CS:APP Web Aside DATA:BOOL: More on Boolean Algebra and Boolean Rings*

Randal E. Bryant
David R. O'Hallaron

December 29, 2014

Notice

The material in this document is supplementary material to the book Computer Systems, A Programmer's Perspective, Third Edition*, by Randal E. Bryant and David R. O'Hallaron, published by Prentice-Hall and copyrighted 2016. In this document, all references beginning with "CS:APP3e " are to this book. More information about the book is available at* `csapp.cs.cmu.edu`*.*

1 Motivation

Since binary values are at the core of how computers encode, store, and manipulate information, a rich body of mathematical knowledge has evolved around the study of the values 0 and 1. This started with the work of George Boole (1815–1864) around 1850 and thus is known as *Boolean algebra*. Boole observed that by encoding logic values TRUE and FALSE as binary values 1 and 0, he could formulate an algebra that captures the properties of propositional logic.

~	
0	1
1	0

&	0	1
0	0	0
1	0	1

|	0	1
0	0	1
1	1	1

^	0	1
0	0	1
1	1	0

Figure 1: **Operations of Boolean algebra.** Binary values 1 and 0 encode logic values TRUE and FALSE, while operations `~`, `&`, `|`, and `^` encode logical operations NOT, AND, OR, and EXCLUSIVE-OR, respectively.

2 From Logic to Algebra

There is an infinite number of different Boolean algebras, where the simplest is defined over the two-element set $\{0, 1\}$. Figure 1 defines several operations in this Boolean algebra. Our symbols for representing these operations are chosen to match those used by the C bit-level operations, as will be discussed later. The Boolean operation ~ corresponds to the logical operation NOT, denoted in propositional logic as $\neg$. That is, we say that $\neg P$ is true when P is not true, and vice-versa. Correspondingly, $\sim p$ equals 1 when p equals 0, and vice-versa. Boolean operation & corresponds to the logical operation AND, denoted in propositional logic as $\wedge$. We say that $P \wedge Q$ holds when both P and Q are true. Correspondingly, p & q equals 1 only when $p = 1$ and $q = 1$. Boolean operation | corresponds to the logical operation OR, denoted in propositional logic as $\vee$. We say that $P \vee Q$ holds when either P or Q is true. Correspondingly, $p \mid q$ equals 1 when either $p = 1$ or $q = 1$. Boolean operation ^ corresponds to the logical operation EXCLUSIVE-OR, denoted in propositional logic as $\oplus$. We say that $P \oplus Q$ holds when either P or Q are true, but not both. Correspondingly, $p \hat{} q$ equals 1 when either $p = 1$ and $q = 0$, or $p = 0$ and $q = 1$.

Claude Shannon (1916–2001), who later founded the field of information theory, first made the connection between Boolean algebra and digital logic. In his 1937 master's thesis, he showed that Boolean algebra could be applied to the design and analysis of networks of electromechanical relays. Although computer technology has advanced considerably since, Boolean algebra still plays a central role in the design and analysis of digital systems.

3 Properties of Boolean Algebras and Rings

There are many parallels between integer arithmetic and Boolean algebra, as well as several important differences. In particular, the set of integers, denoted $\mathcal{Z}$, forms a mathematical structure known as a *ring*, denoted $\langle \mathcal{Z}, +, \times, -, 0, 1 \rangle$, with addition serving as the *sum* operation, multiplication as the *product* operation, negation as the additive inverse, and elements 0 and 1 serving as the additive and multiplicative identities. The Boolean algebra $\langle \{0, 1\}, |, \&, \sim, 0, 1 \rangle$ has similar properties. Figure 2 highlights properties of these two structures, showing the properties that are common to both and those that are unique to one or the other. One important difference is that $\sim a$ is not an inverse for a under $|$.

Aside: What good is abstract algebra?
Abstract algebra involves identifying and analyzing the common properties of mathematical operations in different domains. Typically, an algebra is characterized by a set of elements, some of its key operations, and some important elements. As an example, modular arithmetic also forms a ring. For modulus n, the algebra is denoted $\langle \mathcal{Z}_n, +_n, \times_n, -_n, 0, 1 \rangle$, with components defined as follows:

$$\begin{aligned} \mathcal{Z}_n &= \{0, 1, \ldots, n-1\} \\ a +_n b &= a + b \bmod n \\ a \times_n b &= a \times b \bmod n \\ -_n a &= \begin{cases} 0, & a = 0 \\ n - a, & a > 0 \end{cases} \end{aligned}$$

Even though modular arithmetic yields different results from integer arithmetic, it has many of the same mathematical properties. Other well-known rings include rational and real numbers. **End Aside.**

Shared properties

Property	Integer ring	Boolean algebra
Commutativity	$a + b = b + a$ $a \times b = b \times a$	$a \mid b = b \mid a$ $a \,\&\, b = b \,\&\, a$
Associativity	$(a + b) + c = a + (b + c)$ $(a \times b) \times c = a \times (b \times c)$	$(a \mid b) \mid c = a \mid (b \mid c)$ $(a \,\&\, b) \,\&\, c = a \,\&\, (b \,\&\, c)$
Distributivity	$a \times (b + c) = (a \times b) + (a \times c)$	$a \,\&\, (b \mid c) = (a \,\&\, b) \mid (a \,\&\, c)$
Identities	$a + 0 = a$ $a \times 1 = a$	$a \mid 0 = a$ $a \,\&\, 1 = a$
Annihilator	$a \times 0 = 0$	$a \,\&\, 0 = 0$
Cancellation	$-(-a) = a$	$\tilde{}(\tilde{}a) = a$

Unique to Rings

Inverse	$a + -a = 0$	—

Unique to Boolean Algebras

Distributivity	—	$a \mid (b \,\&\, c) = (a \mid b) \,\&\, (a \mid c)$
Complement	— —	$a \mid \tilde{}a = 1$ $a \,\&\, \tilde{}a = 0$
Idempotency	— —	$a \,\&\, a = a$ $a \mid a = a$
Absorption	— —	$a \mid (a \,\&\, b) = a$ $a \,\&\, (a \mid b) = a$
DeMorgan's laws	— —	$\tilde{}(a \,\&\, b) = \tilde{}a \mid \tilde{}b$ $\tilde{}(a \mid b) = \tilde{}a \,\&\, \tilde{}b$

Figure 2: **Comparison of integer ring and Boolean algebra.** The two mathematical structures share many properties, but there are key differences, particularly between $-$ and $\tilde{}$.

If we replace the OR operation of Boolean algebra by the EXCLUSIVE-OR operation, and the complement operation ~ with the identity operation I—where $I(a) = a$ for all a—we have a structure $\langle \{0,1\}, \hat{}, \&, I, 0, 1\rangle$. This structure is no longer a Boolean algebra—in fact it's a ring. It can be seen to be a particularly simple form of the ring consisting of all integers $\{0, 1, \ldots, n-1\}$ with both addition and multiplication performed modulo n. In this case, we have $n = 2$. That is, the Boolean AND and EXCLUSIVE-OR operations correspond to multiplication and addition modulo 2, respectively. One curious property of this algebra is that every element is its own additive inverse: $a \hat{} I(a) = a \hat{} a = 0$.

Aside: Who, besides mathematicians, care about Boolean rings?
Every time you enjoy the clarity of music recorded on a CD or the quality of video recorded on a DVD, you are taking advantage of Boolean rings. These technologies rely on *error-correcting codes* to reliably retrieve the bits from a disk even when dirt and scratches are present. The mathematical basis for these error-correcting codes is a linear algebra based on Boolean rings. **End Aside.**

4 General forms of Boolean Algebras and Rings

We can extend the four Boolean operations to also operate on bit vectors, i.e., strings of zeros and ones of some fixed length w. We define the operations over bit vectors according their applications to the matching elements of the arguments. Let a and b denote the bit vectors $[a_{w-1}, a_{w-2}, \ldots, a_0]$ and $[b_{w-1}, b_{w-2}, \ldots, b_0]$, respectively. We define $a \mathbin{\&} b$ to also be a bit vector of length w, where the ith element equals $a_i \mathbin{\&} b_i$, for $0 \leq i < w$. The operations $|$, $\hat{}$, and ~ are extended to bit vectors in a similar fashion

As examples, consider the case where $w = 4$, and with arguments $a = [0110]$ and $b = [1100]$. Then the four operations $a \mathbin{\&} b$, $a \mid b$, $a \hat{} b$, and $\tilde{} b$ yield

$$\begin{array}{r} 0110 \\ \& \;\; 1100 \\ \hline 0100 \end{array} \qquad \begin{array}{r} 0110 \\ \mid \;\; 1100 \\ \hline 1110 \end{array} \qquad \begin{array}{r} 0110 \\ \hat{} \;\; 1100 \\ \hline 1010 \end{array} \qquad \begin{array}{r} \\ \tilde{} \;\; 1100 \\ \hline 0011 \end{array}$$

To express this generalization in algebraic terms, let $\{0,1\}^w$ denote the set of all strings of zeros and ones having length w, and a^w denote the string consisting of w repetitions of symbol a. Then one can see that the resulting algebras: $\langle \{0,1\}^w, |, \&, \tilde{}, 0^w, 1^w\rangle$ and $\langle \{0,1\}^w, \hat{}, \&, I, 0^w, 1^w\rangle$ form Boolean algebras and rings, respectively. Each value of w defines a different Boolean algebra and a different Boolean ring.

Aside: Are Boolean rings the same as modular arithmetic?
The two-element Boolean ring $\langle \{0,1\}, \hat{}, \&, I, 0, 1\rangle$ is identical to the ring of integers modulo two $\langle \mathcal{Z}_2, +_2, \times_2, -_2, 0, 1\rangle$. The generalization to bit vectors of length w, however, yields a very different ring from modular arithmetic. **End Aside.**

5 Representing and Manipulating Sets

One useful application of bit vectors is to represent finite sets. We can encode any subset $A \subseteq \{0, 1, \ldots, w-1\}$ with a bit vector $[a_{w-1}, \ldots, a_1, a_0]$, where $a_i = 1$ if and only if $i \in A$. For example, (recalling that we write a_{w-1} on the left and a_0 on the right), bit vector $a \doteq [01101001]$ encodes the set $A = \{0, 3, 5, 6\}$, while

bit vector $b \doteq [01010101]$ encodes the set $B = \{0, 2, 4, 6\}$. With this way of encoding sets, Boolean operations | and & correspond to set union and intersection, respectively, and ~ corresponds to set complement. Continuing our earlier example, the operation a & b yields bit vector $[01000001]$, while $A \cap B = \{0, 6\}$.

In fact, for any set S, the structure $\langle \mathcal{P}(S), \cup, \cap, \bar{}, \emptyset, S \rangle$ forms a Boolean algebra, where $\mathcal{P}(S)$ denotes the set of all subsets of S, and $\bar{}$ denotes the set complement operator. That is, for any set A, its complement is the set $\overline{A} = \{a \in S | a \notin A\}$. The ability to represent and manipulate finite sets using bit-vector operations is a practical outcome of a deep mathematical principle.

Index