Switching Algebras (the algebra of Boolean formulas)

Madhav P. Desai

February 10, 2023

1 Representing a finite Boolean algebra

Recall that we have shown that any Boolean algebra is iso-morphic to a set algebra on the universe set of atoms of the Boolean algebra. That is, if B is a boolean algebra, and if A is the set of atoms in the Boolean algebra, then

- Every element x in the Boolean algebra corresponds to the subset A_x of A consisting of those atoms that are $\leq x$.
- If x and y are two elements of the Boolean algebra, then $x + y$ corresponds to $A_x \cup A_y$, $x.y$ corresponds to $A_x \cap A_y$, and $\overline{x}$ corresponds to the complement of A_x in A (that is $A - A_x$).

If the atoms in A are ordered as $a_0, a_1, \ldots a_{n-1}$, then each subset Q of A can be represented by an ordered n-tuple of bits (0/1)

$$(b_0, b_1, b_2, \ldots b_{n-1})$$

with $b_i = 1$ if atom a_i is in Q and $b_i = 0$ otherwise.

Recall the Boolean algebra

$$\mathbf{B}_2 \ = \ \{\{0,1\}, ., +\}$$

with two elements 0 and 1. Then, based on the discussion above, we see that any *finite* Boolean algebra can be represented in the form $\mathbf{B}_2^n$, for some integer $n > 0$. The elements of $\mathbf{B}_2^n$ can be listed as $n-$tuples $(b_0, b_1, b_2, \ldots b_{n-1})$ where each $b_i \in \{0, 1\}$.

To summarize: henceforth, any finite Boolean algebra is equivalent to $\mathbf{B}_2^n$ for some $n \geq 0$.

2 Boolean functions and formulas

Let $n > 0$, and consider a Boolean function

$$f : \mathbf{B}_2^n \rightarrow \mathbf{B}_2$$

Corresponding to this Boolean function, we can define an ON-set of the function as

$$f_{ON} = \{(b_0, b_1, \ldots b_{n-1}) \in \mathbf{B}_2^n : f(b_0, b_1, \ldots b_{n-1}) = 1\}$$

The OFF-set of f, f_{OFF} is the set complement of f. Note that a Boolean function uniquely corresponds to its ON-set, and thus, the set of Boolean functions from $\mathbf{B}_2^n \rightarrow \mathbf{B}_2$ forms a Boolean algebra. In this algebra of Boolean functions, the atoms are those functions whose ON-sets are singletons. Thus, there are 2^n atoms, and 2^{2^n} Boolean functions from $\mathbf{B}_2^n$ to $\mathbf{B}_2$.

We need a convenient way in which Boolean functions can be represented and manipulated. Formulas provide us such a mechanism. The set of formulas on n variables $x_0, x_1, \ldots x_{n-1}$ is the set of strings which can be constructed by applying the following rules.

1. 0, 1 are formulas.
2. The letters x_0, x_1, …, x_{n-1} are formulas.
3. If A is a formula, then $(\neg A)$ (or $\overline{A}$) is a formula.
4. If A, B are two formulas, then $(A + B)$ and $(A.B)$ are formulas.

To summarize: a formula is a string, and there are infinitely many formulas.

Now suppose we have a formula, lets say $f = (x_0 + (x_1.x_2))$. Can this represent a Boolean function from $\mathbf{B}_2^3 \rightarrow \mathbf{B}_2$? The obvious way is to evaluate f by substitution. The value of the Boolean function at $(b_0, b_1, b_2) \in \mathbf{B}_2^3$ is obtained by substituting b_0 for x_0 etc. in the formula and then defining $f(b_0, b_1, b_2)$ to be the element $(b_0 + (b_1.b_2))$ in $\mathbf{B}_2$.

Thus, a formula defines a Boolean function. The converse is also true. That is, every Boolean function can be represented by a formula. To see this, observe that the set of Boolean functions itself is a Boolean algebra, and thus each Boolean function can be written as a sum of atoms. Each

Boolean function which is an atom is necessarily a singleton, and evaluates to 1 at exactly one point. The formula for an atom is thus a product term (min-term). The formula for a sum of atoms is thus a sum or min-terms. Thus, every Boolean function can be expressed by a sum of products formula.

3 Switching algebra of Boolean formulas

A switching algebra consists of Boolean formulas, together with the $+$, . and operations. If f and g are two formulas, then $f = g$ if and only if f and g represent the same Boolean function.

Formulas can be combined and manipulated:

1. $(x_i + x_i) = x_i$, $x_i.x_i = x_i$.
2. $(x_i.0) = 0$.
3. $(x_i + (x_j + x_k)) = ((x_i + x_j) + x_k)$.
4. etc.

Formulas can be mapped to logic networks in a trivial manner.

4 The verification problem for Boolean formulas

Given two formulas f and g:

- Is $f == g$? If no, find a point b on which $f \neq g$.

5 Normal forms

A normal form is a formula which is uniquely defined. In general, there are several possible formulas for a Boolean function. A normal form is a standard (and unique) formula for a Boolean function. The common normal forms used are

- Disjunctive normal form (DNF): the function is represented as a sum of min-term products.

- Conjunctive normal form (CNF): the function is represented as a product of max-term sums.
- Algebraic normal form (ANF): the function is represented as a minimal exclusive OR of product terms.
- Reduced-ordered Binary decision diagrams (ROBDD): the function is represented as a decision tree (see Section 6).

For example, consider the function on two variables represented by $x_1 \oplus x_2$. The normal forms for this function are

- DNF: $x_1.(\neg x_2) + (\neg x1).x2$
- CNF: $(x_1 + x_2).((\neg x_1) + (\neg x2))$
- ANF: $x_1 \oplus x_2$.

Now consider the function on two variables represented by $x_1 + x_2$. The normal forms for this function are

- DNF: $x_1.(\neg x_2) + x1.x2 + (\neg x1).x2$
- CNF: $(x_1 + x_2)$
- ANF: $x_1 \oplus x_2 \oplus x_1.x_2$.

The advantage of normal forms is that we can check for equivalence of two functions very easily. The problem with normal forms is that the formulas can get very large and are often not of practical use.

We will come back to this later in the course.

6 Shannon's expansion

If $f(x_1, x_2, \ldots, x_n)$ is a formula then the co-factor of f with respect to x_i is the formula obtained by substituting $x_i = 1$ in f. This formula no longer depends on x_i and is denoted by f_{x_i}. The co-factor of f with respect to $\neg x_i$ is the formula obtained by substituting $x_i = 0$ in f, and is denoted by $f_{\overline{x_i}}$.

Then, Shannon's decomposition states that

$$\begin{aligned} f &= x_i.f_{x_i} + \overline{x_i}.f_{\overline{x_i}} \\ &= x_i.f_{x_i} \oplus \overline{x_i}.f_{\overline{x_i}} \end{aligned}$$

Shannon's decomposition plays an important role in many verification problems, and in particular, is used to generate an ROBDD canonical representation of a Boolean function.

7 Quantification

Suppose a Boolean function f has a formula on n variables $x_1, x_2, \ldots x_n$. Then we define the formula

$$\exists_{x_i} f(x_1, x_2, \ldots x_n) = f(0, x_2, x_3, \ldots, x_n) + f(1, x_2, x_3, \ldots, x_n)$$

Things to note: this formula does not depend on x_1. Whenever it is true for some combination of $x_2, x_3, \ldots x_n$, then for that combination, there exists an assignment to x_1 such that the formula f evaluates to 1.

Now consider the formula

$$\forall x_i f(x_1, x_2, \ldots x_n) = f(0, x_2, x_3, \ldots, x_n).f(1, x_2, x_3, \ldots, x_n)$$

This formula also does not depend on x_1. If it is 1 for some combination of $x_2, x_3, \ldots x_n$, then for this combination, no matter what value is assigned to x_1, the formula f evaluates to 1.

These quantifiers are of great importance in describing properties of Boolean functions, and hence, in formal verification.

8 Assignment

1. Show that in a Boolean algebra, any element x can be written as

   $$\Pi_{a \leq \overline{x}} \overline{a}$$

   From this it follows that any Boolean function can be represented as a product of sums formula.

2. If f, g are formulas on n variables then show that
   - It is not always true that $\exists_x(f+g) = (\exists_x f) + (\exists_x g)$
   - It is not always true that $\exists_x(f.g) = (\exists_x f).(\exists_x g)$
   - $\exists_x f = (\neg(\forall_x(\neg f)))$
3. If f is a formula on n variables then show that
$$\exists_x \exists_y f \;=\; \exists_y \exists_x f$$
4. If f is a formula on n variables then show that
$$\forall_x \forall_y f \;=\; \forall_y \forall_x f$$
5. Let $f(x_1, x_2, x_3, \ldots x_4)$ be the Boolean formula
$$(x_1.(\neg x_2).(x_1 + x_2 + x_3)) + ((\neg x_1).(\neg x_4).(x_2 + x_3 + x4))$$
   - Find the DNF and CNF of f.
   - Write $\exists_{x_2} f$ as a formula.
   - Write $\forall_{x_3} f$ as a formula.