R. Hirata Jr, and N. S. T. Hirata. A New Paradigm for the Architecture of Morphological Machines : Binary Decision Diagrams. In J. Stolfi and C. L. Tozzi, editors, *Proc. SIBGRAPI'99 - XII Brazilian Symposium in Computer Graphics and Image Processing*, pages 283–292, Campinas, SP, Brazil, November 1999.

[18] L. Robert and G. Malandain. Fast Binary Image Processing Using Binary Decision Diagrams. *Computer Vision and Image Understanding*, 72(1):1–9, October 1998.

[19] J. Serra. *Image Analysis and Mathematical Morphology. Volume 2: Theoretical Advances*. Academic Press, 1988.

[20] M. Starkey and R. Bryant. Using Ordered Binary-Decision-Diagrams for Compressing Images and Image Sequences. Technical Report CMU-CS-95-105, School of Computer Science, Carnegie Mellon Univeristy, January 1995.