DORDT COLLEGE

Digital Collections @ Dordt

Faculty Work: Comprehensive List

1-2016

Discrete Mathematics: Chapter 7, Posets, Lattices, & Boolean Algebra

Calvin Jongsma
Dordt College, calvin.jongsma@dordt.edu

Follow this and additional works at: http://digitalcollections.dordt.edu/faculty_work

Part of the Christianity Commons, Computer Sciences Commons, and the Mathematics Commons

Recommended Citation
Jongsma, Calvin, "Discrete Mathematics: Chapter 7, Posets, Lattices, & Boolean Algebra" (2016). *Faculty Work: Comprehensive List.* Paper 427.
http://digitalcollections.dordt.edu/faculty_work/427

This Book Chapter is brought to you for free and open access by Digital Collections @ Dordt. It has been accepted for inclusion in Faculty Work: Comprehensive List by an authorized administrator of Digital Collections @ Dordt. For more information, please contact ingrid.mulder@dordt.edu.

Discrete Mathematics: Chapter 7, Posets, Lattices, & Boolean Algebra

Abstract
Algebra deals with more than computations such as addition or exponentiation; it also studies relations. Calculus touches on this a bit with locating extreme values and determining where functions increase and decrease; and in elementary algebra you occasionally "solve" inequalities involving the order relations of $<$ or $\leq$, but this almost seems like an intrusion foreign to the main focus, which is making algebraic calculations.

Relational ideas have become more important with the advent of computer science and the rise of discrete mathematics, however. Many contemporary mathematical applications involve binary or n-ary relations in addition to computations. We began discussing this topic in the last chapter when we introduced equivalence relations. In this chapter we will explore other kinds of relations (these will all be binary relations here), particularly ones that impose an order of one sort or another on a set. This will lead us to investigate certain order-structures (posets, lattices) and to introduce an abstract type of algebra known as Boolean Algebra. Our exploration of these ideas will nicely tie together some earlier ideas in logic and set theory as well as lead us into areas that are of crucial importance to computer science.

Keywords
partially ordered sets, lattice theory, Boolean algebra, equivalence relations

Disciplines
Christianity | Computer Sciences | Mathematics

Comments
- From Discrete Mathematics: An Integrated Approach, a self-published textbook for use in Math 212
- © 2016 Calvin Jongsma

Creative Commons License

This work is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works 4.0 License.

This book chapter is available at Digital Collections @ Dordt: http://digitalcollections.dordt.edu/faculty_work/427

✠ Chapter 7 ✠

POSETS, LATTICES, & BOOLEAN ALGEBRA

7.1 Partially Ordered Sets

Elementary mathematics tends to focus lopsidedly on computational structures. You learn arithmetic in grade school, and when you've got that mastered, you move on to tackle algebra. Algebra is presented as a collection of problem-solving techniques and as a generalized symbolic arithmetic. You make computations with unknowns to determine their hidden values, and you manipulate equations that identify two computational expressions to determine just how the individual variables are related. Toward the end of secondary school or the start of college, you study calculus, which once again computes things related to equations. You determine slopes of tangent lines and rates of change and areas under curves using computational algorithms.

But there is a whole other side to algebra that is largely ignored. Algebra deals with more than computations such as addition or exponentiation; it also studies relations. Calculus touches on this a bit with locating extreme values and determining where functions increase and decrease; and in elementary algebra you occasionally "solve" inequalities involving the order relations of $<$ or $\leq$, but this almost seems like an intrusion foreign to the main focus, which is making algebraic calculations.

Relational ideas have become more important with the advent of computer science and the rise of discrete mathematics, however. Many contemporary mathematical applications involve binary or n-ary relations in addition to computations. We began discussing this topic in the last chapter when we introduced equivalence relations. In this chapter we will explore other kinds of relations (these will all be *binary relations* here), particularly ones that impose an order of one sort or another on a set. This will lead us to investigate certain order-structures (posets, lattices) and to introduce an abstract type of algebra known as *Boolean Algebra*. Our exploration of these ideas will nicely tie together some earlier ideas in logic and set theory as well as lead us into areas that are of crucial importance to computer science.

Partial and Total Orders on a Set

The prototype for partial orderings is the subset relation holding among a collection of sets. We'll begin with this example and then define the more general notion on any set.

✠ **EXAMPLE 7.1 - 1**

Let S be any set and let $\mathcal{A}$ be some collection of its subsets. Discuss the properties of the subset relation $\subseteq$ on elements of $\mathcal{A}$.

Solution

The subset relation is reflexive and transitive, but it is generally not symmetric.
For if X is any element of $\mathcal{A}$, then $X \subseteq X$; and whenever $X \subseteq Y$ and $Y \subseteq Z$, $X \subseteq Z$.
Furthermore, $X \subseteq Y$ does not usually force $Y \subseteq X$. In particular, it fails if $\mathcal{A}$ is the full power set $\mathcal{P}(S)$. (Think: for what sort of collection $\mathcal{A}$ might symmetry hold?)
What is always true, however, is that if $X \subseteq Y$ and $Y \subseteq X$, then $X = Y$.
This is known as the *antisymmetric property* of $\subseteq$.

DEFINITION 7.1 - 1: Antisymmetric Relations

A binary relation R on a set $\mathcal{A}$ is ***antisymmetric*** *iff for all elements x and y of $\mathcal{A}$, whenever $x\,R\,y$ and $y\,R\,x$, then $x = y$.*

DEFINITION 7.1 - 2: Partial Order on a Set; Poset

a) *A binary relation R on a set $\mathcal{A}$ is a* ***partial order*** *on $\mathcal{A}$ iff R is reflexive, antisymmetric, and transitive.*
b) *A set $\mathcal{A}$ together with a partial order R on $\mathcal{A}$ is called a* ***partially ordered set/poset****.*

To be picky, therefore, a poset is an ordered pair $\langle \mathcal{A}, R \rangle$. However, when the intended partial order is understood, it is customary to refer to the set $\mathcal{A}$ itself as the poset.

What about the partial-order relation? Binary relations link elements in a set, but why is a relation that is reflexive, antisymmetric, and transitive called a *partial order*? Why is the term *order* used? And is there such a thing as a *total* order? It turns out there is. Defining that and looking at another example might help us clarify the terminology.

DEFINITION* 7.1 - 3: *Connected Relations

A binary relation R on a set $\mathcal{A}$ is ***connected*** *iff for all elements x and y of $\mathcal{A}$, either $x\,R\,y$ or $y\,R\,x$.*

DEFINITION* 7.1 - 4: *Total Order/Linear Order on a Set

A binary relation R on a set $\mathcal{A}$ is a ***total order/linear order*** *on $\mathcal{A}$ iff R is a connected partial order on $\mathcal{A}$.*

✠ **EXAMPLE 7.1 - 2**

Show that the relation $\leq$ is a total order on the set of real numbers $\mathbb{R}$.

Solution

It should be fairly obvious that $\leq$ is reflexive, antisymmetric, transitive, and connected. Thus $\leq$ is a total or linear order on $\mathbb{R}$. This relation orders real numbers in a linear way: the real numbers can be placed on a number line in the usual way. The ordering here is total because any two real numbers can be compared via this relation.

Every total order R on a set $\mathcal{A}$ is of this sort. We can use points on a vertical line to position the elements of $\mathcal{A}$ in a way that models the relation R: place x below y iff $x\,R\,y$. In fact, since total orders act so much like $\leq$ on sets of numbers, this symbol is used for any total order, whether or not it represents the ordinary relation of "less than or equal to". This practice is extended to partial orders as well: in an *abstract setting*, rather than use a symbol like R to stand for a partial order, the familiar symbol $\leq$ is often used, even though no connection to ordinary numerical order is being suggested and even though no assumption is being made about the relation being connected.* In a concrete setting, of course, the appropriate symbol for the given relation will be used instead, if a standard one exists.

Now, how does a partial order compare to a total one? Well, it's a total order minus the connected property. In a poset $\mathcal{A}$ not every pair of elements needs to be comparable. However, if some subset of $\mathcal{A}$ is such that any two of *its* elements can be compared, that part of $\mathcal{A}$ will be ordered like a line. The entire set $\mathcal{A}$, therefore, may contain a number of these line-like paths, possibly with some branching in various places.

✠ **EXAMPLE 7.1 - 3**

Show that the following divides-relations are partial orders on $\mathcal{A}$. Is either a total order?

a) $m \mid n$ iff m divides n, $\mathcal{A} = \mathbb{N}$.

b) $m \mid n$ iff m divides n, $\mathcal{A} =$ the set of all positive-integer divisors of 36.

Solution

a) Since $n \mid n$ for all $n \in \mathbb{N}$, the relation is reflexive. It is also antisymmetric: if $m \mid n$ and $n \mid m$, then $m = n$. And it is transitive: if $k \mid m$ and $m \mid n$, then $k \mid n$.
It is not connected, however. For example, neither 2 nor 3 divides the other. Thus, the relation is not a total order.

b) Here the divisibility relation is restricted to $\mathcal{A} = \{1, 2, 3, 4, 6, 9, 12, 18, 36\}$.
The relation here remains reflexive, antisymmetric, transitive, and not connected.
Thus, it too is not a total order on $\mathcal{A}$.

---

* Mathematicians dislike inventing new symbols when old ones work just as well and suggest something of the intended meaning.

Finite posets can be graphed by a vertically ordered diagram in which elements are connected to ones above them by an edge if they are related to them by the relation. Since order relations are transitive, any relation between elements that can be deduced by following the edges upward are left implicit instead of cluttering up the diagram. These graphs are called *Hasse diagrams* after the twentieth-century German number theorist Helmut Hasse.

✠ **EXAMPLE 7.1 - 4**

Diagram the following posets:

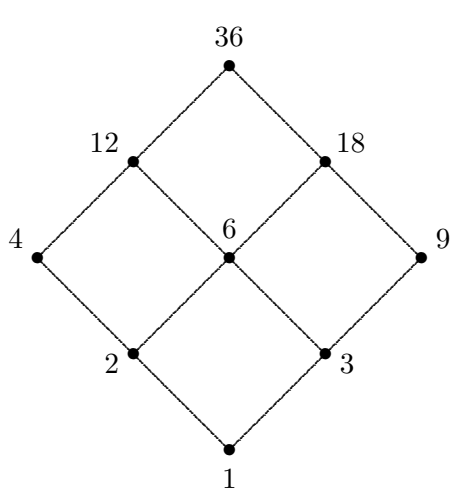
a) The poset of Example 3b: the divisors of 36 ordered by $m \mid n$.

b) The poset $\mathcal{P}(S)$ for $S = \{0, 1, 2\}$, ordered by $R \subseteq T$.

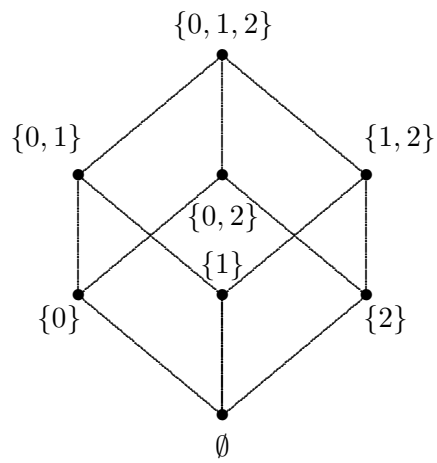
Solution

These are graphed by the following Hasse diagrams.

a)



b)



Strict and Partial Orders

Given a partial order $\leq$ on a set, a related strict order $<$ can also be defined and is quite useful in a variety of settings. We will first define the notion of strict order in general, using the properties of being irreflexive and asymmetric, and then relate it to partial order. It helps here at times to use the set-theoretic definition of a binary relation as a set of ordered pairs.

DEFINITION* 7.1 - 5: *Irreflexive and Asymmetric Relations

a) *A binary relation R on a set $\mathcal{A}$ is* ***irreflexive*** *iff $(x, x) \notin R$ for all $x \in \mathcal{A}$.*

b) *A binary relation R on a set $\mathcal{A}$ is* ***asymmetric*** *iff whenever $(x, y) \in R$, then $(y, x) \notin R$.*

Irreflexive and asymmetric relations are thus strongly non-reflexive and non-symmetric. Ordinary $<$ on $\mathbb{R}$ is both irreflexive and asymmetric.

Using these notions, we now define a strict order as follows.

DEFINITION* 7.1 - 6: *Strict Order

A binary relation R on a set $\mathcal{A}$ is a ***strict order*** *on $\mathcal{A}$ iff R is irreflexive and transitive.*

It's easy to show that *any transitive irreflexive relation is also asymmetric*; i.e., *every strict order is asymmetric* (see Exercise 24a), so asymmetry comes for free here. Similarly, every strict order is antisymmetric (see Exercise 24b).

Given our definitions, we can explore the connection between partial orders and strict orders. Based on what we know of ordinary $\leq$ and $<$, we expect the following result might be true. Which it is.

PROPOSITION* 7.1 - 1: *Strict and Partial Orders

Let $\mathcal{A}$ be any set.

a) *If $\leq$ is a partial order on $\mathcal{A}$, then the relation $<$ defined by $x < y \leftrightarrow x \leq y \wedge x \neq y$ is a strict order on $\mathcal{A}$.*

b) *If $<$ is a strict order on $\mathcal{A}$, then the relation $\leq$ defined by $x \leq y \leftrightarrow x < y \vee x = y$ is a partial order on $\mathcal{A}$.*

***Proof*:**

– For part a), suppose $\leq$ is a partial order with $<$ defined by $x < y$ iff $x \leq y$ and $x \neq y$.
[To show that $<$ is a strict order, we must show that it is irreflexive and transitive.]
The relation $<$ is clearly irreflexive: if $x < y$, then $x \neq y$ according to the definition.
Now suppose $x < y$ and $y < z$. Then by the transitivity of $\leq$, $x \leq z$.
But $x \neq z$; for if $x = z$, the antisymmetry of $\leq$ would mean, for instance, that $x = y$, which contradicts the supposition $x < y$.
Thus $x < z$, which establishes that $<$ is transitive.

– Part *b* is left as an exercise (see Exercise 26a). ■

A more set-theoretic way to go from a partial order to its associated strict order is to simply delete all pairs (x, x) from the partial order (see Exercise 26b). Conversely, starting with a strict order, adjoining all pairs (x, x) generates the associated partial order; i.e., the *reflexive closure* of a strict order remains transitive and is both reflexive and antisymmetric (see Exercise 26c).

As suspected, *Proposition* 1 tells us that the some of the same relations hold between $\leq$ and $<$ in general as they do for the ordinary numerical orders denoted by these symbols. This provides some justification for using these symbols in a more abstract setting, even though this may seem confusing at first. The meaning of $<$ in any concrete setting, of course, is naturally dependent on the meaning of $\leq$; if $\leq$ stands for ordinary numerical inequality, then $<$ denotes strict numerical inequality, but it will have other meanings in other contexts. The same thing is true in reverse; the concrete meaning of $\leq$ depends on the meaning given to $<$.

✠ **EXAMPLE 7.1 - 5**

Discuss the meaning of the strict orders $<$ associated with the following partial orders:

a) if $\leq$ is the partial order of divisibility on the set of natural numbers.

b) if $\leq$ represents the subset ordering on a collection of sets.

Solution

a) If $\leq$ denotes the relation "is a divisor of," then $<$ indicates proper divisibility: here $m < n$ means m is a proper divisor of n.

b) If $\leq$ denotes "is a subset of," then $<$ indicates proper subset inclusion: here $S < T$ denotes the concrete relation $S \subset T$.

Extending Partial Orders to Cartesian Products and Strings

Suppose $\mathcal{A}_1$ and $\mathcal{A}_2$ are two sets partially ordered by $\leq_1$ and $\leq_2$ respectively.* Taking the Cartesian product of the two posets, we get $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$. This set can itself be converted into a poset via the *dictionary* (*lexicographical*) *order relation*, which builds upon the ordering relations on the coordinate components. We will first define this ordering relation and then show that it is indeed a partial order, making $\mathcal{A}$ a poset.

* When it is clear which partial order is involved, we will simply write $\leq$ without a subscript, though the order relations involved may be very different.

DEFINITION* 7.1 - 7: *Dictionary-Order Relation

If $\langle \mathcal{A}_1, \leq_1 \rangle$ and $\langle \mathcal{A}_2, \leq_2 \rangle$ are two posets, then the ***dictionary order*** $\leq$ *on $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ is given by $(x_1, x_2) \leq (y_1, y_2)$ iff $x_1 <_1 y_1 \vee (x_1 = y_1 \wedge x_2 \leq_2 y_2)$.*

PROPOSITION* 7.1 - 2: *Partially Ordered Cartesian Products

The Cartesian product $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ of two posets $\mathcal{A}_1$ and $\mathcal{A}_2$ forms a poset under the dictionary order $\leq$.

***Proof*:**

We will indicate what needs to be argued without carrying along the order subscripts.

- $\leq$ *is reflexive*: $(x_1, x_2) \leq (x_1, x_2)$ because $x_1 = x_1$ and $x_2 \leq x_2$.
- $\leq$ *is antisymmetric*: if $(x_1, x_2) \leq (y_1, y_2)$ and $(y_1, y_2) \leq (x_1, x_2)$, then $x_1 = y_1$ and $x_2 = y_2$ (see Exercise 30a), and so $(x_1, x_2) = (y_1, y_2)$.
- $\leq$ *is transitive*: if $(x_1, x_2) \leq (y_1, y_2)$ and $(y_1, y_2) \leq (z_1, z_2)$, then $(x_1, x_2) \leq (z_1, z_2)$ (see Exercise 30b). ■

Now that the case of two sets is settled, we can generalize to any finite number of sets. Let's first see how to go from two sets to three sets. How should $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{A}_3$ be ordered? If we think of ordered triples as nested ordered pairs [see Exercise 4.3 - 20: (x_1, x_2, x_3) is defined there as $((x_1, x_2), x_3)$], the ordering is immediate. We have a partial ordering on $\mathcal{A}_1 \times \mathcal{A}_2$ by *Proposition* 2, and that partial order joined with the one on $\mathcal{A}_3$ gives the order on $(\mathcal{A}_1 \times \mathcal{A}_2) \times \mathcal{A}_3$, again by *Definition* 7 and *Proposition* 2.

This amounts to the following partial order on $\mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{A}_3$: $(x_1, x_2, x_3) \leq (y_1, y_2, y_3)$ iff $x_1 < y_1$, or $x_1 = y_1 \wedge x_2 < y_2$, or $x_1 = y_1 \wedge x_2 = y_2 \wedge x_3 \leq y_3$ (see Exercise 34). This can be generalized to the Cartesian product of n sets, though writing out the order criterion gets quite messy.

Finally, we'll consider the case of ordering all finite strings (words) composed of characters coming from the same poset $\mathcal{A}$. Suppose $\mathcal{A}$ is a partially ordered alphabet, $\mathcal{A}^k$ denotes all strings of length k, and $\mathcal{A}^* = \bigcup_{k=1}^{\infty} \mathcal{A}^k$ denotes all strings of finite length formed from the given alphabet. First note that two strings are equal if they have exactly the same number of characters and the same characters in each place.

We'll define a strict-order relation $<$ on $\mathcal{A}^*$ as follows. To compare string $x_1 \cdots x_k$ with $y_1 \cdots y_n$, begin at the far left end and compare successive prefix segments of the two words. Suppose $x_1 \cdots x_m$ and $y_1 \cdots y_m$ are the longest prefix segments of the two words where they completely agree (m might be 0). Then $x_1 \cdots x_k < y_1 \cdots y_n$ iff $m = k$ and $m < n$, or $m < k$ and $m < n$ and $x_{m+1} < y_{m+1}$.

If the alphabet $\mathcal{A}$ is totally ordered (as genuine alphabets are), then $\mathcal{A}^*$ will be totally ordered by this relation (see Exercise 36).

Extreme Elements in Posets

In a totally ordered set, any two elements can be compared, without exception. This gives a fairly simple order structure to the set – it's just a line; which makes it less interesting in a way. Posets have more varied and complex structures. Given a pair of elements in a poset, we may or may not be able to compare them via the order relation. But even if we can't compare them, we might still be able to compare them to some common third element. Perhaps they are both *smaller than* or both *greater than* some other elements of the poset. We will look at this situation in more detail in the next section on lattices, but here we will introduce a couple of ideas and terms that are already needed for what we will discuss next.

DEFINITION* 7.1 - 8: *Extremal Elements

Suppose $\langle \mathcal{A}, \leq \rangle$ is a poset, M and m are elements of $\mathcal{A}$, and S is a subset of $\mathcal{A}$.

a) *M is a* ***maximal element of S*** *iff M is in S and there is no x in S such that $M < x$;*
M is a ***maximum of S*** *iff M is in S and $x \leq M$ for all x in S.*

b) *m is a* ***minimal element of S*** *iff m is in S and there is no x in S such that $x < m$;*
m is a ***minimum of S*** *iff m is in S and $m \leq x$ for all x in S.*

Note that all these extremal elements belong to the set of elements they bound: that's part of their definition. Extrema (minimum, maximum) may or may not exist for a given subset, but if they do, they will be unique (see Exercise 27b). Connections holding among these various extremal elements are explored in Exercises 27-29.

✠ **EXAMPLE 7.1 - 6**

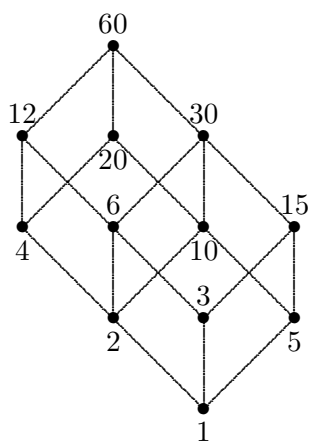
Identify extreme elements in the following posets:

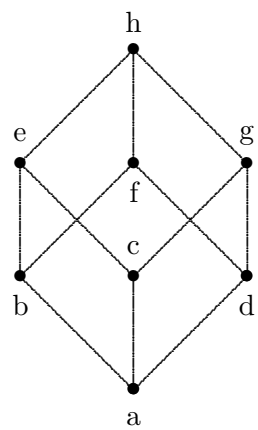
a) The divisors of 60, ordered by divisibility.

b) The set $\{a, b, c, d, e, f, g, h\}$, ordered like the subsets of $\{0, 1, 2\}$ (see Example 4).

Solution

The Hasse diagrams of these posets are given below. Note how the "dimensionality" of the first diagram corresponds to the number of prime factors in the factorization of 60 (see also Exercises 1 – 7).





Every non-empty subset S of the given posets will have maximal and minimal elements (nothing above them or below them respectively), though they need not have either a maximum or a minimum. The set $S = \{b, d\}$ from the second poset is such an example; both elements are maximal as well as minimal for S, but S has no maximum or minimum.

Constructing Total Orders from Partial Orders

In some situations, you may have a natural partial order on a set but require a total order, one that respects or extends the given partial order. This is the case whenever a composite task contains a number of different subactivities, each with its own linear ordering. If you're constructing a building, for example, you have structural components, electrical components, plumbing components, furnishing components, etc. Some of these may require certain activities to be done before others, though not all of them may be so related. However, to put up a building in real time, you have to sequentialize (totally order) all the actions and in a way that is compatible with the various order priorities that do exist. Thus, the partial order needs to be embedded within a total order. Such a total order always exists for finite posets, as we will show, though it will not be not unique unless the original order is already a total order.

PROPOSITION* 7.1 - 3: *Embedding a Partial Order in a Total Order

Finite posets can be embedded in totally ordered sets whose order extends the original order.

***Proof*:**

Let $\langle \mathcal{A}, \leq \rangle$ be a finite poset. $\mathcal{A}$ contains a minimal element (see Exercise 27b): choose one and call it a_1.

The set $\mathcal{A} - \{a_1\}$ remains a poset. If it is non-empty, it too has a minimal element: choose one and call it a_2.

Choose $a_3, \ldots, a_n$ similarly until all elements in $\mathcal{A}$ have been chosen in some order.

Now order the elements of $\mathcal{A}$ by how they were chosen: $a_i \leq a_j$ iff $i \leq j$. This forms a total order on $\mathcal{A}$.

Moreover, if $a_k \leq a_m$ in the original order, it remains so in the new sequentialized order: choosing a_m is done only after everything that precedes it, including a_k, has been placed into the sequence, because at each stage a minimal element is chosen.

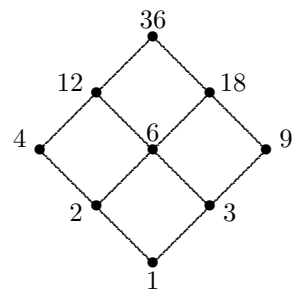
Thus the total order exists and respects the original partial order. ■

✠ **EXAMPLE 7.1 - 7**

Convert the poset of divisors of 36 into a totally ordered set in two ways.

Solution

- One total order is created by choosing lowest level elements, moving left to right within the level: $1; 2, 3; 4, 6, 9; 12, 18; 36$.
- Another total order results from choosing the left-most minimal element each time: $1, 2, 4; 3, 6, 12; 9, 18, 36$.
- Both of these total orders respect the original partial order. They each stretch and squeeze the branching partial order together in some way to convert it into a single line of elements.



EXERCISE SET 7.1

Problems 1 - 8: Divisor Posets

The following explore connections between Hasse diagrams of divisor posets and prime factorization.

1. *Divisors of* 100
   a. Draw the Hasse diagram for the poset of the divisors of 100. Is this poset totally ordered? Explain.
   b. How does the shape of the diagram relate to the prime factorization of 100? Explain.
2. *Divisors of* 59
   a. Draw the Hasse diagram for the poset of the divisors of 59. Is this poset totally ordered? Explain.
   b. How does the shape of the diagram relate to the prime factorization of 59? Explain.

*3. *Divisors of* 64
   a. Draw the Hasse diagram for the poset of the divisors of 64. Is this poset totally ordered? Explain.
   b. How does the shape of the diagram relate to the prime factorization of 64? Explain.

4. *Divisors of* 392
   a. Draw the Hasse diagram for the poset of the divisors of 392. Is this poset totally ordered?
   b. How does the shape of the diagram relate to the prime factorization of 392? Explain.

*5. *Divisors of* 84

a. Draw the Hasse diagram for the poset of the divisors of 84. Is this poset totally ordered? Explain.

b. How does the shape of the diagram relate to the prime factorization of 84? Explain.

c. If the top and bottom points are deleted from this poset, leaving only the proper divisors of 84, is the resulting set still a poset? Explain.

6. *Divisors of* 252

a. Draw the Hasse diagram for the poset of the divisors of 252. Is this poset totally ordered? Explain.

b. How does the shape of the diagram relate to the prime factorization of 252? Explain.

c. If the top and bottom points are deleted from this poset, leaving only the proper divisors of 252, is the resulting set still a poset? Explain.

EC 7. *Hasse Diagrams and Prime Factorization*

a. Based on your work in Problems 1 - 6, conjecture what the Hasse diagram looks like for the poset of the divisors of 900. Then draw the diagram to verify your conjecture.

b. What sort of Hasse diagram do you think the poset of the divisors of 420 has? How could this be envisioned or described? Explain your answers; you do not need to draw the diagrams.

c. What sort of Hasse diagram do you think the poset of the divisors of 2310 has? How could this be envisioned or described? Explain your answers; you do not need to draw the diagrams.

8. Show that the divisibility relation is not a partial order on the set of integers $\mathbb{Z}$. Which property is lacking?

Problems 9 - 12: Hasse Diagrams of Small Posets

The following problems explore Hasse diagrams of some small posets. Note that elements of a poset should be pictured as isolated points if they can't be compared with any other points.

*9. Draw a baseball diamond (connect home plate (4) to first base (1) to second (2) to third (3) and back to home). Then extend the foul lines out to the right field (5) and left field (7) foul poles and connect these points with the wall in straight-away center field (6). Is this diagram a Hasse diagram for a poset? Explain.

10. Draw all possible distinct Hasse diagrams for a two-element poset.

11. Draw all possible distinct Hasse diagrams for a three-element poset.

12. Draw all possible distinct Hasse diagrams for a four-element poset.

Problems 13 - 15: Identifying Extremal Elements

The following problems ask you to identify various extremal elements in a poset, where they exist.

*13. For D, the divisor poset of 60 (see Example 6a), work the following.

a. Let P be the set of all proper divisors of 60 (excluding 1 and 60). Explain why P is a poset. Does have P have a maximum or minimum? Identify all minimal and maximal elements of P.

b. Let $U = \{x \in D : 4 \leq x, 10 \leq x\}$, the set of upper bounds for $\{4, 10\}$. List the elements of U. What is its minimum?

c. Let $L = \{x \in D : x \leq 4, x \leq 10\}$, the set of lower bounds for $\{4, 10\}$. List the elements of L. What is its maximum?

14. Let $S = \{0, 1, 2, 3\}$. For the subset-poset $\mathcal{P}(S)$ (see Example 4b for a similar poset), work the following.

a. Let $U = \{R \subseteq S : \{0, 1\} \subseteq R, \{0, 2\} \subseteq R\}$, the set of all upper bounds for $\{\{0, 1\}, \{0, 2\}\}$. List the elements of U and identify any minimal elements for U. Is any of these a minimum for U? How are your answers related to set-theoretic operations on these sets?

b. Let $L = \{R \subseteq S : R \subseteq \{0, 1\}, R \subseteq \{0, 2\}\}$, the set of all lower bounds for $\{\{0, 1\}, \{0, 2\}\}$. List the elements of L and identify any maximal elements for U. Is any of these a maximum for L? How are your answers related to set-theoretic operations on these sets?

EC 15. Let S be any set, $\mathcal{P}(S)$ be its power set, and $\mathcal{C}$ be a collection of subsets of S.

a. Explain why $\mathcal{P}(S)$ is a poset under the subset relation $\subseteq$.

b. Must $\mathcal{C}$ always contain maximal elements? minimal elements? Prove it or provide a counterexample.

c. Let $U = \{X \subseteq S : (\forall C \in \mathcal{C})(C \subseteq X)\}$, the set of all upper bounds for sets in $\mathcal{C}$. Does U have a minimum, a least upper bound? What is it in set-theoretic terms?

d. Let $L = \{X \subseteq S : (\forall C \in \mathcal{C})(X \subseteq C)\}$, the set of all lower bounds for sets in $\mathcal{C}$. Does L have a maximum, a greatest lower bound? What is it in set-theoretic terms?

Problems 16 - 20: True or False

Are the following statements true or false? Explain your answer.

*16. Any two elements x and y in a poset can be compared: either $x \leq y$ or $y \leq x$.

*17. Every non-empty subset of a poset is also a poset.

18. In order to use the dictionary ordering on two posets $\mathcal{A}_1$ and $\mathcal{A}_2$, the order relations for each poset must be identical.

*19. If M is the maximum for a set S in a poset $\mathcal{A}$, then M is a maximal element of S.

*20. Every total order is an equivalence relation.

Problems 21 - 23: Properties of Relations and Posets

The following problems consider some logical connections between properties of binary relations.

21. Prove or disprove: *Every relation that is both symmetric and antisymmetric is also reflexive.*

22. Determine a necessary and sufficient condition for an equivalence relation to be antisymmetric. When is an equivalence relation a partial ordering?

23. *Poset Duals*
Prove that if $\mathcal{A}$ is a poset under a partial order $\leq$, it also forms a poset under its converse relation: $a \overset{\frown}{\leq} b$ iff $b \leq a$. (This converse relation is symbolized by $\geq$ and its strict ordering by $>$.)

Problems 24 - 26: Strict Ordering Relations

Interpret $\leq$ and $<$ below as associated abstract partial orders and strict orders respectively. Do not assume anything about them except properties that are true of all such relations.

*24. *Properties of Strict Orders*

*a. Prove that if $<$ is a strict order (irreflexive and transitive), then it is *asymmetric*: if $x < y$, then $y \not< x$. In fact, prove the stronger condition: if $x < y$, then $y \not\leq x$.

b. Prove that if $<$ is a strict order (irreflexive and transitive), then it is *antisymmetric*: if $x < y$ and $y < x$, then $x = y$. *Hint*: when will the antecedent occur?

25. *Transitive Properties of Strict and Partial Orders*

a. Prove that if $x < y \leq z$ or $x \leq y < z$, then $x < z$.

b. Is it possible to have $x < y < z$ and $z \leq x$? Prove or disprove this in general.

*26. *Proposition 1*

*a. Prove *Proposition* 1b: *If $<$ is a strict order on $\mathcal{A}$, then $\leq$ defined by $x \leq y \leftrightarrow x < y \vee x = y$ is a partial order.*

b. Prove the set-theoretic version of *Proposition* 1a: *If $\leq$ is a partial order on $\mathcal{A}$, deleting all pairs (x, x) for $x \in \mathcal{A}$ yields a strict order $<$ on $\mathcal{A}$. Furthermore, $x < y$ iff $x \leq y$ and $x \neq y$.*

c. Prove the set-theoretic version of *Proposition* 1b: *If $<$ is a strict order on $\mathcal{A}$, adjoining all pairs (x, x) for $x \in \mathcal{A}$ yields a partial order $\leq$ on $\mathcal{A}$. Furthermore, $x \leq y$ iff $x < y$ or $x = y$.*

Problems 27 - 29: Theory of Extremal Elements

Let $\langle \mathcal{A}, \leq \rangle$ be a poset and S a non-empty subset of $\mathcal{A}$.

*27. *Maximal and Minimal Elements*

a. Give an example to show that maximal and minimal elements of S need not be unique.

*b. Prove that finite non-empty posets contain maximal and minimal elements.

28. *Maximum and Minimum Elements*
    a. Give an example to show that S need not have a maximum or a minimum.
    b. Prove that a maximum and a minimum must be unique, if they exist.

29. *Extremal Elements in Totally Ordered Sets*
    Suppose $\langle \mathcal{A}, \leq \rangle$ is a totally ordered set and S a non-empty subset of $\mathcal{A}$.
    a. Must S have maximal or minimal elements?
    b. Must S have a maximum or a minimum?
    c. How would you answer parts a and b for a finite set S?

Problems 30 - 37: Partially Ordered Cartesian Products

The following problems deal with Cartesian product posets and dictionary order.

*30. *Proposition* 2
    *a. Prove that the dictionary ordering $\leq$ of $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ is antisymmetric by considering the necessary cases and using what you know about $\leq$ and $<$.
    b. Prove that the dictionary ordering $\leq$ of $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ is transitive by considering the necessary cases and using what you know about $\leq$ and $<$.

31. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are linearly ordered sets, need $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ be linearly ordered by the dictionary order placed upon it? Prove or disprove this.

EC 32. Dixie Nary thinks the dictionary order placed on $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ is unnecessarily complicated, so she offers the following *product ordering* for consideration: $(x_1, x_2) \leq (y_1, y_2)$ iff $x_1 \leq_1 y_1$ and $x_2 \leq_2 y_2$.
    a. Is this product order a partial order? Justify your answer.
    b. Does the new product order agree with the usual dictionary order? Explain.
    c. If both component orders are total orders, is the new product order also a total order? Explain.

*33. *Strict Dictionary Order on Ordered Pairs*
    Let $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ be the Cartesian product of two posets.
    *a. If $\leq$ is the usual dictionary order on $\mathcal{A}_1 \times \mathcal{A}_2$, state and prove a necessary and sufficient condition for the associated strict order: $(x_1, x_2) < (y_1, y_2)$ iff _____ .
    b. Prove that if $<$ is defined as you stated in part a, then $<$ is a strict order, and the associated partial order $\leq$ is the dictionary order; i.e., the associated $\leq$ satisfies the biconditional stated in the definition of the dictionary order.

34. *Dictionary Order on Triples*
    a. Given that dictionary order on pairs of elements is a partial order and that triples are defined in terms of pairs, show that the order extended to the Cartesian product $\mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{A}_3$ (see the discussion following *Proposition* 2) is a partial order relation. (A lengthy argument is not required here.)
    b. Prove that the resulting partial order on $\mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{A}_3$ satisfies $(x_1, x_2, x_3) \leq (y_1, y_2, y_3)$ iff $x_1 < y_1$ or $x_1 = y_1 \wedge x_2 < y_2$ or $x_1 = y_1 \wedge x_2 = y_2 \wedge x_3 \leq y_3$.

*35. *Dictionary Order on Strings*
    Let $\mathcal{A}$ be the English alphabet, with letters ordered in the usual way, and let $\mathcal{A}^*$ denote the set of all finite strings/words formed using this alphabet. Use the definition of strict dictionary order on $\mathcal{A}^*$ to answer the following questions.
    a. Why does the string 'earlier' appear earlier in the dictionary than 'later'?
    *b. Explain why the following words are listed in their dictionary order: 'short', 'shorter', 'shortest'.

36. Prove that the order relation defined on $\mathcal{A}^*$, the set of all finite strings formed from a totally ordered alphabet $\mathcal{A}$, is a total ordering.

37. Let $\mathcal{B}^n$ denote the set of all length-n strings of 0s and 1s, and let $\leq$ denote the dictionary order on $\mathcal{B}^n$.
    a. Prove that $\mathcal{B}^n$ is a totally ordered set.
    b. Prove that if the elements of $\mathcal{B}^n$ are considered as numbers in binary notation (see Exercise 3.2-46), then $\leq$ agrees with ordinary numerical order on $\{0, 1, \ldots, 2^{n-1}\}$.

Problems 38 - 39: Embedding Posets in Totally Ordered Sets

The following problems deal with extending a partial order to make it a total order.

*38. Extend the partial order for $\mathcal{P}(\{0,1,2\})$ (see Example 4b) in two different ways to make it into a total order.

39. If a poset is infinite, can it be embedded in a totally ordered set? Prove it or disprove it.

Problems 40 - 45: Well-Ordered Sets

The following problems explore the notion of a well-ordered set, defined as follows.

DEFINITION: *A totally ordered set $\langle \mathcal{A}, \leq \rangle$ is* ***well-ordered*** *iff every non-empty subset S of $\mathcal{A}$ contains a least element (a minimum).*

40. Is $\mathbb{Z}$ under the usual $\leq$ ordering a well-ordered set? Explain.

41. Is $(0,1)$, the set of all real numbers between 0 and 1, well-ordered under the usual ordering? Explain.

42. Is every totally ordered set with a least element well ordered?

EC 43. Use mathematical induction to prove that $\mathbb{N}$ under the usual $\leq$ ordering is a well-ordered set.

44. Let $\mathbb{N}$ be ordered by the ordinary $\leq$ order. Show that $\mathbb{N} \times \mathbb{N}$ is well-ordered under the induced dictionary order.

EC 45. Prove that the following *Principle of Induction* holds for any well-ordered set $\langle \mathcal{A}, \leq \rangle$:
If T is a non-empty subset of $\mathcal{A}$, and if T contains an element whenever it contains all the predecessors of that element, then $T = \mathcal{A}$. (Note: x is a *predecessor* of y iff $x < y$.)

Problems 46 - 47: Subset Posets

The following problems involve posets ordered by subset inclusion.

46. Let $\mathcal{A}$ be the set of all *initial segments* of $\mathbb{N}$; i.e., $\mathcal{A}$ is the collection of all subsets of $\mathbb{N}$ that are of the form $\{0, 1, \ldots, n\}$ for some n.
    a. Tell why $\mathcal{A}$ forms a poset under the subset-order relation.
    b. Does $\mathcal{A}$ form a totally ordered set? Carefully prove your claim.
    c. Suppose S is a finite subset of $\mathcal{A}$. Must S have maximal or minimal elements? A maximum or a minimum? Carefully prove your answers.
    d. If S is an infinite subset of $\mathcal{A}$, must S have maximal or minimal elements? A maximum or a minimum? Explain.

47. Let $\mathcal{A}$ be the set of all *cofinite* subsets of $\mathbb{N}$; i.e., subsets of $\mathbb{N}$ whose *co*mplement in $\mathbb{N}$ is *finite*.
    a. Tell why $\mathcal{A}$ forms a poset under the subset-order relation.
    b. Suppose S is a finite decreasing chain in $\mathcal{A}$; i.e., $S = \{A_1, A_2, \ldots, A_n\}$, where $A_1 \supseteq A_2 \supseteq \cdots \supseteq A_n$. Must S have a least upper bound or a greatest lower bound in $\mathcal{A}$? If it does, must these belong to S?
    d. If S is an infinite decreasing chain of cofinite subsets $A_1 \supseteq A_2 \supseteq \cdots \supseteq A_n \supseteq \cdots$, must S have a least upper bound or a greatest lower bound in $\mathcal{A}$? If it does, must these belong to S?

HINTS TO STARRED EXERCISES 7.1

3. a. Example 4a is similar to this problem.
   b. The prime factorization of 64 is 2^5.

5. a. Example 4a is similar to this problem.
   b. The prime factorization of 84 is $2^2 \cdot 3 \cdot 7$.
   c. Review the definition of poset.

9. Review the definition of poset.

13. Example 6 is similar to this problem.
   a. Review the definition of being a poset as well as those for extremal elements.
   b. Review the definition of minimum.
   c. Review the definition of maximum.

16. [No hint.]

17. [No hint.]

19. [No hint.]

20. [No hint.]

24. a. Suppose that $x < y$, and then use the fact that $<$ is irreflexive and transitive to prove by contradiction that $y \not\leq x$.

26. a. Review the definitions of strict order and partial order to see what properties are given and what ones need to be demonstrated.

27. b. Use proof by contradiction here to show a finite set must have a minimal element. A similar proof shows it must have a maximal element.

30. a. Start by supposing $(x_1, x_2) \leq (y_1, y_2)$ and $(y_1, y_2) \leq (x_1, x_2)$. You'll need to use the facts that $<$ is asymmetric and $\leq$ is antisymmetric, but the rest is *Sentential Logic* simplification.

33. a. As you might expect, you can just replace the $\leq$ in the definition with $<$. Arguing that this is correct involves a chain of biconditionals, using logical simplification.

35. b. Use the definition developed in the text for comparing strings.

38. Example 7 is similar to this problem.

7.2 Theory of Lattices

In Section 7.1 we began our study of partially ordered sets (posets). In this section we will explore a particular type of poset known as a *lattice*. Lattices have a number of applications, and they provide one way for us to introduce and become familiar with *Boolean Algebra*, a field of prime importance to computer science.

In order to concentrate on lattices that are most appropriate for this final focus, we will need to consider a fair number of new properties of relations. We will present these not so much for their own sake as for arriving at the notion of a *Boolean lattice*. We will do this *en masse*, so as not to prolong the process. Perhaps the best way to get familiar with the new terms this generates is to look up their definitions as needed when you are working through material that involves them.

Definition of a Lattice

To define a lattice, we must first introduce the dual notions of a meet and a join for pairs of elements in a poset. Meets and joins are unique when they exist (see Exercise 1), so we will assume this in our definition.

DEFINITION* 7.2 - 1: *Meet, Join

Let $\langle \mathcal{A}, \leq \rangle$ be a poset and let x and y be any pair of elements of $\mathcal{A}$.

a) *The* ***meet*** *of x and y, denoted by $x \wedge y$, is the maximum of all lower bounds for x and y; i.e., $x \wedge y = \max\{w \in \mathbb{A} : w \leq x, w \leq y\}$, the greatest lower bound for x and y.*

b) *The* ***join*** *of x and y, denoted by $x \vee y$, is the minimum of all upper bounds for x and y; i.e., $x \vee y = \min\{z \in \mathbb{A} : x \leq z, y \leq z\}$, the least upper bound for x and y.*

DEFINITION* 7.2 - 2: *Lattice

A poset $\langle \mathcal{A}, \leq \rangle$ is a ***lattice*** *iff every pair of elements in $\mathcal{A}$ have both a meet and a join.*

Since each pair of elements in a lattice has something above and below it, no lattice (besides the one-point lattice) can have isolated points. The Hasse diagram of a lattice is thus a connected (single component) graph: there is a path of edges linking any two points in the diagram (see Exercise 9).

You'll notice that we are reusing the logical symbols for *and* and *or* to stand for meets and joins in a lattice. We'll see in Section 7.3 that there is a good reason for doing so; here, however, we will just take them as abstract symbols standing for meet and join, defined in terms of the poset's partial order $\leq$. In some other contexts, the symbols used for meet and join are those of multiplication and addition, $\cdot$ and $+$, only once again interpreted abstractly instead of in their ordinary numerical meaning (again, see Section 7.3). All of this notational repetition and flexibility makes for some understandable confusion when one is starting to learn about lattices. Unfortunately, it can't really be avoided, because different books and different areas of thought use different symbols to denote the same concept and the same symbols to denote several concepts. (And you thought mathematics was the epitome of precise logical practice and uniform notation!)

✠ **EXAMPLE 7.2 - 1**

Show that the following posets are lattices, and interpret their meets and joins:

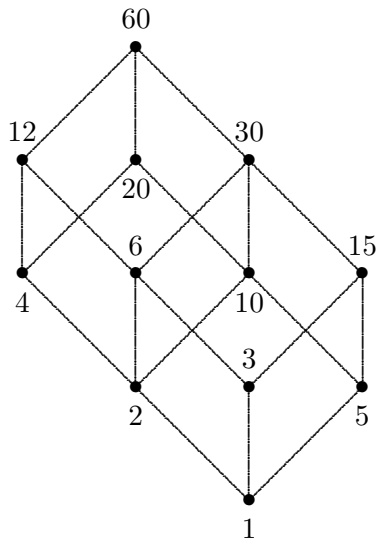
a) The poset of the divisors of 60, ordered by divisibility (see Example 7.1-6a).

b) The poset of the subsets of $\{0, 1, 2\}$, ordered by the subset relation (see Example 7.1-4b).

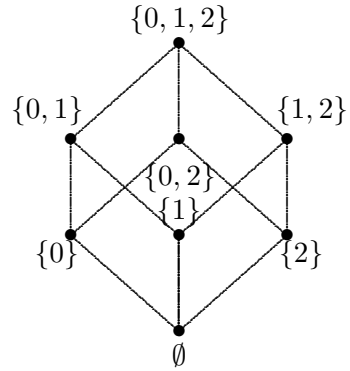
Solution

These are graphed by the following Hasse diagrams.

a)



b)



a) The poset consisting of all the divisors of 60 is a lattice; every pair of elements has both a meet and a join.
The meet (greatest lower bound) of two divisors is their *greatest common divisor*:
for example, $6 \wedge 20 = \gcd(6, 20) = 2$.
The join (least upper bound) of two divisors is their *least common multiple*:
for example, $6 \vee 20 = \text{lcm}(6, 20) = 60$.

b) The poset consisting of all the subsets of $\{0, 1, 2\}$ is also a lattice.
The meet (greatest lower bound) of two subsets is the *intersection* of the two subsets:
for example, $\{0, 1\} \wedge \{1, 2\} = \{0, 1\} \cap \{1, 2\} = \{1\}$.
The join (least upper bound) is the *union* of the two subsets:
for example, $\{0, 1\} \vee \{1, 2\} = \{0, 1\} \cup \{1, 2\} = \{0, 1, 2\}$.

Both of these examples generalize. Given any positive integer n, the set of all its divisors forms a lattice in which the meet of two divisors is their greatest common divisor and their join is the least common multiple of the divisors (see Exercise 4a).

The power set $\mathcal{P}(S)$ of any set S also forms a lattice (see Exercise 7a). Given two subsets R and T of S, $R \wedge T = R \cap T$ and $R \vee T = R \cup T$. This example helps explain the terminology chosen for meet and join: here the meet is the set in which the two given sets meet, and the join is the set that joins the two given sets together. [Note in passing that it now makes good sense to write $R \wedge T$, whereas it didn't before when $\wedge$ was taken to mean *and*, for then it was supposed to operate on sentences, not sets. Note also that in statements involving the meet and join of objects it would be quite confusing to combine sentential clauses using the logical symbolism for *and* and *or*. Rather a symbolic mess, which we will avoid by using the words *and* and *or*, where needed. It really would be good to standardize mathematical notation in a less confusing way, wouldn't it?!)

Basic Properties of Meet and Join

Meet and join satisfy a number of properties as the greatest lower bound and least upper bound of pairs of elements in a poset. These are summarized in the following two propositions. The first proposition contains some basic relational properties that are helpful for establishing the more important operational properties of the second proposition.

PROPOSITION* 7.2-1: *Basic Order Properties of Meet and Join

Let $\langle \mathcal{A}, \leq \rangle$ be a lattice. Then

a) $x \wedge y \leq \{x, y\} \leq x \vee y$.

b) $x \leq y$ *iff* $x \wedge y = x$.

c) $x \leq y$ *iff* $x \vee y = y$.

d) *If* $x \leq y$, *then* $x \wedge z \leq y \wedge z$ *and* $x \vee z \leq y \vee z$.

e) *If* $x \leq y$ *and* $z \leq w$, *then* $x \wedge z \leq y \wedge w$ *and* $x \vee z \leq y \vee w$.

***Proof*:**

a) This holds because the meet $x \wedge y$ is a lower bound for x and y, while the join $x \vee y$ is an upper bound. In fact, we know more than this: no other element of the lattice lies between these extremal elements and the pair of elements.

b) This follows from the meet being the greatest lower bound. See Exercise 10a.

c) This follows from the join being the least upper bound. See Exercise 10b.

d) This monotonicity result follows from the definition of meet and join. See Exercise 10c.

e) This is a generalized monotonicity result: the meet or join of the two smaller elements is less than or equal to the meet or join of the two larger elements. This can be proved using part *d*. See Exercise 10d. ■

PROPOSITION* 7.2-2: *Basic Operational Properties of Meet and Join

Let $\mathcal{A}$ be a lattice with order relation $\leq$. Then the following properties hold:

a) ***Commutativity***: $x \wedge y = y \wedge x$; $\quad x \vee y = y \vee x$.

b) ***Associativity***: $(x \wedge y) \wedge z = x \wedge (y \wedge z)$; $\quad (x \vee y) \vee z = x \vee (y \vee z)$.

c) ***Idempotence***: $x \wedge x = x$; $\quad x \vee x = x$.

d) ***Absorption***: $x \wedge (x \vee y) = x$; $\quad x \vee (x \wedge y) = x$.

***Proof*:**

a) These two results are immediate: the greatest lower bound and the least upper bound of a pair of elements only depend on which x and y are involved, not on the order in which they are taken up.

b) We will prove the first identity and leave the second one as an exercise (Exercise 11b). [We can show that two elements of $\mathcal{A}$ are equal via the antisymmetric property of $\leq$: we show each element is $\leq$ the other. (This exactly parallels the way we show two sets are equal in the poset of sets.)]
$(x \wedge y) \wedge z \leq x \wedge y \leq x$ by *Proposition* 1a (twice).
So $(x \wedge y) \wedge z \leq x$ due to transitivity of $\leq$.
Also by *Proposition* 1a, $x \wedge y \leq y$.
By *Proposition* 1d, $(x \wedge y) \wedge z \leq y \wedge z$.
Thus $(x \wedge y) \wedge z$ is below both x and $y \wedge z$. Since the meet of these two elements is their *greatest* lower bound, we must have $(x \wedge y) \wedge z \leq x \wedge (y \wedge z)$.
A similar argument establishes the opposite inequality (see Exercise 11a).

c) See Exercise 11c.

d) See Exercise 11d. ■

Distributive Lattices

If you were paying attention to the sort of properties that were being proved in the last proposition, you may have been wondering what happened to the distributive property. Why wasn't that included? Well, there's a good reason. Not all lattices are *distributive*. Only the distributive ones are, though all lattices satisfy a semi-distributive law (see Exercise 24).

DEFINITION* 7.2 - 3: *Distributive Lattice

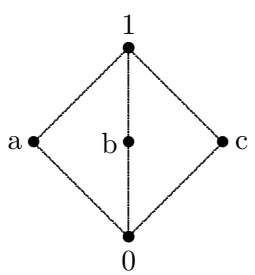
A lattice $\mathcal{A}$ is ***distributive*** *iff the following* ***distributive laws*** *hold for any x, y, and z in $\mathcal{A}$:*
$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$; $\quad x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$.

We already know that meet and join are commutative, so if a lattice satisfies the above distributive laws, it automatically satisfies the other distributive laws as well (see Exercise 25): $(x \wedge y) \vee z = (x \vee z) \wedge (y \vee z)$ and $(x \vee y) \wedge z = (x \wedge z) \vee (y \wedge z)$.

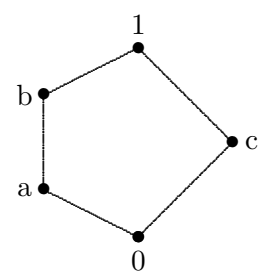
✠ **EXAMPLE 7.2 - 2**

Show that the following simple but significant lattices are not distributive. (In fact, it has been proved that every non-distributive lattice contains a copy of one of these lattices.)

a)



b)



Solution

a) To see that the diamond lattice is not distributive, use the middle elements of the lattice: $a \wedge (b \vee c) = a \wedge 1 = a$, but $(a \wedge b) \vee (a \wedge c) = 0 \vee 0 = 0$, and $a \neq 0$.
Similarly, the other distributive law fails for these three elements (see Exercise 22a).

b) The pentagon lattice is also not distributive. See Exercise 22b.

Working by hand, it is usually easier to show that a lattice is *not* distributive than that it is. For example, if a lattice has 5 elements, like the ones in Example 2, you would need to check the validity of $5 \cdot 15 = 75$ essentially different equations for each distributive-law equation in *Definition* 3 (see Exercise 23; order is irrelevant for the final pair, due to commutativity). Presumably, if a lattice is not distributive, you will stumble upon a counterexample before reaching the final equation that needs checking. Programming a computer to do the work, however, levels the playing field for checking identities to see if a finite lattice is distributive.

✠ **EXAMPLE 7.2 - 3**

Explain why the power-set lattice $\mathcal{P}(U)$ is a distributive lattice for any set U.

Solution

Since meet is intersection and join is union, and since each set-theoretic operation distributes over the other one (see *Proposition* 4.1 - 8), $\mathcal{P}(U)$ is a distributive lattice.

Bounded and Complemented Lattices

A second omission from *Proposition* 2 that you might have noticed is the lack of any *De Morgan's Law* for meet and join. We had such laws for both *Sentential Logic* and *Set Theory*; might a version hold here as well?

At this point, we don't even have the equivalent of negation or set complementation for lattices, so we can't say one way or the other. However, we know that all power sets form lattices under the subset relation, so let's look at them to see if we can generalize the set-theoretic concept of complementation to arbitrary lattices. Once that is done, we can decide whether *De Morgan's Laws* hold in general.

Let U be any set and suppose complements are taken with respect to U. The complement of S is defined by $\overline{S} = U - S$. In order to identify an element's complement in a lattice, we will need to recast this in terms of intersection and union, *Set Theory's* meet and join; then we may be able to formulate it as a general lattice concept. Doing this, we have the following characterization of set complements (see Exercise 13): *$\overline{S}$ is that set such that $\overline{S} \cap S = \emptyset$ and $\overline{S} \cup S = U$.* This gives us a further wrinkle to consider: what are the counterparts to $\emptyset$ and U in a lattice? In the power-set lattice, they're the smallest and largest sets, the minimum and maximum points of the lattice. So it seems we may first need to be in a *Bounded Lattice* before we can define complements.

DEFINITION* 7.2 - 4: *Bounded Lattices

A lattice $\langle \mathcal{A}, \leq \rangle$ is ***bounded*** *iff it has a minimum element and a maximum element. These are denoted by* 0 *and* 1 *respectively.*

The extreme elements of bounded lattices interact with other elements of the lattice in the obvious ways, captured by the next proposition.

PROPOSITION* 7.2 - 3: *Extreme Elements in a Bounded Lattice

Suppose $\langle \mathcal{A}, \leq \rangle$ is a bounded lattice having minimum 0 *and maximum* 1*, and let x be any element in $\mathcal{A}$. Then*

a) $0 \vee x = x = x \vee 0; \qquad 1 \wedge x = x = x \wedge 1$

b) $0 \wedge x = 0 = x \wedge 0; \qquad 1 \vee x = 1 = x \vee 1$

***Proof*:**

These all hold because $0 \leq x \leq 1$; apply *Proposition* 1b (see Exercise 16). ■

DEFINITION* 7.2 - 5: *Complements in a Bounded Lattice

Suppose $\langle \mathcal{A}, \leq \rangle$ is a bounded lattice with minimum 0 *and maximum* 1*. A* ***complement of an element x*** *is an element z such that $x \wedge z = 0$ and $x \vee z = 1$.*

Note that the notion of a complement only makes sense in a bounded lattice. But even in bounded lattices, complements need not exist. Totally ordered posets are all lattices – in fact, distributive lattices (see Exercise 19) – but those that are bounded are rarely complemented (see Exercises 17 – 18). Thus we need still another category of lattice for taking complements.

DEFINITION* 7.2 - 6: *Complemented Lattices

A bounded lattice $\langle \mathcal{A}, \leq \rangle$ is ***complemented*** *iff every element has a complement.*

Complemented lattices are, by definition, lattices with complements for all their elements; but the existence of complements still does not guarantee uniqueness (see Exercises 14 – 15). Nevertheless, if the lattice is distributive as well, this leeway disappears, as the next proposition demonstrates. The corollary that follows is immediate.

PROPOSITION* 7.2 - 4: *Uniqueness of Complements in Distributive Lattices

If $\langle \mathcal{A}, \leq \rangle$ is a bounded distributive lattice with minimum 0 *and maximum* 1*, then complements are unique, provided they exist.*

***Proof*:**

Suppose both $\overline{x}$ and z denote complements of x in $\mathcal{A}$. We will show that $\overline{x} = z$.

The following argument is not quite as easy as it looks. To appreciate it, try to prove $\overline{x} = z$ on your own before continuing to the next page.

By the definition of being a complement, $\overline{x} \wedge x = 0 = x \wedge z$ and $\overline{x} \vee x = 1 = x \vee z$.
Thus, $\overline{x} = \overline{x} \wedge 1$

$$\begin{aligned}
&= \overline{x} \wedge (x \vee z) \\
&= (\overline{x} \wedge x) \vee (\overline{x} \wedge z) \\
&= 0 \vee (\overline{x} \wedge z) \\
&= (x \wedge z) \vee (\overline{x} \wedge z) \\
&= (x \vee \overline{x}) \wedge z \\
&= 1 \wedge z \\
&= z \qquad \blacksquare
\end{aligned}$$

COROLLARY* 1: *Complemented Distributive Lattices Have Unique Complements
Every element in a complemented distributive lattice has a unique complement. The complement of x will be denoted by $\overline{x}$.

Our notation for complements in the abstract is the same as what we used for sets. Not surprisingly, given our initial motivation for defining complements in lattices, power-set lattices turn out to be good examples of complemented distributive lattices (see Exercise 7).

Boolean Lattices: Definition and Properties

Complemented distributive lattices are an important type of lattice. Rather than call them by this mouthful, they are given a special name: they are called *Boolean lattices*.

DEFINITION* 7.2-7: *Boolean Lattices
A lattice $\langle \mathcal{A}, \leq, \overline{}, 0, 1 \rangle$ is a ***Boolean lattice*** *iff it is a complemented distributive lattice.*

Suppose now that $\langle \mathcal{A}, \leq, \overline{}, 0, 1 \rangle$ is a Boolean lattice under the order relation $\leq$, with minimum 0 and maximum 1. Then it must finally be a structure satisfying *De Morgan's Laws*.

PROPOSITION* 7.2-5: *De Morgan's Laws
If $\langle \mathcal{A}, \leq, \overline{}, 0, 1 \rangle$ is a Boolean lattice, then
a) $\overline{x \wedge y} = \overline{x} \vee \overline{y}$
b) $\overline{x \vee y} = \overline{x} \wedge \overline{y}$

***Proof*:**
a) Let x and y be any elements of $\mathcal{A}$. We will show that $\overline{x} \vee \overline{y}$ acts like the complement of $x \wedge y$; uniqueness of complements then forces the equality $\overline{x \wedge y} = \overline{x} \vee \overline{y}$.

$$\begin{aligned}
(x \wedge y) \wedge (\overline{x} \vee \overline{y}) &= (x \wedge y \wedge \overline{x}) \vee (x \wedge y \wedge \overline{y}) && \text{Distrib, Assoc} \\
&= 0 \vee 0 && \text{Comm, Compl} \\
&= 0. && \text{Extrm Elts} \\
(x \wedge y) \vee (\overline{x} \vee \overline{y}) &= ((x \wedge y) \vee \overline{x}) \vee \overline{y} && \text{Assoc} \\
&= ((x \vee \overline{x}) \wedge (y \vee \overline{x})) \vee \overline{y} && \text{Distrib} \\
&= (1 \wedge (y \vee \overline{x})) \vee \overline{y} && \text{Compl} \\
&= (y \vee \overline{x}) \vee \overline{y} && \text{Extrm Elts} \\
&= \overline{x} \vee (y \vee \overline{y}) && \text{Comm, Assoc} \\
&= \overline{x} \vee 1 && \text{Compl} \\
&= 1. && \text{Extrm Elts}
\end{aligned}$$

b) See Exercise 30. $\blacksquare$

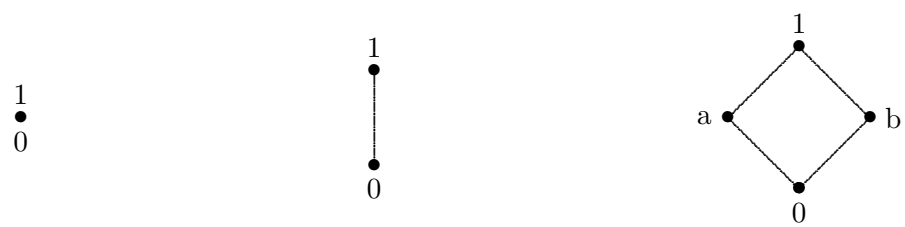
We now know quite a few properties about Boolean lattices. But what do Boolean lattices really look like? Can we say anything definitive about their size? What shape Hasse diagrams do they have? Do the properties held by Boolean lattices constrain their structure in certain predictable ways? We'll look at a few examples to initiate this exploration and then continue it in the next section.

✠ **EXAMPLE 7.2 - 4**

Investigate all Boolean lattices with cardinalities 1 – 5.

Solution

- There is only one lattice structure of size 1: a single point. This forms a Boolean lattice, but not a very interesting one.
- There is also only one lattice structure of size 2: two points in a line. This lattice is also a Boolean lattice, but as a totally ordered set it, too, is not very interesting.
- There are no lattice structures of size 3 except the totally ordered one of three points in a line. However, this structure has no complement for the middle point (why not?), so it cannot be a Boolean lattice; in fact, no totally ordered set with more than two elements is a Boolean lattice (see Exercise 18).
- There are two different lattice structures on four elements. There must be a top point and a bottom point; the other two must lie in the middle. These can either be side by side or in line. This gives two lattice structures : a four-point line and a diamond. The first is not a Boolean lattice, as we just noted, but the latter is.
- Finally, given five points, one must be the top and another the bottom. The other three must lie between these two in some order. No such lattice structure turns out to be a Boolean lattice (see Exercise 26).
- Thus, the only order structures of cardinality 5 or less that are Boolean lattices are ones with 1, 2, or 4 elements. Hmm. That's an interesting sequence! (See also Exercises 26 – 29).
- The Boolean lattice structures found so far are pictured below.



EXERCISE SET 7.2

1. Let $\langle \mathcal{A}, \leq \rangle$ be a poset, x and y elements of $\mathcal{A}$. Show that $x \wedge y$ and $x \vee y$ are unique, if they exist.

Problems 2 - 4: Divisor Lattices

The following problems have to do with lattices in which the partial order is the divisibility relation.

*2. Let $\langle \mathcal{D}_{12}, \mid \rangle$ denote the poset of all divisors of 12.

*a. Show that $\mathcal{D}_{12}$ is a lattice by drawing out the Hasse diagram for the poset and then verifying that each pair of divisors has both a meet and a join. How do meet and join relate to the numbers in terms of divisibility?

*b. Is $\mathcal{D}_{12}$ a complemented lattice? Explain.

EC c. Is $\mathcal{D}_{12}$ a distributive lattice? To check whether $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$, how many different equations must be checked? Explain.

d. If the bottom point 1 and the top point 12 are deleted from $\mathcal{D}_{12}$, is the result still a lattice? Explain.

e. Is $\mathcal{D}_{12}$ a Boolean lattice? Explain.

3. Let $\langle \mathcal{D}_{30}, \mid \rangle$ denote the poset of all divisors of 30.
   a. Show that $\mathcal{D}_{30}$ is a lattice. Explain.
   b. Is $\mathcal{D}_{30}$ a complemented lattice? Explain.
   c. Is $\mathcal{D}_{30}$ a distributive lattice? Explain.
   d. Is $\mathcal{D}_{30}$ a Boolean lattice? Explain.

4. Let $\langle \mathcal{D}_n, \mid \rangle$ denote the poset of all divisors of n, where n is a positive integer.
   a. Prove that $\mathcal{D}_n$ is a lattice. Carefully explain why $x \wedge y = \gcd(x, y)$ and $x \vee y = \operatorname{lcm}(x, y)$.
   b. When will $\mathcal{D}_n$ be a complemented lattice? Explain.
   EC c. When will $\mathcal{D}_n$ be a distributive lattice? Explain.
   Hint: to check $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$, express x, y, and z via their prime factorizations and relate meet and join to how many factors of each prime (possibly 0) must be used. You may assume that $\langle \mathbb{N}, \leq \rangle$ is a distributive lattice (see Exercise 19).
   d. For which n will $\mathcal{D}_n$ be a Boolean lattice?

Problems 5 - 6: True or False

Are the following statements true or false? Explain your answer.

5. Complements are unique in a complemented lattice.

6. If a Boolean lattice has more than two elements, then it is not totally ordered.

*7. *Power-Set Lattices*
   *a. Prove that all power-set posets $\mathcal{P}(S)$ are lattices. Carefully explain why the meet and join of this poset are set intersection and set union, as claimed in the text.
   b. Given part *a*, briefly explain why all $\mathcal{P}(S)$ are complemented and distributive lattices (and hence Boolean lattices).
   c. If the bottom point $\emptyset$ and the top point S are deleted from the power-set lattice $\mathcal{P}(S)$, is the resulting set a lattice?
   d. If a single subset of S is deleted from its power-set lattice $\mathcal{P}(S)$, will the result still be a lattice? Explain.

Problems 8 - 9: Lattice Diagrams

The following problems explore Hasse diagrams for lattices.

*8. *Small Lattices*
   Draw Hasse diagrams for all possible lattices having the following number of points (see also Example 4). Explain why you have considered all possibilities.
   a. Lattices with 1 element.
   b. Lattices with 2 elements.
   c. Lattices with 3 elements.
   *d. Lattices with 4 elements.
   e. Lattices with 5 elements.
   f. Which of the lattices in parts *a* - *e* are complemented lattices? Explain.
   g. Which of the lattices in parts *a* - *e* are distributive lattices? Explain.

*9. *Lattice Paths*
   *a. Prove or disprove: *A Hasse diagram for a lattice is one in which given any two elements a and b it is possible to pass from a to b by first passing through a sequence of connected points in the upward direction and then passing though a sequence of connected points in the downward direction.*
   b. Is your answer to part *a* different if the terms *upward* and *downward* are interchanged? Explain.

Problems 10 - 12: Basic Properties of Lattices

Suppose $\langle \mathcal{A}, \leq \rangle$ is an arbitrary lattice. Prove the following. Note: the only things you may assume about $\leq$ in this context are properties that belong to all such relations.

*10. *Proposition* 1: *Order Properties*
   - *a. Prove *Proposition* 1b: $x \leq y$ iff $x \wedge y = x$.
   - b. Prove *Proposition* 1c: $x \leq y$ iff $x \vee y = y$.
   - *c. Prove *Proposition* 1d: If $x \leq y$, then $x \wedge z \leq y \wedge z$ and $x \vee z \leq y \vee z$.
   - d. Prove *Proposition* 1e: If $x \leq y$ and $z \leq w$, then $x \wedge z \leq y \wedge w$ and $x \vee z \leq y \vee w$.

11. *Proposition* 2: *Operational Properties*
   - a. Complete the proof of the first part of *Proposition* 2b: $x \wedge (y \wedge z) \leq (x \wedge y) \wedge z$.
   - b. Prove the second part of *Proposition* 2b: $(x \vee y) \vee z = x \vee (y \vee z)$.
   - c. Prove *Proposition* 2c: $x \wedge x = x = x \vee x$.
   - d. Prove *Proposition* 2d: $x \wedge (x \vee y) = x = x \vee (x \wedge y)$

12. *Dual Lattice*
   Let $\langle \mathcal{A}, \leq \rangle$ be a lattice, and let $\geq$ denote the converse order relation $\widehat{\leq}$.
   - a. If $w = x \wedge y$ in $\mathcal{A}$ under the order relation $\leq$, how is w related to x and y under the relation $\geq$? Explain this carefully.
   - b. If $z = x \vee y$ in $\mathcal{A}$ under the order relation $\leq$, how is z related to x and y under the relation $\geq$? Explain this carefully.
   - c. Explain why $\langle \mathcal{A}, \geq \rangle$ is a lattice (the dual lattice).

13. Show for any subset S inside a set U, $\overline{S} = C$ iff C is that set such that $C \cap S = \emptyset$ and $C \cup S = U$.

Problems 14 - 16: Bounded and Complemented Lattices

Suppose $\langle \mathcal{A}, \leq, 0, 1 \rangle$ is an arbitrary bounded lattice with least element 0 and greatest element 1. Prove the following. Note: the only things you may assume about $\leq$, 0, and 1 in this context are properties that belong to all such relations and extremes.

14. *Complements for Example* 2
   - a. Find all complements, when they exist, for the elements in the lattice of Example 2a.
   - b. Find all complements, when they exist, for the elements in the lattice of Example 2b.

15. *Complements for Exercise* 21.
   - a. Find all complements, when they exist, for the elements in the lattice of Exercise 21a below. Is the lattice there a complemented lattice?
   - b. Find all complements, when they exist, for the elements in the lattice of Exercise 21b below. Is the lattice there a complemented lattice?

16. *Proposition* 3: *Extreme Element Laws*
   - a. Prove *Proposition* 3a: $0 \vee x = x = x \vee 0\,; \quad 1 \wedge x = x = x \wedge 1$.
   - b. Prove *Proposition* 3b: $0 \wedge x = 0 = x \wedge 0\,; \quad 1 \vee x = 1 = x \vee 1$.

Problems 17 - 20: Totally Ordered Sets and Lattices

Let $\langle \mathcal{A}, \leq \rangle$ be a totally ordered set.

*17. Prove the following.
   - a. $\mathcal{A}$ is a lattice.
   - b. What is the meet and join of any two elements a and b in $\mathcal{A}$? Explain.

*18. Prove that if $\mathcal{A}$ has more than two elements, then it is not a complemented lattice, even if it has a minimum and a maximum.

*19. Prove that the distributive laws hold when the three elements x, y, and z chosen all lie along a common line within a lattice (consider cases). Hence, totally ordered sets are distributive lattices.

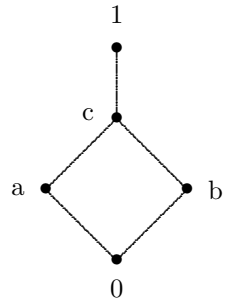
*20. Explain, based on Problem 19, why in determining whether a lattice is distributive, it suffices to check distributive law equations for distinct elements not all in a line with one another.

Problems 21 - 25: Distributive Lattices

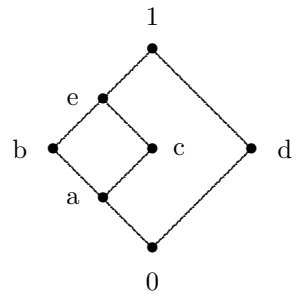
The following problems relate to distributive lattices.

*21. Are the following lattices distributive or not? Explain.

EC a.



*b.



22. *Distributive Laws for Example 2*
    a. Show that the second distributive law $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ fails for the lattice of Example 2a.
    b. Show that the lattice of Example 2b is not distributive.

23. Explain why the calculation given in the text right after Example 2 for the number of distributive law equations that need to be checked for a lattice with 5 elements is correct.

24. Show that the following semi-distributive laws hold in any lattice.
    a. $x \vee (y \wedge z) \leq (x \vee y) \wedge (x \vee z)$
    b. $x \wedge (y \vee z) \geq (x \wedge y) \vee (x \wedge z)$

25. Show that the following distributive laws hold in a distributive lattice.
    a. $(x \wedge y) \vee z = (x \vee z) \wedge (y \vee z)$
    b. $(x \vee y) \wedge z = (x \wedge z) \vee (y \wedge z)$

Problems 26 - 29: Finite Boolean Lattices

The following problems explore finite Boolean lattices for small lattices.

EC 26. Show that there are no Boolean lattices with cardinality 5.

27. Show that there are no Boolean lattices with cardinality 6.

28. Show that there are no Boolean lattices with cardinality 7.

EC 29. Show that there is a Boolean lattice with cardinality 8.

Problems 30 - 32: Boolean Lattices

Suppose that $\langle \mathcal{A}, \leq, \overline{}, 0, 1 \rangle$ is an arbitrary Boolean lattice. Prove the following. Note: the only things you may assume about $\leq$, 0, 1, and $\overline{}$ in this context are properties that belong to all such relations, extremes, and complements.

30. *Proposition* 5: *De Morgan's Laws*
    Prove *Proposition* 5b: if then $\overline{x \vee y} = \overline{x} \wedge \overline{y}$.

31. *Equivalent Formulations for Boolean Partial Order*
    Prove that each of the following is logically equivalent to $x \leq y$ in a Boolean lattice.

EC a. $x \wedge \overline{y} = 0$
    b. $x \vee \overline{y} = 1$

*32. *Redundancy Laws*
    Prove the following redundancy laws.
    *a. $x \wedge (\overline{x} \vee y) = x \wedge y$
    b. $x \vee (\overline{x} \wedge y) = x \vee y$
    c. $x \wedge y \leq (x \wedge z) \vee (y \wedge z)$

HINTS TO STARRED EXERCISES 7.2

2. a. Example 1a is similar to this problem.
   b. Review the definition, and then check whether elements here have complements.

7. a. The meet here for two elements X and Y would be that set M such that M is a subset of both X and Y, and if S is any set that is a subset of both X and Y, then $S \subseteq M$. Show that $M = X \cap Y$ satisfies this condition.

8. d. Example 4 gives the result; argue why each structure is a lattice.

9. a. Remember that in a lattice any pair of elements has both a meet and a join.

10. a. Use *Proposition* 1a along with the definition of meet.
    c. One way to prove this is by showing $x \wedge z \leq y$ and $x \wedge z \leq z$ and then using the definition of meet for $y \wedge z$. Alternatively, use the distributive property on $x \vee (y \wedge z)$ along with *Proposition* 1b and 1c.

17. Since $\mathcal{A}$ is totally ordered, either $a \leq b$ or $b \leq a$.

18. Suppose $\mathcal{A}$ has three or more elements, then use Exercise 17 to show an element has no complement.

19. Consider six cases, depending on the ordering of x, y, and z. Grunt work.

20. Besides using Exercise 19, as suggested, consider the cases where two elements are the same and the third is not in line with it.

21. b. Use Exercise 20 to shorten your work.

32. a. Expand the left side using the distributive property for meets and joins.

7.3 From Boolean Lattices to Boolean Algebra

Relations were first introduced in Section 6.3. In Section 7.1 we noted that relations provide a second focus for algebra, which is mainly preoccupied with calculations. The last two sections have now developed some of this lesser-known side of algebra. However, in this section, we move back from the relational side to the operational side. Boolean lattices provide us with a good context in which to introduce *Boolean Algebra*, which has a computational cast to it.

We'll begin, though, with some additional exploration of what a Boolean lattice can look like. Next we'll show how to convert a Boolean lattice into an operational algebraic structure called a *Boolean algebra*. This will then give us an opening for presenting the theory of *Boolean Algebra* axiomatically. Finally, we'll see how we can recover a Boolean lattice structure from a Boolean algebra. The material in this section will provide a solid foundation for our work with Boolean functions in later sections.

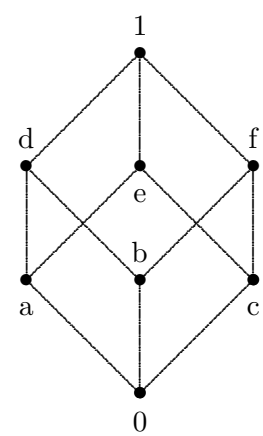
Finite Boolean Lattices

First recall that a Boolean lattice $\langle \mathcal{A}, \leq, \overline{}, 0, 1 \rangle$ was defined as a complemented distributive lattice. As a lattice, it has a meet $\wedge$ and a join $\vee$, which satisfy a number of basic relational and operational properties (*Propositions* 7.2-1 and 7.2-2). Because $\mathcal{A}$ has a minimum 0 and a maximum 1, these satisfy some laws for extreme elements (*Proposition* 7.2-3); and because it satisfies the distributive laws and has complements, it satisfies some additional laws for them, such as *De Morgan Laws* (*Proposition* 7.1-5). These laws put a fair bit of constraint on what such a structure can be. We'll look at a Boolean lattice of size 8 to get some sense of what finite Boolean lattices are.

✠ **EXAMPLE 7.3 - 1**

Investigate the structure of the Boolean lattice $\mathcal{A}$ of order 8 given below. This is essentially the power set of a three-element set, differently labeled.

Solution



Since $\mathcal{A}$ is a finite Boolean lattice, it can be broken up into n distinct levels. Call the bottom level where the minimum 0 is located Level 0, the next level up (those elements right above 0) Level 1, and so on. Here $\mathcal{A}$ has 3 levels above 0.

– Level 1 contains a layer of elements we will call the *atoms* of the lattice. The atoms in this case are a, b, and c. These atoms generate the lattice above it via the join operation $\vee$.

– Level 2 of the lattice $\mathcal{A}$ consists of elements d, e, and f. Note that these ${}_3C_2 = 3$ elements are all possible joins of the atoms: they are $a \vee b$, $a \vee c$, and $b \vee c$.

– Finally, the top level, Level 3, consists only of the maximum element 1. It lies directly above all the elements on Level 2. Just as importantly, it can be thought of as the join of the atoms taken altogether, as $a \vee b \vee c$. There is only ${}_3C_3 = 1$ element here.

– An insightful way to denote the elements here uses triplet binary notation for indicating the eight elements, thinking of them, say, as representing the elements of the power set of $S = \{x, y, z\}$. The bottom element would be denoted by 000 (no elements present); the next level by 100 (only x present), 010 (only y present), and 001 (only z present); the level above this by 110 (x and y present), and so on. The *product order* on the poset B^3 (see Exercise 34), where $B = \{0, 1\}$, ordered in the usual way ($0 \leq 1$), turns out to yield the partial order stipulated here: $(x_1, x_2, x_3) \leq (y_1, y_2, y_3)$ iff $x_1 \leq y_1$ and $x_2 \leq y_2$ and $x_3 \leq y_3$.

The example just given generalizes to any finite Boolean lattice. This turns out to severely restrict the structure of finite Boolean lattices: they are all of cardinality 2^n for some n, and there is only one lattice structure for each power of 2, the power-set lattice. First a definition, then the result.

DEFINITION* 7.3 - 1: *Atom of a Boolean Lattice

*An **atomic element** (**atom**) of a Boolean lattice $\langle \mathcal{A}, \leq, \overline{}, 0, 1 \rangle$ is a non-trivial minimal element of $\mathcal{A} - \{0\}$.*

THEOREM* 7.3 - 1: *Stone Representation Theorem for Finite Boolean Lattices (1936)

Suppose $\langle \mathcal{A}, \leq, \overline{}, 0, 1 \rangle$ is a finite Boolean lattice. Then $|\mathcal{A}| = 2^n$ for some n, and the structure of $\mathcal{A}$ is that of the power-set lattice $\langle \mathcal{P}(S), \subseteq \rangle$ for $S = \{1, 2, \ldots, n\}$.

***Proof*:**

We will outline the main claims of the proof, leaving the technical details as exercises.

- First note that $\mathcal{A}$ does have atomic elements (non-0 minimal elements) when $|\mathcal{A}| > 1$; their existence is guaranteed by the fact that $\mathcal{A}$ is finite (see Exercise 7). So let $a_1, a_2, \ldots, a_n$ denote all the atoms of $\mathcal{A}$.
- Secondly, all atomic elements lie directly above 0 (see Exercise 8).
- Moreover, every element in $\mathcal{A}$ lies above one or more of these atoms (see Exercise 9).

Given these observations, we can now prove two major claims.

1. *Each element can be expressed as the join of all the atoms that lie below it.*
   Suppose element x lies above atoms $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$.
   This claim is proved by showing both $x \leq a_{i_1} \vee a_{i_2} \vee \cdots \vee a_{i_k}$ and $a_{i_1} \vee a_{i_2} \vee \cdots \vee a_{i_k} \leq x$ (see Exercise 11). One direction uses the fact that if $x \wedge \overline{y} = 0$, then $x \leq y$ (see Exercise 10).
2. *Atomic join-combination representations of elements are unique.*
   This asserts that two atomic join-combinations $a_{i_1} \vee a_{i_2} \vee \cdots \vee a_{i_k}$ represent the same element iff they join exactly the same atoms. This can be proved by taking the meet of the various atoms with the given expressions, using the fact that the meet of two distinct atoms must be 0 (see Exercise 12).

By means of these two claims, a one-to-one onto correspondence is established between the elements of $\mathcal{A}$ and all possible join-combinations. Furthermore, join-combinations of the atoms correspond in an obvious way to distinct subsets of $S = \{a_1, a_2, \ldots, a_n\}$. Thus, $\mathcal{A}$ perfectly matches up with $\mathcal{P}(S)$. Since this matching respects the ordering of the two Boolean lattices, we've shown that $\mathcal{A}$ is essentially $\mathcal{P}(S)$. Thus $|\mathcal{A}| = 2^n$ as well. ■

The *Stone Representation Theorem* is quite a pleasant and satisfying surprise. We've been exploring posets and lattices as abstract objects, without specifying what the elements are or what order relation connects them. Yet this theorem says that when we have a finite Boolean lattice, it is merely a full power-set lattice in disguise. That may still seem abstract to you, since it involves the power set, but it is quite specific. We now know exactly what finite Boolean lattices must look like. Studying finite power-set lattices $\mathcal{P}(S)$, therefore, amounts to exploring all finite Boolean lattices. Infinite Boolean lattices need not be full power-set lattices, but they, too, are essentially subset lattices for families of subsets of some set S.

Boolean Lattices Form Boolean Algebras

A Boolean lattice $\langle \mathcal{A}, \leq, \overline{}, 0, 1 \rangle$ has meets, joins, and complements, all defined in terms of the order relation $\leq$ and the extreme elements 0 and 1. Meet and join are really *binary operations* on $\mathcal{A}$, however: given any two elements x and y, $x \wedge y$ and $x \vee y$ are uniquely defined elements of the lattice. Furthermore, complementation is a unary operation on $\mathcal{A}$: given any x

in $\mathcal{A}$, $\overline{x}$ is uniquely defined. So while $\mathcal{A}$ may have begun life as a relational algebraic structure, it gains a second life as an operational algebraic structure.

We have already summarized the basic operational properties of these operations in *Proposition* 7.2-2. Meet and join satisfy *Commutative Laws*, *Associative Laws*, *Idempotent Laws*, *Absorption Laws*, *Distributive Laws*, and more. The extreme elements 0 and 1 satisfy *Extreme Element Laws* (*Identity Laws* plus an *Absorption* and *Annihilation Law*). Complementation interacts with meet and join in a way that satisfies *De Morgan's Laws*.

We have here, then, a theory that might be developed independently of its relational origin. Such a structure is called a *Boolean algebra*, and its theory is called *Boolean Algebra*.* Historically speaking, *Boolean Algebra* arose before Boolean lattices were studied. It has its roots in the work of George Boole, who used algebra as a way of investigating logic. We will explore this connection shortly.

If we are to treat $\langle \mathcal{A}, \wedge, \vee, \overline{}, 0, 1 \rangle$ as an operational algebra, without reference to its relational structure, cut off from its origins, as it were, we will need another way to think about its operations and extreme elements. What are these operations and elements apart from the order relation that we used to define them? Modern *Boolean Algebra* makes no attempt to define these things but takes an abstract approach with respect to their meaning. It treats them simply as uninterpreted binary operations and distinguished elements that satisfy certain formal conditions.

It is possible to assume all the laws already proved for these operations, but this is more than we need, for as we have already seen in Section 7.2, some of these properties can be used to prove others; not all are equally basic. So if we were to stipulate certain properties as axioms, automatically holding for a Boolean algebra, which ones should we take? The following list gives a fairly common axiomatization of *Boolean Algebra*, though it still contains some slight redundancy (see Exercise 27).

Boolean Algebra as an Axiomatic Theory

A Boolean algebra $\langle \mathcal{A}, +, \cdot, \overline{}, 0, 1 \rangle$ is a set $\mathcal{A}$ together with two binary operations $+$ and $\cdot$, a unary operation $\overline{}$, and two distinguished elements 0 and 1 satisfying the following ten axioms:

1. ***Commutative Laws***
   a) $x + y = y + x$ b) $x \cdot y = y \cdot x$
2. ***Associative Laws***
   a) $(x + y) + z = x + (y + z)$ b) $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
3. ***Distributive Laws***
   a) $x \cdot (y + z) = x \cdot y + x \cdot z$ b) $x + (y \cdot z) = (x + y) \cdot (x + z)$
4. ***Identity Laws***
   a) $x + 0 = x = 0 + x$ b) $x \cdot 1 = x = 1 \cdot x$
5. ***Complementation Laws***
   a) $x + \overline{x} = 1 = \overline{x} + x$ b) $x \cdot \overline{x} = 0 = \overline{x} \cdot x$

You'll notice we've changed our binary operation symbols from $\vee$ and $\wedge$ to $+$ and $\cdot$. There is nothing special about these new signs; we could have kept the old ones. But this choice makes a notational break that will help us think about $\mathcal{A}$ apart from any possible lattice structure. It also aligns it more with standard algebraic notation. Of course, this is *not* ordinary addition or multiplication. In fact, number systems like $\mathbb{Z}$ or $\mathbb{R}$ do *not* form a

* Modern algebra uses the term *algebra* in this dual sense, both as the *theory* being developed, and also as the *structure* or algebraic *system of entities* being theorized about. We'll distinguish these with upper and lower case letters, *Algebra* denoting the theory and *algebra* the structured set of elements.

Boolean algebra under their addition and multiplication. We know this because the second distributive law fails there: $1+2\cdot 3 \neq (1+2)\cdot(1+3)$ because $7 \neq 12$. Some laws are the same as for ordinary algebra, but there are very important differences as well. Notwithstanding this divergence, most treatments of *Boolean Algebra* use $+$ and $\cdot$ for the binary operations. The same conventions will be used here as in elementary algebra: $\cdot$ has higher priority than $+$, and xy is typically used in place of $x \cdot y$.

If $+$ and $\cdot$ are not ordinary addition and multiplication, what are they? They are simply binary operations satisfying the five axioms listed above: *that and nothing more.* Similarly, $\overline{}$ is not necessarily set complementation; it is characterized solely in terms of the *Complementation Law* equations. These operations and $\mathcal{A}$ as a whole form an abstract template, as it were, which may be given many different interpretations. If binary and unary operations along with two distinguished elements can be found for a given set $\mathcal{A}$ so that these five pairs of laws hold, then $\mathcal{A}$ forms a Boolean algebra. These laws act like a definition: they help us identify those structures we will be calling Boolean algebras.

Although *Boolean Algebra* is an abstract theory, its Boolean-algebra models are concrete structures of a sort. We already know a number of them from our work in Section 7.2, but there are others as well.

✠ **EXAMPLE 7.3 - 2**

Given a set U, $\mathcal{P}(U)$ forms a Boolean lattice under the partial order relation of $\subseteq$. Interpret $\mathcal{P}(U)$ as a Boolean algebra.

Solution

The operation of $+$ in this case is $\cup$, that of $\cdot$ is $\cap$, and $\overline{}$ is set complementation.
The elements 0 and 1 are $\emptyset$ and U respectively.
All the laws hold for this interpretation, as we already know from our study of *Set Theory.*
Thus, there are infinitely many models of *Boolean Algebra* included in this example.

✠ **EXAMPLE 7.3 - 3**

Let $B = \{0, 1\}$, strictly ordered in the usual way by $0 < 1$.
Define $x + y = \max(x, y)$ and $x \cdot y = \min(x, y)$, and let $\overline{x} = 1 - x$ (ordinary subtraction).
Show that B forms a Boolean algebra under these operations.

Solution

We need to show that all five pairs of axioms hold for this interpretation.
The *Commutative Laws* hold, for example, because $\max(x, y) = \max(y, x)$ and $\min(x, y) = \min(y, x)$. The other four pairs of laws can be shown to hold as well, considering cases where necessary (see Exercises 2 – 6).

Taking an abstract viewpoint can give us a unified theory for widely divergent things, capturing some essential algebraic features of very different structures. For example, the divisors of a number like 60 have nothing to do words in a dictionary, but both sets of things form lattices under their respective partial orders. The same is true here for Boolean algebras. Very different structures can satisfy the same set of Boolean algebra conditions. This abstract approach typifies much of contemporary mathematics and is due to the influence of Hilbert and his school in the early twentieth century. *Boolean Algebra* today has applications in numerous areas of advanced mathematics as well as in some applied areas, such as computer science and engineering. We will look at the latter in Sections 7.4 – 7.6.

Once we know that a structure is a Boolean algebra, satisfying the five basic pairs of axioms, what else do we know about it? What other properties follow necessarily from the laws we've postulated? If we can prove a result based solely on these axioms (without assuming anything additional in particular about the operations or elements involved), then it will hold in general. A conclusion that follows from the axioms will be true of anything that models the axioms, of

all Boolean algebras, regardless of how different the nature of their elements and operations are. It is this sort of unified perspective and economy of thought that attracted mathematicians to develop a theory like *Boolean Algebra* abstractly.

Here are some key results that can be derived from the axioms of *Boolean Algebra*. For all of the following propositions, we assume that $\langle \mathcal{A}, +, \cdot, \overline{}, 0, 1 \rangle$ is a Boolean algebra, satisfying the axioms given above. These propositions have been ordered so that each one can be proved using only the axioms and previous propositions, but other orders are also possible. What you must be careful to do, however, is to not use a result you think is true unless it has already been proved. The proofs of these results are left as exercises to provide you with some practice at proving theorems in an abstract setting.

PROPOSITION* 7.3 - 1: *Uniqueness Laws for Identities and Complements

a) *If* $x + z = x$ *for all* x*, then* $z = 0$.
b) *If* $x \cdot u = x$ *for all* x*, then* $u = 1$.
c) *If* $xz = 0$ *and* $x + z = 1$*, then* $z = \overline{x}$.

***Proof*:**
See Exercise 18. ■

PROPOSITION* 7.3 - 2: *Complements of Elements Laws

a) $\overline{0} = 1$
b) $\overline{1} = 0$
c) $\overline{\overline{x}} = x$

***Proof*:**
See Exercise 19. ■

PROPOSITION* 7.3 - 3: *Idempotence Laws

a) $x \cdot x = x$
b) $x + x = x$

***Proof*:**
See Exercise 20. ■

PROPOSITION* 7.3 - 4: *Annihilation and Absorption Laws

a) $x \cdot 0 = 0$
b) $x + 1 = 1$
c) $x(x + y) = x$
d) $x + xy = x$

***Proof*:**
See Exercise 21. ■

PROPOSITION* 7.3 - 5: *De Morgan's Laws

a) $\overline{xy} = \overline{x} + \overline{y}$
b) $\overline{x + y} = \overline{x}\,\overline{y}$

***Proof*:**
See Exercise 22. ■

PROPOSITION* 7.3 - 6: *Boolean Operation Linkage Laws

The following equations are logically equivalent:

a) $xy = x$
b) $x + y = y$
c) $x\overline{y} = 0$
d) $\overline{x} + y = 1$

***Proof*:**

Remark: Note the difference between this proposition and the others.
What is being asserted here, for example, is a biconditional: $\forall x \forall y (xy = x \leftrightarrow x + y = y)$.
The individual equations of parts a) through d) are *not* being universally asserted as was the case in the other propositions.
See Exercise 23. ■

PROPOSITION* 7.3 - 7: *Redundancy Laws

a) $x(\overline{x} + y) = xy$
b) $x + \overline{x}y = x + y$

***Proof*:**

See Exercise 24. ■

PROPOSITION* 7.3 - 8: *Consensus Laws

a) $xy + \overline{x}z + yz = xy + \overline{x}z$
b) $(x + y)(\overline{x} + z)(y + z) = (x + y)(\overline{x} + z)$

***Proof*:**

See Exercise 25. ■

PROPOSITION* 7.3 - 9: *Cancellation Law

If $xy = xz$ and $x + y = x + z$, then $y = z$.

***Proof*:**

Both of the equations in the antecedent are needed in order for cancellation to be legitimate.
See Exercise 26ab. ■

Boolean Algebra and Logic

Boolean Algebra originated in the mid-nineteenth century with the work of George Boole, for whom the theory is named. It arose in the context of his investigation of logic. Attracted by an acrimonious dispute between De Morgan and a Scottish philosopher over their respective extensions of *Aristotelian Logic*, Boole came up with the novel idea of using algebra to develop a rival methodology for logic (see Section 1.1 for the broader context).

Boole first transformed *Aristotelian Logic*, which was based on categorical assertions; i.e., on statements about how classes were related. He chose 1 to stand for the universe of discourse, 0 for the class with nothing in it, and letters for particular classes or categories of objects. Multiplication resulted in the class of things common to the two classes, addition in their disjoint union.* Using this interpretation, he could assert the identity of two classes by an equation, and he would then calculate with the equations representing the premises of an argument to determine their consequences. Deductive argumentation, he thought, would become nothing more than a species of algebraic computation; logic would be a branch of algebra.

Though there are some important differences, Boole's system has some strong similarities to *Set Theory*. Here is an example of how we can use his system to demonstrate the most important argument form in *Aristotelian Logic*, the one traditionally known as *Barbara*.

* Later logicians changed this to ordinary union, which is based on non-exclusive *or*, to make it a full binary operation.

✠ **EXAMPLE 7.3 - 4**

Using Boole's algebraic approach, show the validity of the following syllogistic argument.

All Xs are Ys
All Ys are Zs
———————
All Xs are Zs

Solution

We can formulate the sentence form "All As are Bs" by means of the equation $A = AB$. To deduce the desired conclusion from its premises, we proceed as follows:

$X = XY$	Prem
$= X(YZ)$	Sub, using the second premise $Y = YZ$
$= (XY)Z$	Assoc Law
$= XZ$	Sub, using $X = XY$ again

Boole's system had a number of complications and idiosyncracies (not our concern here) and was not completely satisfactory, but it inaugurated an era in which mathematics and logic were drawn ever closer together. His work stimulated other mathematicians to take a closer look at logic and its connections to mathematics. And it introduced a new kind of algebra, whose laws were not identically those of ordinary algebra. The power law $X^2 = X$ [idempotence for multiplication] was such a law. Boole knew that this variance would be used by some to argue against his system being a *bona fide* algebraic system; he forestalled them by noting that if one restricted the possible values of the variables to just 0 and 1, which were the solutions to $X^2 = X$, then all the algebraic laws of logic would hold.

Boole also developed a version of his algebraic logic to handle *Sentential Logic* inferences. There variables represented sentences, 1 and 0 stood for true and false respectively, addition and multiplication for disjunction and conjunction, and the complement $\overline{x}$ for a negated x.

✠ **EXAMPLE 7.3 - 5**

Exhibit a derivation of the *Law of Non-Contradiction* from the *Law of Excluded Middle*.

Solution

Here is an abbreviated argument, using results from above; details are left for the exercises (see Exercise 29).

$X + \overline{X} = 1$	LEM
$X \cdot \overline{X} = 0$	Compln, DeM, Compln, Comm

Today we don't refer inside a system of logic to sentences being true or false, and we view *Sentential Logic*'s relation to *Boolean Algebra* a bit differently than Boole did, but the contemporary approach goes back nevertheless to his seminal work. This is spelled out further in the next example.

✠ **EXAMPLE 7.3 - 6**

Interpret *Sentential Logic* in terms of *Boolean Algebra*.

Solution

Interpret the variables of Boolean algebra as representing sentences of SL, 0 as a logical falsehood, and 1 as a logical truth (pick any favorites: say, $P \wedge \neg P$ and $P \vee \neg P$).

Interpret $P \cdot Q$ as the conjunction $P \wedge Q$, $P + Q$ as the disjunction $P \vee Q$, and $\overline{P}$ as the negation $\neg P$.

Then, if we interpret $=$ as $\vDash\!\dashv$,* we can see that all the axioms of *Boolean Algebra* are true. Many of them are just old familiar friends, being the *Replacement Rules* we had available for making SL deductions.
For example, the *Distributive Law* becomes the rule $P \wedge (Q \vee R) \vDash\!\dashv (P \wedge Q) \vee (P \wedge R)$. The *Identity Laws* involving 0 and 1 are a bit unusual from the natural deduction perspective we adopted earlier in our study of SL, but they are also true (see Exercise 31).

There is more that can be done to relate *Boolean Algebra* and logic, but we will reserve this for the next section, when we look at some later developments in the field.

Boolean Algebras Generate Boolean Lattices

We introduced *Boolean Algebra* via Boolean lattices. We will finish our exploration of this topic by returning the favor. Given a Boolean algebra, we'll show how to generate the associated Boolean lattice. Combining these two procedures, we come full circle: if we start with a Boolean lattice, pass to its Boolean algebra, and then generate the associated Boolean lattice, we end up right where we began (see Exercise 37a). The same holds true if we begin and end with a Boolean algebra (see Exercise 37b).

THEOREM* 7.3-2: *Boolean Algebras are Boolean Lattices
Suppose $\langle \mathcal{A}, +, \cdot, ^{-}, 0, 1 \rangle$ *is a Boolean algebra.*
Define a relation $\leq$ *on* $\mathcal{A}$ *by* $x \leq y$ *iff* $x \cdot y = x$.
Then the resulting structure $\langle \mathcal{A}, \leq, ^{-}, 0, 1 \rangle$ *forms a Boolean lattice.*

***Proof*:**
We'll again sketch the main outline of the proof, reserving the details for the exercises.
First recall that in a Boolean algebra, $x \cdot y = x$ iff $x + y = y$ (see *Proposition* 6ab), so we could have defined the order relation equivalently by $x \leq y$ iff $x + y = y$. And we can use this equivalence, wherever useful, to help prove the following claims.

1. $\leq$ *is a partial order on* $\mathcal{A}$.
   Showing that $\leq$ is reflexive, antisymmetric, and transitive is easy, given the definition of $\leq$ and the properties holding for all Boolean algebras (see Exercise 36a).
2. $x \cdot y$ *is the meet* $x \wedge y$, *and* $x + y$ *is the join* $x \vee y$ *relative to the partial order* $\leq$.
   Showing that $x \cdot y$ is a lower bound and $x + y$ is an upper bound for x and y is immediate from the definition and the *Idempotent Laws*; proving that they are the greatest lower bound and the least upper bound is almost as easy (see Exercise 36bc).
3. $\overline{x}$ *is the complement of* x *in the lattice* $\mathcal{A}$.
   This is guaranteed by the *Complementation Laws*.
4. $\mathcal{A}$ *is a distributive lattice.*
   This follows from how $\wedge$ and $\vee$ were defined and from the *Distributive Laws*.

Thus, $\langle \mathcal{A}, \leq, ^{-}, 0, 1 \rangle$ forms a Boolean lattice under the relation $\leq$. ■

* We're cheating here to keep things simple. Strictly speaking, equality is a logical relation that doesn't get (re)interpreted. We do it here in order to avoid first introducing the equivalence relation $\vDash\!\dashv$ on the set of all sentences and then taking the equivalence classes it generates as the elements of the Boolean algebra of logic. If we did that, we could use $=$ in its usual meaning, but the outcome would be more abstract since it would involve a quotient structure with sets as objects.

EXERCISE SET 7.3

Note: *In the problems below, interpret $+$, $\cdot$, and $^{-}$ as abstract operations in a Boolean algebra, not as ordinary addition, multiplication, or complementation, unless otherwise instructed. The only things you can assume about them in this context are properties that hold in Boolean Algebra.*

***Problems 1 - 6: Boolean Algebras of* 0 *and* 1**

Let $\mathcal{B} = \{0, 1\}$ denote the Boolean algebra of Example 3, where $+$ denotes taking the maximum and $\cdot$ denotes taking the minimum.

*1. Write down the addition and multiplication tables for $\mathcal{B}$. Make your tables by putting first elements x down along the side, second elements y across the top, and calculation results $x + y$ and $x \cdot y$ inside their respective tables' cells.

*2. Verify the *Commutative Laws* for addition and multiplication in $\mathcal{B}$.

3. Verify the *Associative Laws* for addition and multiplication in $\mathcal{B}$.

4. Verify the *Distributive Laws* for addition and multiplication in $\mathcal{B}$.

5. Verify the *Identity Laws* for addition and multiplication in $\mathcal{B}$.

6. Verify the *Complementation Laws* for addition and multiplication in $\mathcal{B}$.

Problems 7 - 13: Proving the Stone Representation Theorem

Prove the following results used in the proof of the Stone Representation Theorem for Finite Boolean Lattices. In your deductions, you may use any result that precedes the one you are proving.

7. Prove that every finite Boolean lattice with more than one element has atomic elements.

8. Prove that all atomic elements in a Boolean lattice lie directly above 0.

*9. Explain why every non-zero element in a finite Boolean lattice must lie above (is greater than or equal to) some atom of the lattice.

*10. Prove for a Boolean lattice that if $b \wedge \overline{c} = 0$, then $b \leq c$. Hint: join the expressions in this equation with an appropriate element and use the *Distributive Law* along with what you know about the relational properties of the join.

11. Prove the first main claim in the *Stone Representation Theorem*: *Each element of a finite Boolean lattice can be expressed as the join of all the atoms that lie below it.*

12. Prove the second main claim in the *Stone Representation Theorem*: *Atomic join-expression representations of elements are unique.* Hint: take the meet of these expressions with the atoms of the lattice and use the *Distributive Law* plus the fact that the meet of two distinct atoms is 0.

13. Using the results cited in the last two problems, explain why, using what you know about counting principles, a Boolean algebra with n atoms must have 2^n distinct elements in it.

Problems 14 - 17: True or False

Are the following statements true or false? Explain your answer.

*14. A lattice with 10 elements cannot be a complemented distributive lattice.

*15. Finite Boolean lattices have the same lattice structure as the power-set lattice formed on the set of its atoms.

**16. There is no Boolean algebra having exactly 7 elements in it.

**17. Algebra can be used to represent logical statements, but equation solution procedures don't capture the ordinary reasoning process of deduction very well.

Problem 18 - 28: Propositions of Boolean Algebra

Prove the following propositions, using the axioms of Boolean Algebra or any proposition that precedes it.

18. *Proposition* 1: *Uniqueness Laws for Identities and Complements*
    a. If $x + z = x$ for all x, then $z = 0$.
    b. If $x \cdot u = x$ for all x, then $u = 1$.
    c. If $xz = 0$ and $x + z = 1$, then $z = \overline{x}$.

*19. *Proposition* 2: *Complements of Elements Laws*
    a. $\overline{0} = 1$
    b. $\overline{1} = 0$
    *c. $\overline{\overline{x}} = x$

20. *Proposition* 3: *Idempotence Laws*
    a. $x \cdot x = x$
    b. $x + x = x$

*21. *Proposition* 4: *Annihilation and Absorption Laws*
    a. $x \cdot 0 = 0$
    *b. $x + 1 = 1$
    *c. $x(x + y) = x$
    d. $x + xy = x$

22. *Proposition* 5: *De Morgan's Laws*
    a. $\overline{xy} = \overline{x} + \overline{y}$
    b. $\overline{x + y} = \overline{x}\,\overline{y}$

**23. *Proposition* 6: *Operation Linkage Laws*
    Prove that the following statements are logically equivalent in the context of *Boolean Algebra*.
    a. $xy = x$
    b. $x + y = y$
    c. $x\overline{y} = 0$
    d. $\overline{x} + y = 1$

**24. *Proposition* 7: *Redundancy Laws*
    **a. $x(\overline{x} + y) = xy$
    b. $x + \overline{x}y = x + y$

EC 25. *Proposition* 8: *Consensus Laws*
    a. $xy + \overline{x}z + yz = xy + \overline{x}z$
    b. $(x + y)(\overline{x} + z)(y + z) = (x + y)(\overline{x} + z)$

**26. *Proposition* 9: *Cancellation Law*
    **a. If $xy = xz$ and $x + y = x + z$, then $y = z$.
    **b. Is one equation in the antecedent condition of *Proposition* 9 (part *a*) sufficient for its conclusion? Explain.
    c. Prove or disprove the following *Cancellation Law*: if $xy = xz$ and $\overline{x}y = \overline{x}z$, then $y = z$.
    d. Prove or disprove the following *Cancellation Law*: if $xy = xz$ and $x\overline{y} = x\overline{z}$, then $y = z$.

EC 27. *Associative Law*
    Show that the *Associative Law* can be proved from the other four pairs of axioms and their consequences. Thus, it can be dropped from the list of axioms without losing it.†

EC 28. *Duality Principle*
    True or False: *Every theorem in Boolean Algebra has a dual theorem associated with it when* $+$ *and* $\cdot$ *as well as* 0 *and* 1 *are exchanged.* Carefully justify your answer.

---

† The remaining axioms are independent of one another, a result shown by American postulate theorist Edward Huntington in a 1904 paper on *Boolean Algebra*.

Problems 29 - 30: Boole's Algebraic Logic

The following problems relate to Boole's algebraic approach to logic.

**29. *Boole's Deduction of LNC from LEM*

a. Fill in all the details of the abbreviated argument given in Example 5 to show that the *Law of Non-Contradiction* follows from the *Law of Excluded Middle*. Begin by explaining why the LEM and the LNC are algebraically formulated the way they are.

b. Give a deduction of LNC from LEM using inference rules from *Sentential Logic*. Compare your proof with that of Example 5 (part *a*). Which one captures the process of sequential reasoning better?

**30. *Deducing Aristotelian Syllogisms*

a. Determine an equation that formulates the universal negative statement *No As are Bs*. Hint: start with a set-theoretic statement involving classes A and B that captures the meaning of this assertion and then translate it into Boolean notation.

b. Show how Boole's approach to deduction can be used to validate the following key argument-form *Celarent* from Aristotelian Logic: *No Xs are Ys, All Zs are Xs; therefore, No Zs are Ys.* Use only basic algebra to make your argument.

Problems 31 - 33: Sentential Logic as a Boolean Logic

The following problems relate to the interpretation of Sentential Logic as a Boolean algebra (*see Example* 6).

31. Interpret the *Identity Laws* of *Boolean Algebra* for *Sentential Logic* and explain why they are true, using what you know about SL.

a. $x + 0 = x = 0 + x$

b. $x \cdot 1 = x = 1 \cdot x$

**32. Interpret the *Complementation Laws* of *Boolean Algebra* for *Sentential Logic* and explain why they are true, using what you know about SL.

**a. $x + \overline{x} = 1 = \overline{x} + x$

b. $x \cdot \overline{x} = 0 = \overline{x} \cdot x$

33. Interpret the *Uniqueness Laws* of *Boolean Algebra* for *Sentential Logic* and explain why they are true, using what you know about SL.

a. If $x + z = x$ for all x, then $z = 0$.

b. If $x \cdot u = x$ for all x, then $u = 1$.

EC c. If $xz = 0$ and $x + z = 1$, then $z = \overline{x}$.

Problems 34 - 37: Boolean Algebras and Boolean Lattices

The following results explore the interconnections between Boolean algebras and Boolean lattices.

EC 34. *Alternative Characterization of Finite Boolean Lattices*

Let $\mathcal{B}^n$ denote the n-fold Cartesian product of $\mathcal{B} = \{0, 1\}$ with itself. The *product order* $\leq$ on $\mathcal{B}^n$ is defined by $(x_1, x_2, \ldots, x_n) \leq (y_1, y_2, \ldots, y_n)$ iff $x_i \leq y_i$ for all i.

a. Prove that $\leq$ is a partial order on $\mathcal{B}^n$. (This generalizes the order considered in Exercise 7.1-32).

b. Prove that $\langle \mathcal{B}^n, \leq \rangle$ forms a Boolean lattice.

c. Prove that the n-tuples $x = (x_1, x_2, \ldots, x_n)$ in $\mathcal{B}^n$ can be placed in natural one-to-one correspondence with $\mathcal{P}(S)$ for $S = \{1, 2, \ldots, n\}$. Hint: take $i \in R_x \subseteq S$ iff $x_i = 1$.

d. Prove that $\mathcal{B}^n$ and $\mathcal{P}(S)$ are *isomorphic* as lattices; i.e., prove that $(x_1, x_2, \ldots, x_n) \leq (y_1, y_2, \ldots, y_n)$ iff $R_x \subseteq R_y$, where R_x is the set of indices associated with $x = (x_1, x_2, \ldots, x_n)$ as in part *c*. Thus, by the *Stone Representation Theorem*, all finite Boolean lattices are given by $\langle \mathcal{B}^n, \leq \rangle$ for some n.

EC 35. *The Product Order and Finite Boolean Algebras $\mathcal{B}^n$*

a. Show that when the Boolean lattice $\mathcal{B}^n$, ordered by the product order (see Exercise 34), is converted into a Boolean algebra in the normal way, the resulting operations are performed coordinate-wise: $\overline{(x_1, x_2, \ldots, x_n)} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n)$, $(x_1, x_2, \ldots, x_n) + (y_1, y_2, \ldots, y_n) = (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n)$, and $(x_1, x_2, \ldots, x_n) \cdot (y_1, y_2, \ldots, y_n) = (x_1 \cdot y_1, x_2 \cdot y_2, \ldots, x_n \cdot y_n)$, where the component operations are the usual ones on the basic Boolean algebra $\mathcal{B}$ of 0 and 1.

b. Explain why the result of part *a* along with that of Exercise 34 establishes the fact that all finite Boolean algebras are essentially $\mathcal{B}^n$ under standard coordinate-wise operations.

36. *Theorem* 2
Suppose $\langle \mathcal{A}, +, \cdot, ^{-}, 0, 1 \rangle$ is a Boolean algebra, and $\leq$ is defined on $\mathcal{A}$ by $x \leq y$ iff $x \cdot y = x$. Prove the following claims asserted in the proof of Theorem 2.
   a. $\leq$ *is a partial order.*
   b. $x \cdot y$ *is the meet of* x *and* y.
   c. $x + y$ *is the join of* x *and* y.

37. *Boolean Conversions Undo One Another*
   a. Suppose $\langle \mathcal{A}, \leq, ^{-}, 0, 1 \rangle$ is a Boolean lattice. Then, as the text shows, it forms a Boolean algebra under the operations of $\vee$ being $+$ and $\wedge$ being $\cdot$. Show that if this Boolean algebra is reconverted into a Boolean lattice via the appropriate definition of partial order, $x \leq_1 y$ iff $x \cdot y = x$, the resulting order is exactly the same as the original partial order; i.e., $x \leq_1 y$ iff $x \leq y$.
   b. Suppose $\langle \mathcal{A}, +, \cdot, ^{-}, 0, 1 \rangle$ is a Boolean algebra. Show that if this is converted into a Boolean lattice via the appropriate definition of partial order, $x \leq y$ iff $x \cdot y = x$, the result is a structure whose join and meet agree with the original $+$ and $\cdot$; i.e., $x \vee y = x + y$ and $x \wedge y = x \cdot y$. Thus the result of reconverting this lattice into a Boolean algebra will give exactly the same structure as the original one.

38. Let S be any set, $\mathcal{P}(S)$ its collection of subsets.
   a. Show that the collection $\mathcal{A}$ of all finite subsets of S is not a Boolean algebra under the set operations of $\cup$ and $\cap$. What things are missing?
   b. Show that the collection $\mathcal{A}^*$ consisting of all finite and cofinite subsets of S (see Exercise 7.1-49) forms a Boolean algebra, the smallest one extending $\mathcal{A}$.
   c. Carefully explain why part b implies that not all Boolean algebras are power-set structures.

HINTS TO STARRED EXERCISES 7.3

1. Recall how 0 and 1 are added and multiplied in $\mathcal{B}$.

2. There are only a few equations to check for this.

9. Let $a \neq 0$ be any non-zero element in $\mathcal{A}$. Either a is minimal or it isn't. Consider these cases and continue.

10. See the hint given in the text.

14. [No hint.]

15. [No hint.]

16. [No hint.]

17. [No hint.]

19. c. Use uniqueness of complements and show x acts like $\overline{\overline{x}}$, the complement of $\overline{x}$; that is, show that x satisfies the conditions of *Proposition* 1c with x in for z and $\overline{x}$ in for x, and then apply that result to get your conclusion.

21. Be sure to use good proof strategy in deducing an equation: work from the left side to the right side; don't start with the equation you're trying to prove and transform it.
    b. Begin by multiplying the left side by 1 in an equivalent form and then expand by the *Distributive Law*.
    c. Begin by adding 0 to the left side in an equivalent form and then factor the result, using the *Distributive Law*. You will also need part b as you proceed.

23. Use a *CycBI* strategy here; i.e, prove a $\rightarrow$ b, b $\rightarrow$ c, c $\rightarrow$ d, and d $\rightarrow$ a.

24. a. Use the *Distributive Law*.

26. It bears repeating: make sure you use good proof strategy in deducing an equation. Start with something you know to be true.
    a. One way to work this: start with $y = y + yx$, an identity for y given by *Absorption* and continue through a sequence of equations, using the *Distributive Laws*, substitution of the given equations, and *Absorption*, until you finally end up with z. (There may be a shorter argument than this; so try your own approach first.)
    b. Look for a particular Boolean algebra that will offer a counterexample.

29. a. Use the reasons provided in the text to flesh out the argument.
    b. LNC needs no premises to be deduced, of course. But to show the logical connection between LNC and LEM, use the DeM and DN *Replacement Rules*.

30. a. How would you say in *Set Theory* that A and B have nothing in common? Translate your statement into Boolean notation.
    b. This is similar to Example 4.

32. a. This is fairly straightforward, given the interpretation for the symbols. See Example 6.

7.4 Boolean Functions and Logic Circuits

After Section 7.3, we know that Boolean lattices are Boolean algebras, and vice versa. Each structure can be converted into the other by using the appropriate definitions. Even better, we can think of a Boolean system simultaneously in two ways: as a relational structure ordered by $\leq$ (a lattice) and as an operational structure with two operations $+/\vee$ and $\cdot/\wedge$ (an algebra). Sometimes it will be handier to adopt a relational perspective; at other times an operational outlook will be easier to use. Both points of view are helpful for recalling the various laws that hold for such a structure and for testing the validity of an identity, and both viewpoints are legitimate: we need not choose for one over the other. A Boolean system always has both sorts of interconnected structure.

In Section 7.3 we also introduced a connection between *Boolean Algebra* and *Sentential Logic*. Here we will explore this in more detail. Our main goal will be to show how logic and *Boolean Algebra* apply to computational circuitry. In preparation for this, we will discuss some ideas related to two-valued Boolean algebras and functions defined on them.

Two-Valued Boolean Algebras

The *Stone Representation Theorem* tells us that finite Boolean lattices/algebras must be of size 2^n, structured like the power set of $\{1, 2, \ldots, n\}$ under the subset relation and the operations of union, intersection, and complementation.

Another way to realize such Boolean structures, as noted in Example 7.3-1, is in terms of the two-valued Boolean lattice $\mathcal{B} = \{0, 1\}$, ordered by $0 < 1$. The n-fold Cartesian product $\mathcal{B}^n = \{(x_1, x_2, \ldots, x_n) : x_i \in \mathcal{B}\}$ is then the set of all length-n sequences of 0s and 1s. Ordering its elements under the *product order*, given by $(x_1, x_2, \ldots, x_n) \leq (y_1, y_2, \ldots, y_n)$ iff $x_i \leq y_i$ for all $i = 1, 2, \ldots, n$, turns $\mathcal{B}^n$ into a Boolean lattice of size 2^n (see Exercise 7.3-34). This induces a natural operational structure on it as well; the operations of $+$ and $\cdot$ turn out to be those done coordinate-wise on the elements of $\mathcal{B}$ (see Exercise 7.3-35).

The simplest non-trivial Boolean algebra is $\mathcal{B} = \{0, 1\}$. Its addition and multiplication tables are given as follows, the values being those dictated by the *Identity*, *Annihilation*, and *Absorption Laws* of *Boolean Algebra*.

$+$	0	1
0	0	1
1	1	1

$\cdot$	0	1
0	0	0
1	0	1

We will be working with this Boolean algebra in what follows, though much of what we do can be further generalized to other Boolean algebras.

Boolean Functions

A Boolean function is one whose domain and codomain involve the two-valued Boolean algebra $\mathcal{B} = \{0, 1\}$. More precisely, we have the following.

DEFINITION* 7.4-1: *Boolean Function
A Boolean function is a function $f : \mathcal{B}^n \to \mathcal{B}$.

Thus, Boolean functions can be thought of as Boolean-valued functions of several Boolean variables, the only inputs and outputs allowed being 0 and 1. You already know a number of such Boolean functions, though to this point you've thought of them as Boolean operations.

✠ **EXAMPLE 7.4 - 1**

Exhibit the operations of Boolean addition and multiplication as Boolean functions.

Solution

The operations of $+$ and $\cdot$ can be considered functions from $\mathcal{B}^2$ to $\mathcal{B}$. Here are the tables (sometimes referred to as truth tables) that represent these operations. Notice the similarity of these tables to those for $\vee$ and $\wedge$ in SL if 0 is interpreted as F and 1 as T.

x_1	x_2	$+(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	$\cdot(x_1, x_2)$
0	0	0
0	1	0
1	0	0
1	1	1

✠ **EXAMPLE 7.4 - 2**

Write out the table of the *ternary majority function* given by $f(x_1, x_2, x_3) = m$, where m is the value 0 or 1 that appears among the variables x_1, x_2, x_3 a majority of the time.

Solution

The following table gives an explicit function output for each input sequence. Note the dictionary order used in the table for listing the input values. This is the exact opposite of how truth tables were organized for *Sentential Logic*.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Boolean Algebra and Sentential Logic

We'll now look at a few simple examples of Boolean functions and use them to connect *Boolean Algebra* more closely with *Sentential Logic*.

✠ **EXAMPLE 7.4 - 3**

Discuss the relation of the following Boolean functions to *Sentential Logic* connectives.

a)

x	$N(x)$
0	1
1	0

b)

x_1	x_2	$C(x_1, x_2)$
0	0	1
0	1	1
1	0	0
1	1	1

c)

x_1	x_2	$D(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	1

Solution

a) The first Boolean function N can be thought of as assigning truth values to negations. For if x denotes (the truth value of) a given proposition, with 0 standing for being false and 1 for being true, then $N(x)$ is just the truth value of the negation of that proposition.

b) Similarly, the Boolean function C can be thought of as assigning truth values to conditional sentences $x_1 \to x_2$.

c) The Boolean function D corresponds to assigning truth values to disjunctions.

It should be clear from these examples that associated with every Boolean function there is a *Sentential* connective (possibly an unfamiliar one, defined simply by its truth table) and that every *Sentential* connective generates a Boolean function via its truth table. This correspondence between *Sentential Logic* and *Boolean Algebra* has some very fruitful consequences. Whatever we can say about *Sentential* connectives can be translated into an associated statement about Boolean functions, and conversely. We will see the value of this as we continue.

Historical Aside: From Boole to Shannon

As we noted in Section 7.3, in the mid-nineteenth century George Boole introduced his ideas about an algebra of 0 and 1 in the context of theorizing about logic and reasoning. Logic was central in his thought; *Boolean Algebra* (in germinal form) was put forward in order to give some legitimacy to his using algebra to treat logical derivations. Since his system of algebra was unorthodox, containing as it did such novel laws as $x^2 = x$, Boole was happy to point out that such a result holds if algebra is restricted to the numbers 0 and 1. He never adopted an abstract approach to *Boolean Algebra*, however, nor was he concerned about its possible application to other fields of thought.

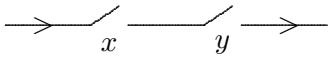
The American mathematician Claude Shannon first recognized the broader potential of an algebraic approach to logic. His path-breaking 1938 M.A. thesis, *A Symbolic Analysis of Relay and Switching Circuits*, showed how *Boolean Algebra* could be physically realized. Conversely, Shannon showed how to use *Boolean Algebra* to analyze, simplify, and design electrical circuits, thus establishing it as an essential tool of contemporary digital electronics.

Shannon's later work continued to exploit the possibilities inherent in the algebra of 0 and 1. His most important publication, *A Mathematical Theory of Communication*, came out in 1948. Here he laid the groundwork for the information revolution of the last half-century by conceptualizing information in terms of bits (0's and 1's). He also showed that information noise and data corruption could be overcome by transmitting messages with redundancy built into the coding. Other ideas of his were instrumental to encrypting messages. The widely used Data Encryption Standard (DES), for example, arose from work Shannon did during the 1940s.

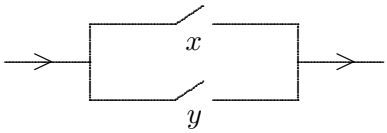
Boolean Algebra and Switching Circuits

The main link between *Boolean Algebra* and computer logic involves logic gates, which we will turn to shortly, but we will start by looking at the simpler case of switching circuits. This was Shannon's original idea for how *Boolean Algebra* could be applied to telecommunication.

Simple switches have two states: open (no current passing through) and closed (current flowing through). We will represent an open-switch state with 0 and a closed-switch state with 1.* Switches can be connected either in series or in parallel, as shown in the following diagrams. Switches in series are represented by the product of the switch labels since current flows through the circuit (yields output 1) iff it flows through both switches. Switches in parallel are similarly denoted by a Boolean sum of the switch labels.



$\mathbf{x \cdot y}$



$\mathbf{x + y}$

Series and Parallel Circuits

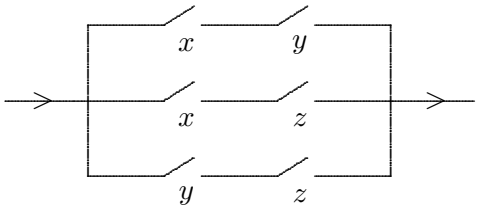
Complex circuits can be constructed out of these basic components. If two switches in a circuit are controlled so that they are always in the same state, the same letter is used for both switches. If one switch is always in the opposite state of another one, an overbar is used to indicate the negative relationship. In this way, every circuit can be associated with a Boolean expression involving switch labels: switching circuits provide a concrete representation for Boolean functions.

✠ **EXAMPLE 7.4 - 4**

Design a series-parallel switching circuit and a Boolean expression for the ternary majority function of Example 2.

Solution

Since we want output 1 (current flowing through) when two or more switches are closed (value 1), the following circuit models the ternary majority function. The Boolean expression for this function/circuit is given below the diagram.



$\mathbf{x \cdot y + x \cdot z + y \cdot z}$

✠ **EXAMPLE 7.4 - 5**

Design a switching circuit for two switches controlling a single light in the usual way (confusingly called connecting "three-way switches" in U.S. circles).

* This is now the standard assignment, though Shannon's original choice reversed these values and also the operators used for circuit types.

Solution

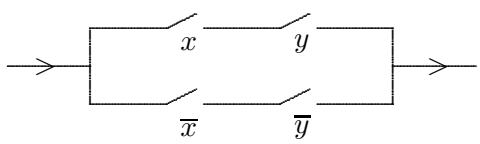
We will first determine an appropriate Boolean function F.

Taking output 1 to represent the light being on and letting variables x and y stand for the two switches, with input 1 standing for closed and 0 for open, we choose $F(1,1) = 1$. Changing the state of either switch while keeping the other one the same will turn the light off. So $F(0,1) = 0$ and $F(1,0) = 0$. Changing the state of both switches should turn the light back on, so $F(0,0) = 1$. This gives the following function table.

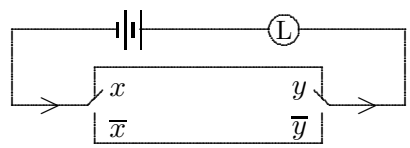
x_1	x_2	$F(x_1, x_2)$
0	0	1
0	1	0
1	0	0
1	1	1

This resembles the truth table for $x_1 \leftrightarrow x_2$. This expression is logically equivalent by the last form of *Bicndnl* to $(x \wedge y) \vee (\neg x \wedge \neg y)$, which is represented in *Boolean Algebra* by $x \cdot y + \overline{x} \cdot \overline{y}$. This is an expression that can be developed into a series-parallel circuit. The following switching circuit on the left, therefore, is our solution.

Naturally, this is not yet a wiring diagram for having two light switches control a light, but it can be converted into one, as indicated by the three-way switch diagram on the right.

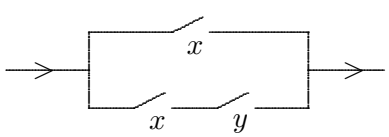


$\mathbf{x \cdot y + \overline{x} \cdot \overline{y}}$



✠ **EXAMPLE 7.4 - 6**

Determine the Boolean expression representing the following switching circuit. Then design a simpler circuit to accomplish the same thing.

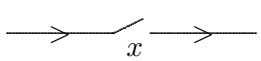


Solution

The Boolean expression for this circuit is $x + xy$.

By the *Absorption Law* for *Boolean Algebra*, this expression is equivalent to x, so the circuit can be simplified in this case merely by dropping off the lower portion of the circuit.

A simplified circuit is therefore as follows.



As this example demonstrates, given a compound circuit, one can represent it via a Boolean expression and then use *Boolean Algebra* to simplify the expression in some way. This equivalent simplified expression will represent the same Boolean function. Using this expression, one can then design a simpler switching circuit that acts exactly like the original one. The insights provided by *Boolean Algebra*, abstract as it is, may allow a substantial savings in concrete circuit design. Because of the profitability of this sort of mathematical theorizing, Bell Labs has for years had mathematicians such as Shannon on its Research and Development payroll.

Boolean Algebra and Circuits from Logic Gates

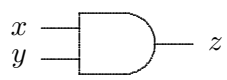
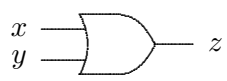
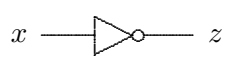
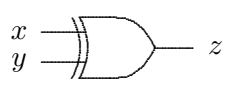
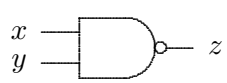
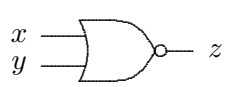
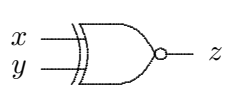
Computers make use of integrated electronic circuits built out of *logic gates* of various sorts. Logic gates are circuits whose outputs are completely determined by the inputs in fixed ways; they are still incapable of generating an output based both upon the input and the present state/memory of the machine. To handle this, existing mathematical models of computation go beyond combinatorial logic-gate circuitry to include structures known as finite-state machines and Turing machines. These are outside the scope of this text but are taken up in a number of discrete mathematics texts and computer science texts. However, already with logic-gate circuits we can begin to see how basic arithmetic is performed.

We will start by considering the logic gates corresponding to standard binary and unary logical connectives. The various logic gates available are symbolized as shown in the table below. The inputs are placed left of the diagram, the outputs on the right. Note how negating or inverting another operator is indicated by including an inverter node before the output.

Strictly speaking, each of the basic logic gates except for the inverter NOT-gate is a double-input gate. However, given the associative and commutative properties for the Boolean operators AND and OR, we can allow their respective logic gates (as well as those for NAND and NOR) take on more than two input lines; no ambiguity will be introduced by this practice.

A value of 1 for input or output represents high voltage; 0 indicates low voltage. If we consider the output 1 as the accepting state of a logic gate, we can specify just which string inputs xy it *accepts*, putting it into that state. We will use this same terminology when speaking of Boolean functions; a Boolean function *accepts* an input string if it outputs a 1 for that string.

Logic Gates

Name	**Symbol**	**Accepts Strings**	**Boolean Expression**
AND		$xy = 11$	$z = x \cdot y$
OR		$xy = 01$, 10, or 11	$z = x + y$
NOT		$x = 0$	$z = \overline{x}$
XOR		$xy = 01$ or 10	$z = x \oplus y$ $= x \cdot \overline{y} + \overline{x} \cdot y$
NAND		$xy = 00$, 01, or 10	$z = \overline{x \cdot y}$ $= \overline{x} + \overline{y}$
NOR		$xy = 00$	$z = \overline{x + y}$ $= \overline{x} \cdot \overline{y}$
XNOR		$xy = 00$ or 11	$z = \overline{x \oplus y}$ $= x \cdot y + \overline{x} \cdot \overline{y}$

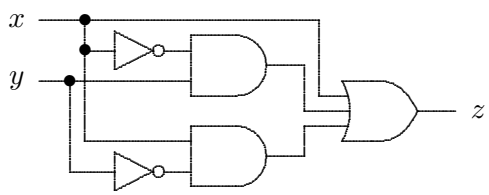
Logic gates can be represented by Boolean expressions (the last column of the table), just as with switching circuits. The choice of expression, like the name of the gate itself, depends on the functional output of the circuit. The gate that outputs high voltage iff it has two high voltage inputs (the AND gate) corresponds to multiplication because that matches the way we get 1 as a product in *Boolean Algebra*. Other Boolean expressions are assigned in a similar manner.

Logic gates can be connected by using their outputs as inputs for other gates, making more complex circuits. The most primitive logic gates from a logical and an algebraic point of view are the AND, OR, and NOT gates. These can obviously be combined to duplicate the action of NAND and NOR gates, but as a matter of fact, any desired output whatsoever can be generated using them. Together they form a complete set of connectives, something we will prove in Section 7.5. The gate most often used in integrated circuits, however, is the NAND gate. It, too, is adequate: any logical connective/Boolean expression can be modeled with NAND gates appropriately connected (see Exercises 41 – 43).

Different compound circuits may be equivalent, yielding the same outputs for the same inputs. Thus, even though their Boolean expressions may differ, they may still represent the same Boolean function. As before, the laws of *Boolean Algebra* may be used to show this equivalence. And, more importantly, they can be used to simplify logic-gate-circuit expressions in order to design simpler equivalent circuits. This topic will be explored in more depth in the next two sections.

✠ **EXAMPLE 7.4 - 7**

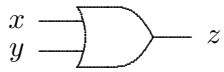
Determine the Boolean expression representing the following logic-gate circuit; then design a simpler circuit that accomplishes the same thing.



Solution

This circuit realizes the Boolean expression $x+\overline{x}y+x\overline{y}$. By the *Absorption* and *Redundancy Laws* of *Boolean Algebra*, this expression simplifies to $x+\overline{x}y+x\overline{y}=x+\overline{x}y=x+y$.

A simpler logic-gate circuit, therefore, is given by the following diagram.



At this point we are using Boolean identities to simplify our expressions. This gives some valuable practice in working with *Boolean Algebra*. In Section 7.6, however, we will learn some easier, more systematic techniques for doing such simplifications.

Computation Performed Via Logic Circuits

Computation in the form of *Boolean Algebra* contributed its resources for Boole to model a version of logic. Logic in the form of logic-gate circuits reciprocates by providing resources for doing numerical computation. Being able to do this is a rather amazing accomplishment and required some ingenious thinking about the potential of logical operations in their algebraic format. This development provided the foundation for modern electronic computation. In its

most elementary form, computation is performed by means of *Half Adders* and *Full Adders*, using binary (base two) representation of numbers for doing addition.

The *Half Adder* initiates the process by enabling us to add two 0s or 1s. It is a two-output circuit that adds two bits (ordinary numbers in binary form) in the usual way, yielding their arithmetic sum (*not* the sum done according to the addition operation of *Boolean Algebra*). The only sum that requires some care is that of $1+1=10$ (i.e., one plus one equals two). The combined table for adding any two bits x_0 and y_0, giving binary expression $z_1\,z_0$, is given as follows.

$$\begin{array}{r} x_0 \\ +\,y_0 \\ \hline z_1\,z_0 \end{array}$$

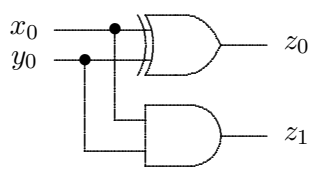
x_0	y_0	z_1	z_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

✠ **EXAMPLE 7.4-8**

Design a *Half Adder* to realize the Boolean functions $z_0 = f_0(x_0, y_0)$ and $z_1 = f_1(x_0, y_0)$ given above for adding two bits.

Solution

Familiarity with Boolean expressions for the basic logic gates shows that $z_1 = x_0$ AND y_0 while $z_0 = x_0$ XOR y_0. The *Half Adder* thus has the following circuit diagram. A diagram using only AND, OR, and NOT gates will be left as an exercise (see Exercise 45a).



The *Half Adder* works fine for adding two single-bit numbers, for then there are only two inputs involved. However, in adding two multiple-bit numbers, such as $111 + 11$, there will often be a third bit to add resulting from the carry of the bit sum in the previous place. To accommodate this, a *Full Adder* is needed, a circuit having three inputs and two outputs. The table for the associated Boolean function is given below; a logic gate diagram for this using logic gates and *Half Adders* is left as an exercise (see Exercise 46).

$$\begin{array}{r} w_0 \\ x_0 \\ +\,y_0 \\ \hline z_1\,z_0 \end{array}$$

w_0	x_0	y_0	z_1	z_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

EXERCISE SET 7.4

Problems 1 - 6: Boolean Functions and Sentential Logic

Write out Boolean function tables to represent the following Sentential connectives/operators.

1. AND $(x_1, x_2) = (x_1 \wedge x_2)$

*2. NAND $(x_1, x_2) = \neg(x_1 \wedge x_2)$

3. NOR $(x_1, x_2) = \neg(x_1 \vee x_2)$

4. XOR $(x_1, x_2) = (x_1 \vee x_2) \wedge \neg(x_1 \wedge x_2) = (x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$

*5. XNOR $(x_1, x_2) = (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)$. What other name does this connective go under?

6. FALSE $(x) = x \wedge \neg x$. This is the constantly-false operator. Which binary connective can be used to represent this?

Problems 7 - 10: True or False

Are the following statements true or false? Explain your answer.

7. Every sentence in SL can be associated with a corresponding Boolean function.

*8. Distinct Boolean polynomial expressions in n variables define distinct Boolean functions $F : \mathcal{B}^n \to \mathcal{B}$.

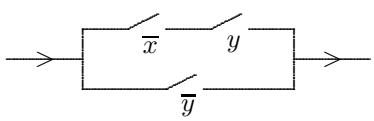
9. If an XOR gate rejects a string of inputs, a NOR accepts it.

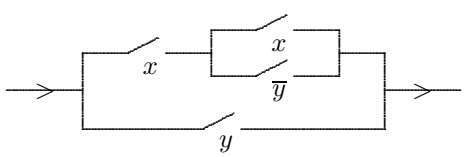
*10. Boole was the first to see the possibility of using algebra and logic for performing computations.

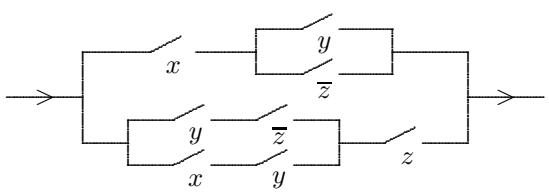
Problems 11 - 14: Boolean Expressions for Switching Circuits

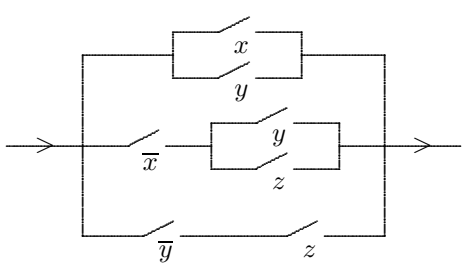
For each switching circuit given below, do the following:

a) *Determine the Boolean expression corresponding to the switching circuit.*
b) *Simplify the Boolean expression as much as possible, using the laws of Boolean Algebra.*
c) *Draw the circuit corresponding to your simplified expression.*

11. 

12. 

*13. 

14. 

Problems 15 - 18: Switching Circuits for Boolean Expressions

For each Boolean expression given below, do the following:

a) *Draw a switching-circuit diagram to realize the given Boolean expression.*

b) *Simplify the Boolean expression as much as possible, using the laws of Boolean Algebra.*

c) *Draw the circuit corresponding to your simplified expression.*

15. $(x + \overline{y})\, xy$

*16. $xy + x\overline{y} + \overline{x}y$

17. $xyz + \overline{y}z + \overline{x}yz$

18. $(x + y)(x + z)(y + z)$

Problems 19 - 20: Circuit Applications

Use a switching circuit or a logic-gate circuit to model the situation given in each problem.

19. Develop a Boolean function table, with rationale, and a switching circuit to model three switches controlling a light.

*20. Four top-level naval officials in a submarine have sufficient rank to authorize the firing of a nuclear warhead. Because of the seriousness of this action, however, the decision to fire can only be taken when at least three of the four agree to fire. Give a Boolean expression and then design a logic circuit that realizes this protocol. You may use generalized logic gates having more than two inputs.

Problems 21 - 24: Boolean Expressions for Logic Circuits

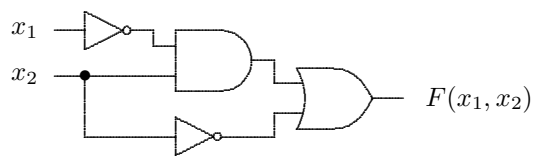
For each logic circuit given below, do the following:

a) *Determine the Boolean expression corresponding to the circuit.*

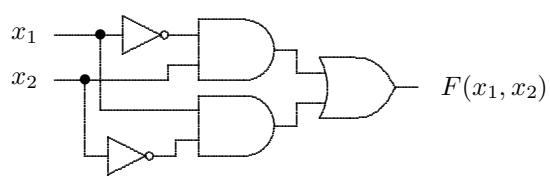
b) *Simplify the Boolean expression as much as possible, using the laws of Boolean Algebra.*

c) *Draw the logic circuit corresponding to your simplified expression.*

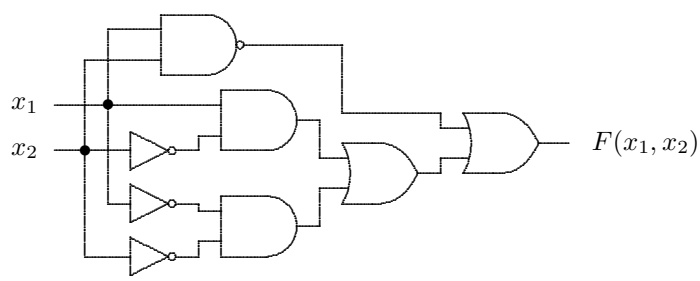
*21.



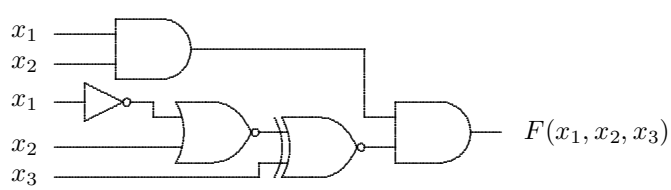
22.



23.



*24.



Problems 25 - 28: Logic Circuits for Boolean Expressions

For each Boolean expression given below, do the following:

a) *Draw a logic-gate-circuit diagram to realize the given Boolean expression.*

b) *Simplify the Boolean expression as much as possible, using the laws of Boolean Algebra.*

c) *Draw the logic-gate circuit corresponding to your simplified expression.*

25. $\overline{x}y + (\overline{x+y})$

26. $xy + x\overline{y} + \overline{x}y$

*27. $(xy + \overline{y})(x + \overline{x}\,\overline{y})$

28. $xyz + x\,\overline{y}\,\overline{z} + \overline{x}yz + \overline{xyz}$

Problems 29 - 30: Logic Circuits for Boolean Functions

For each Boolean function given below, do the following:

a) *Determine a Boolean expression to represent the given function.*

b) *Draw a logic-gate circuit that will implement the function.*

29. $f(x, y, z) = 1$ iff $y = 1$ and $x \neq z$.

EC 30. $f(x, y, z)$ is given by the following table.

x	y	z	$f(x, y, z)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Problems 31 - 37: Logic-Gate Equivalents

Show how the following logic gates can be simulated using only ***NOT***, ***AND***, *and* ***OR*** *gates; that is, find a logic-gate circuit using only these gates to represent the same Boolean function as the given one.*

31. NAND

32. NOR

33. XOR

34. XNOR

*35. Show how AND can be simulated using only NOT and OR gates.

36. Show how OR can be simulated using only NOT and AND gates.

EC 37. Can NOT be simulated using only AND and OR gates? Explain.

Problems 38 - 43: Boolean Functions/Logic Gates Having One or Two Inputs

The text gives 7 *basic logic gates/Boolean functions, one with one variable and* 6 *with two variables. These are not the only ones possible, however. The following problems explore logic circuits for other connectives.*

38. Is there a basic logic gate corresponding to the logical connective $\rightarrow$? corresponding to the logical connective $\leftrightarrow$? Explain.

39. How many distinct Boolean functions of a single variable are there? Write down function tables for those that are not already included among the basic logic gates and give them a name of your choice.

40. How many distinct Boolean functions of two variables are there? Write down function tables for those that are not already included among the basic logic gates and give them a name of your choice.

*41. Show how NOT can be simulated using only a NAND gate.

EC 42. Show how AND can be simulated using only NAND gates.

43. Show how OR can be simulated using only NAND gates.

Problems 44 - 47: Ordinary Computation Performed Via Logic Circuits

The following problems explore integer computations and comparisons done via logic-gate circuits.

44. *Need for Full Adders*
    a. Explain in your own words, considering the binary sum $111 + 11$, why using a *Half Adder* to add the bits in each place value fails to generate the necessary sum.
    b. Explain in general why adding two binary numbers in a given place value may require three inputs but only needs two outputs. How many inputs and outputs will be required for the ones' place?

*45. *Half-Adder Diagrams*
    *a. Provide an alternative circuit diagram for the *Half Adder*, using only AND, OR, and NOT logic gates.
    EC b. Design a *Half-Adder* circuit that uses only NAND logic gates.
    c. Design a *Half-Adder* circuit that uses only NOR logic gates.

EC 46. *Full Adders from Half Adders*
    a. By analyzing what is needed to add three bits, $w_0 + x_0 + y_0 = z_1 z_0$, show that a *Full Adder* can be built out of three *Half Adders* that each add two bits at a time. For simplicity, you may use represent each *Half Adder* as a rectangle with two inputs and two outputs, labeled appropriately. Explain how your circuit works on the inputs $(1, 0, 1)$ and $(1, 1, 1)$.
    b. By analyzing the possibilities that can occur in adding three bits, show that the final *Half Adder* in part *a* can be replaced with a simple logic gate and identify what that gate is.
    c. Using the logic-circuit diagram in Example 8 or your work in Exercise 45, draw a logic circuit for a *Full Adder* that uses only basic logic gates.

47. *Addition of Three-Bit Numbers*
    a. Show how one can use one *Half Adder* and two *Full Adders* to add two three-bit numbers, $x_2x_1x_0 + y_2y_1y_0$, yielding the sum $z_3z_2z_1z_0$. To simplify your diagram, you may represent each type of adder as a rectangle with two or three inputs and two outputs, labeling each component, input, and output appropriately.
    b. Explain how your circuit from part *a* adds the binary numbers $101 + 11$ to yield 1000.

48. *Ordinary Multiplication using Logic Circuits*
    a. Write down a function table for ordinary integer multiplication of two one-bit numbers ($z_0 = x_0 \cdot y_0$) and then draw the simple logic circuit that implements it.
    b. Design a logic circuit to multiply a two-bit number x_1x_0 by a one-bit number y_0 to get a two-bit number z_1z_0.
    c. Design a logic circuit to multiply two two-bit numbers x_1x_0 and y_1y_0 to get a four-bit number $z_3z_2z_1z_0$. Use your knowledge of the ordinary multiplication algorithm to help you design this.

49. *Ordering Numbers in Binary Notation*
    For the purposes of this problem, consider a number x to be *smaller than* a number y iff $x \leq y$.
    Recall: Boolean algebras are also Boolean lattices.
    a. Design a simple logic circuit that compares two one-bit numbers x_0 and y_0 and then outputs the smaller number.
    b. Design a logic circuit that compares two genuine two-bit numbers x_1x_0 and y_1y_0 and then outputs the smaller number.
    c. Design a logic circuit that compares two two-bit numbers x_1x_0 and y_1y_0 and then outputs the smaller number. This time allow the lead bit to be either a 0 or a 1.

HINTS TO STARRED EXERCISES 7.4

2. Like Example 3, write out the truth table for NAND, using 0s and 1s.

5. Write out the truth table for XNOR, using 0s and 1s.

8. [No hint.]

10. [No hint.]

13. See Example 6 for a similar, though simpler, problem.

16. Use the switching circuit realizations for $+$ and $\cdot$ to draw the circuit diagram. Then simplify and redraw the circuit, as in Example 6.

20. Knowing which input strings such a logic circuit should accept, build the circuit out of generalized logic gates that can handle more than two inputs.

21. See Example 7 for a similar problem. Use the Logic-Gates chart to interpret the gates involved.

24. See Example 7 for a similar problem. Use the Logic-Gates chart to interpret the gates involved.

27. Use the logic gates corresponding to the operations of Boolean addition, Boolean multiplication, and Boolean complementation to draw a diagram for the expression. Simplify and redraw it as in Example 7.

35. Use your knowledge of logical equivalents to do this: how can x AND y be expressed in terms of ORs and NOTs?

41. Find a way to express NOT-x using NANDs.

45. a. Rework the diagram given in Example 8, replacing the XOR gate with its equivalent.

7.5 Representing Boolean Functions

A Boolean function, according to Definition 7.4-1, is a function from $\mathcal{B}^n$ to $\mathcal{B}$, where $\mathcal{B} = \{0, 1\}$. Since $\mathcal{B}$ is a Boolean algebra, we can use Boolean operations to construct complex Boolean functions out of simpler ones in the usual way by operating on their outputs: the output of the sum/product of two functions is the sum/product of the functions' outputs, and the output of the complement of a function is the complement of the function's output (see the definition preceding Exercise 5). Conversely, we can decompose Boolean functions into simpler ones using Boolean operations.

In this section we will explore uniform ways to represent Boolean functions and expressions by generating them from elementary component functions and expressions, called minterms and maxterms. And, since Boolean functions can be used to model *Sentential Logic* connectives, our findings will give us corresponding results about so-called normal-form representations for SL formulas. At the same time it will establish the pleasantly surprising fact that certain small sets of familiar logical connectives are expressively complete; that is, they suffice for representing all possible connectives of any degree of complexity. Finally, we will note an interesting implication of this work for developing the theory of *Boolean Algebra*.

Boolean Functions and Boolean Expressions

In many respects Boolean functions are simpler than functions you've worked with in earlier mathematics courses: after all, their inputs and outputs only involve the numbers 0 and 1. On the other hand, computation with these values as elements of the Boolean algebra $\mathcal{B}$ is different than ordinary computation, so this introduces a degree of complexity and unfamiliarity.

For real-valued functions of a real variable, different polynomial expressions represent different functions. In the case of Boolean polynomials, however, there turns out to be a great deal of duplication.

✠ **EXAMPLE 7.5-1**

Compare the functions $f(x) = x$ and $g(x) = x^2$, both as functions from $\mathbb{R}$ to $\mathbb{R}$ and as functions from $\mathcal{B}$ to $\mathcal{B}$.

Solution

As functions from $\mathbb{R}$ to $\mathbb{R}$, f and g are distinct. Their graphs are very different: f is a linear function, while g is a quadratic. Their action differs on 2, for instance: $f(2) = 2 \neq 4 = g(2)$. However, both functions agree on inputs 0 and 1 (from either domain), so as functions from $\mathcal{B}$ into $\mathcal{B}$, they are completely identical: $f = g$. In fact, we know from *Boolean Algebra* that $x^2 = x$ regardless of input: this is the content of the *Idempotence Law* for multiplication. Thus, different Boolean polynomials may give rise to the same function.

We can make the same point in an even stronger way, as the next example shows. Thus, while there are infinitely many distinct polynomial formulas of a single variable, they all reduce down to one of the four basic expressions given below.

✠ **EXAMPLE 7.5-2**

Determine all functions $f : \mathcal{B} \to \mathcal{B}$.

Solution

Since $\mathcal{B} = \{0, 1\}$, there are exactly 4 distinct functions from $\mathcal{B}$ to $\mathcal{B}$. We can even identify them using standard Boolean formulas: they are $f_{00}(x) = 0$, $f_{11}(x) = 1$, $f_{01}(x) = x$, and $f_{10}(x) = \overline{x}$ (see Exercise 2).

To represent a Boolean function by means of a formula, we can use the algebraic structure of $\mathcal{B}$ to generate expressions involving addition, multiplication, and complementation. We will call such formulas *Boolean expressions*.

DEFINITION* 7.5 - 1: *Boolean Expressions

A ***Boolean expression*** *in the n variables $x_1, x_2, \ldots, x_n$, is any expression $E(x_1, x_2, \ldots, x_n)$ formed from these variables and the constants* 0 *and* 1 *via the operations of Boolean Algebra.*

More explicitly, Boolean expressions in $x_1, x_2, \ldots, x_n$ are defined recursively as follows:

1) Constants: 0 *and* 1 *are Boolean expressions in $x_1, x_2, \ldots, x_n$.*
2) Variables: x_1, x_2, ..., x_n *are Boolean expressions in $x_1, x_2, \ldots, x_n$.*
3) Sums, Products, and Complements: *If E and F are Boolean expressions in $x_1, x_2, \ldots, x_n$, then so are $E + F$, $E \cdot F$, and $\overline{E}$.*

Thus, E is a Boolean expression in $x_1, x_2, \ldots, x_n$ iff it has been generated in a finite number of steps from the constants and variables using sums, products, or complements.

✠ **EXAMPLE 7.5 - 3**

Explain why each of the following is a Boolean expression in x and y, and then simplify the expression.

a) $x^2 + xy + 1$
b) $(x^3 + \overline{x}y)^2$

Solution

a) The formula $x^2 + xy + 1$ is generated by first multiplying the variables x and x, and x and y, and then adding these products together along with the constant 1. Thus, $x^2 + xy + 1$ is a Boolean expression in x and y.
However, since 1 added to anything yields 1 in *Boolean Algebra*, this expression is equal to 1 for all values of x and y: $x^2 + xy + 1 = 1$.

b) The formula $(x^3 + \overline{x}y)^2$ is formed from the variables x and y by first multiplying three x factors together and multiplying the complement of x by y, then adding these products together, and multiplying the result by itself. Thus, $(x^3 + \overline{x}y)^2$ is a Boolean expression in x and y.
Since squaring an input yields that input in *Boolean Algebra*, $(x^3 + \overline{x}y)^2 = x^3 + \overline{x}y$. For the same reason, $x^n = x$, so $x^3 + \overline{x}y = x + \overline{x}y$. But this last expression can be simplified by a *Redundancy Law* to $x + y$. Thus, $(x^3 + \overline{x}y)^2$ boils down to $x + y$.

If $E(x_1, x_2, \ldots, x_n)$ and $F(x_1, x_2, \ldots, x_n)$ are Boolean expressions and $E(x_1, x_2, \ldots, x_n) = F(x_1, x_2, \ldots, x_n)$ is an *identity* derivable in *Boolean Algebra* (i.e., $E = F$ is a theorem of *Boolean Algebra*), it must therefore hold for all Boolean algebras, including $\mathcal{B}$. Thus, the functions from $\mathcal{B}^n$ to $\mathcal{B}$ defined by these expressions are also equal since the identity must hold for any assignment of 0s or 1s to the variables. Determining that two expressions represent the same function, then, can be done either by showing that their outputs agree on all possible input strings of 0s and 1s (give identical function tables) or by showing the identity of the expressions via the laws of *Boolean Algebra*. At this point we don't know conversely that two expressions which agree for all assignments of 0s and 1s to the variables are also a *Boolean* identity, but we will show this as a corollary to *Theorem* 2 below.

✠ **EXAMPLE 7.5 - 4**

Find a simpler Boolean formula for the function $f : \mathcal{B}^2 \to \mathcal{B}$ defined by $f(x, y) = x + xy$.

Solution

According to one of the *Absorption Laws*, $x + xy = x$. A simpler formula is thus given by $f(x, y) = x$; f is the projection map onto the first coordinate.

Representing Boolean Functions via Minterm Expressions

We've seen that multiple Boolean expressions, some simpler than others, can represent the same Boolean function. Going in the other direction, is it always possible to find some Boolean expression to represent a Boolean function? The answer here turns out to be a very satisfying "yes": there is a standard class of natural expressions that can be used to represent Boolean functions. In fact, this can be done in two uniform ways. We will begin by working with minterm expressions; maxterm expressions will be treated at the end of the section. We'll introduce the process via an example.

✠ **EXAMPLE 7.5 - 5**

Determine a Boolean expression for the Boolean function f defined by the following table.

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	0
1	0	0
1	1	1

Solution

- We can tackle this in several ways. One way to do it is to use what you already know about *Sentential Logic*. As a truth table (in upside-down order), you may recognize that this function represents $x_1 \leftrightarrow x_2$ and then continue to find a Boolean expression for this; we followed this approach in Example 7.4 - 5.
- Another way is to use what you know about addition, multiplication, and complementation in the target Boolean algebra $\mathcal{B}$. We'll concentrate on when a Boolean-function output is 1; that is, on which input strings the function accepts. We'll work our way up from the bottom of the table.

  $f(x_1, x_2) = 1$ iff $x_1 = 1$ and $x_2 = 1$ (bottom row), or $x_1 = 0$ and $x_2 = 0$ (top row);
  iff $x_1 = 1$ and $x_2 = 1$, or $\overline{x}_1 = 1$ and $\overline{x}_2 = 1$;
  iff $x_1 \cdot x_2 = 1$, or $\overline{x}_1 \cdot \overline{x}_2 = 1$;
  iff $x_1 \cdot x_2 + \overline{x}_1 \cdot \overline{x}_2 = 1$.

  Thus, $f(x_1, x_2) = x_1x_2 + \overline{x}_1\overline{x}_2$ is a Boolean representation for the function.

We can generalize the reasoning used in this example and make it more explicit. This is based upon the following three simple propositions for Boolean functions, whose proofs are also easy. Furthermore, each of these can be extended to cases where more than two variables are involved (see Exercises 14b and 15b).

PROPOSITION* 7.5 - 1: *Acceptance for Complements

$\overline{x} = 1$ *iff* $x = 0$.

***Proof*:**

See Exercise 13. ■

PROPOSITION* 7.5 - 2: *Acceptance for Products

$xy = 1$ *iff* $x = 1$ *and* $y = 1$.

***Proof*:**

See Exercise 14a. ■

While these two propositions hold for any Boolean algebraic structure, being simple theorems of *Boolean Algebra*, the next proposition requires the variables to range over the values 0 and 1; the forward direction is not true for a general Boolean algebra (see Exercise 15c).

PROPOSITION* 7.5 - 3: *Acceptance for Sums

For w and z in $\mathcal{B} = \{0, 1\}$, $w + z = 1$ iff $w = 1$ or $z = 1$.

***Proof*:**

See Exercise 15a. ■

The descriptive names given to the above propositions use the notion of acceptance (yielding output 1); this proves to be a good way to think about how to generate a Boolean expression for a Boolean function based on its table. We will formalize the procedure of the last example after first stating and illustrating the relevant definitions.

DEFINITION* 7.5 - 2: *Literal, Minterm, Minterm Expansion

a) *A **literal** is a Boolean variable or its complement.*

b) *A **minterm** in the Boolean variables $x_1, x_2, \ldots, x_n$ is an n-fold product $m_1 m_2 \cdots m_n$, where for each i either $m_i = x_i$ or $m_i = \overline{x}_i$.*

b) *A **minterm expansion** in the variables $x_1, x_2, \ldots, x_n$ is a sum of distinct minterms.*

A minterm is thus a fairly simple Boolean expression, being the product of n literals. While this concept assumes the context of *Boolean Algebra*, the terminology draws upon its connection with Boolean lattices. Converting a Boolean algebra into a lattice in the conventional way turns products like minterms into meets, which lie below all the factors involved, being their greatest lower bound. Hence the *min* of *minterm*.

✠ **EXAMPLE 7.5 - 6**

Are $x_1\overline{x}_2x_3$ and x_1x_3 minterms in the variables x_1, x_2, x_3?

Solution

Minterms must have a literal for each variable, either the variable itself or its complement. Thus, $x_1\overline{x}_2x_3$ is a minterm in x_1, x_2, x_3, while x_1x_3 is not because it has no factor associated with x_2.

PROPOSITION* 7.5 - 4: *Acceptance and Uniqueness of Minterm Expansions

Boolean functions defined by minterm expansions accept as many strings as they have terms. Furthermore, Boolean functions defined by distinct minterm expansions are distinct.

***Proof*:**

Let $m_1 m_2 \cdots m_n$ be a minterm in the variables $x_1, x_2, \ldots, x_n$. Then, by *Propositions* 1 and 2, the function defined by this minterm accepts exactly one input string $(s_1, s_2, \ldots, s_n)$, the one in which $s_i = 1$ if $m_i = x_i$ and $s_i = 0$ if $m_i = \overline{x}_i$. It should be clear from this association that functions defined by different minterms accept different input strings.

By *Proposition* 3, functions defined by an expansion containing k minterms will accept k input strings, the ones associated with each of its minterms.

Thus, distinct minterm expansions define distinct Boolean functions. ■

Proposition 4 gives us the necessary groundwork for proving the following central theorem for Boolean functions.

THEOREM* 7.5 - 1: *Minterm Representation for Boolean Functions

Every non-zero Boolean function can be uniquely represented by a minterm expansion, the one that encodes the acceptance information in its function table.

***Proof*:**

Let $f(x_1, x_2, \ldots, x_n)$ be a non-zero Boolean function.

For each input row having output 1, form the minterm that accepts that input string. Sum these minterms up to form a minterm expansion $E(x_1, x_2, \ldots, x_n)$.

By *Proposition* 4, the function defined by this minterm expansion accepts exactly the same input strings as the given function f. $E(x_1, x_2, \ldots, x_n)$ thus encodes f's function table and provides a representation for it: $f(x_1, x_2, \ldots, x_n) = E(x_1, x_2, \ldots, x_n)$.
Also by *Proposition* 4, this minterm-expansion representation must be unique. ■

✠ **EXAMPLE 7.5 - 7**

Determine the minterm expansion for the ternary majority function, whose table is given below.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	Minterm
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{x}_1 x_2 x_3$
1	0	0	0	
1	0	1	1	$x_1 \overline{x}_2 x_3$
1	1	0	1	$x_1 x_2 \overline{x}_3$
1	1	1	1	$x_1 x_2 x_3$

Solution

The minterms associated with the strings accepted by f are (from the bottom up) $x_1x_2x_3$, $x_1x_2\overline{x}_3$, $x_1\overline{x}_2x_3$, and $\overline{x}_1x_2x_3$, as indicated. Adding these gives the minterm expansion representing the ternary majority function: $f(x_1, x_2, x_3) = x_1x_2x_3 + x_1x_2\overline{x}_3 + x_1\overline{x}_2x_3 + \overline{x}_1x_2x_3$.

Minterm Expansions for Boolean Expressions

We now have a uniform way to generate a minterm expansion for a Boolean function: write out its Boolean function table and then use it to generate the minterms to need to be added together.

This process works, but it may not be optimal, particularly if the function is already given in terms of a Boolean expression. So we will exhibit another method that can be used in this situation. This second procedure also enables us to show that each Boolean *expression* can be uniquely represented by (is identical to) a minterm expansion. We will demonstrate this first in connection with the ternary majority function just considered.

✠ **EXAMPLE 7.5 - 8**

Determine the minterm expansion for $f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3$, the ternary majority function.

Solution

We earlier saw that the ternary majority function is, in fact, formulated by the expression $x_1x_2 + x_1x_3 + x_2x_3$ since its output is 1 when any two inputs are 1 (see Example 7.4-4). This is not yet a minterm expansion, though, since each term lacks one literal factor.
We will correct this by first multiplying each term by 1 in the form $x_i + \overline{x}_i$ for the missing literal and then deleting any duplicate minterms.

$$\begin{aligned} x_1x_2 + x_1x_3 + x_2x_3 &= x_1x_2(x_3 + \overline{x}_3) + x_1x_3(x_2 + \overline{x}_2) + x_2x_3(x_1 + \overline{x}_1) \\ &= x_1x_2x_3 + x_1x_2\overline{x}_3 + x_1x_2x_3 + x_1\overline{x}_2x_3 + x_1x_2x_3 + \overline{x}_1x_2x_3 \\ &= x_1x_2x_3 + x_1x_2\overline{x}_3 + x_1\overline{x}_2x_3 + \overline{x}_1x_2x_3 \end{aligned}$$

Checking, this is the same result we just found in Example 7.

The procedure used in Example 8 gives us a way to find minterm expansions for Boolean expressions in general. Before we prove this, we should first remark on how this ties in with *Theorem* 1, which asserts the same thing for Boolean functions. Since any Boolean expression defines a Boolean function, and since a Boolean function can be uniquely represented by a minterm expansion, won't this expansion represent the original Boolean expression? Yes, but not immediately; this needs to be argued. So far we only know they agree on the values 0 and 1, not that they must agree for all possible values in any Boolean algebra. The next theorem extends the agreement, giving us a genuine Boolean identity. To prove the theorem, we first separate out a lemma worth stating in its own right and illustrate it with an example.

LEMMA*: *Sum-of-Products Decomposition

Every Boolean expression not equal to 0 *in the variables* $x_1, x_2, \ldots, x_n$ *can be written as a sum of products of distinct literals in these variables.*

***Proof*:**

We do this using an inductive argument of sorts, based upon the recursive definition of Boolean expressions.

* First note that if an expression contains a 1, it can be replaced by $x_1 + \overline{x}_1$. Furthermore, any 0s can be eliminated by doing the indicated operation on them and simplifying. So we only need to consider expressions containing literals in the given variables.
* For our anchor step, then, note that all variables x_i are already in the required form.
* For our induction hypothesis, suppose that expressions E and F can be written as sums of products of distinct literals in the given variables.
  - Then the sum $E + F$ can as well: convert each part separately and add them together. Eliminate any duplicate terms by rearranging the sum and using the *Idempotence Law*.
  - The product $E \cdot F$ can be written as the product of such sums. Using the *Distributive Law*, as needed, multiply this out. The resulting sum may contain product terms with duplicate variables involved or the product of a variable with its complement. These can be simplified using the *Idempotence Laws* and the *Complementation Law* for products, reducing terms with duplicate factors and dropping 0-terms. Duplicate terms being added can again be eliminated. Further simplification may be achieved using some other law of *Boolean Algebra*, such as an *Absorption Law*.
  - The complement $\overline{E}$ can be written either as the complement of a product (if only one term is present in E's sum-of-products representation), which yields what is needed, or as the complement of a sum of such products, which becomes the product of sums of complements and variables (via *De Morgan's Laws*). By what we said about sums and products, this reduces to the form needed.
* The final result of such a process (though not unique) will be a non-zero sum of products of distinct literals in the variables $x_1, x_2, \ldots, x_n$. ■

✠ **EXAMPLE 7.5 - 9**

Determine a sum-of-products decomposition for the Boolean expression $(x+\overline{y}z)^2+y(\overline{yz})+z$.

Solution

Following the method of the theorem's proof, we have the following sequence of equivalent expressions. Both of the last two lines give a solution.

$$\begin{aligned}
(x+\overline{y}z)^2 + y(\overline{yz}) + z &= x + \overline{y}z + y(\overline{y} + \overline{z}) + z && \text{Idem, DeM}\\
&= x + \overline{y}z + y\overline{y} + y\overline{z} + z && \text{Distrib}\\
&= x + \overline{y}z + y\overline{z} + z && \text{Compl}\\
&= x + y\overline{z} + z && \text{Absorp}
\end{aligned}$$

THEOREM* 7.5 - 2: *Unique Minterm Representation for Boolean Expressions

Every non-zero Boolean expression in the variables $x_1, x_2, \ldots, x_n$ can be uniquely represented as a minterm expansion.

***Proof*:**

From the Lemma, every non-zero Boolean expression can be written as a sum of products of distinct literals. Take each of these product terms in turn and multiply it by $x_i + \overline{x}_i$ if neither x_i nor $\overline{x}_i$ is present. This yields an expression equal to the original one since we are merely multiplying each product by 1.

The resulting expansion is a sum of minterms. Dropping all duplicate minterms, we end up with a minterm expansion that represents/is equal to the original Boolean expression.

Uniqueness follows from what we've already done: since there is only one minterm expansion agreeing with the expression on 0 and 1, that must be the one we've found here. The resulting minterm expansion is the same, regardless of which sum-of-products decomposition is used to represent the original Boolean expresssion. ■

Minterm Expansions and Boolean Algebra

The *Minterm Representation Theorem for Boolean Expressions* has a very interesting beneficial consequence for the theory of *Boolean Algebra*. It provides a justification for a practice adopted by some discrete mathematics textbooks and many writers on *Boolean Algebra*, though usually without commentary or justification: *truth tables suffice to prove the identity theorems of Boolean Algebra*. In other words, showing that two expressions agree when the variables are 0 or 1 is enough to show that these expressions are identical. The converse (*identical expressions share the same truth tables*) is rather obvious, but this new result is somewhat unexpected and may even seem questionable. A Boolean algebra typically has many more elements than just 0 and 1. Why should the agreement of two Boolean expressions on all 0 - 1 input combinations warrant their universal agreement for any input string from any Boolean algebra? The *Minterm Representation Theorem for Boolean Expressions* is the key to showing this rather surprising result. We will prove this as one direction of a corollary to that theorem.

COROLLARY*: *Truth-Table Verification of Boolean Identities

If $E(x_1, x_2, \ldots, x_n)$ and $F(x_1, x_2, \ldots, x_n)$ are Boolean expressions in variables $x_1, \ldots, x_n$, $E(x_1, x_2, \ldots, x_n) = F(x_1, x_2, \ldots, x_n)$ iff their truth tables are identical.

***Proof*:**

- Suppose first that $E(x_1, x_2, \ldots, x_n) = F(x_1, x_2, \ldots, x_n)$ for all values of x_i. Then since every Boolean algebra contains identity elements 0 and 1, these values can replace the x_i in any input combination. The identity of the expressions forces the outputs to agree. Thus, their truth tables will be identical.
- Conversely, suppose that the truth tables for $E(x_1, x_2, \ldots, x_n)$ and $F(x_1, x_2, \ldots, x_n)$ are identical. Then the minterm expansions associated with these tables are likewise equal. By *Theorem* 2, these minterm expansions must represent the given expressions, so the expressions themselves are also equal: $E(x_1, x_2, \ldots, x_n) = F(x_1, x_2, \ldots, x_n)$. ■

✠ **EXAMPLE 7.5 - 10**

Show that the *Consensus Law* (*Proposition* 7.3-8) is a theorem of *Boolean Algebra*:

$$xy + \overline{x}z + yz = xy + \overline{x}z.$$

Solution

The following truth table proves the claim. The last two columns agree; hence by the above corollary, the two expressions heading up those columns are equal for all x, y, and z.

In other words, the *identity* $xy + \overline{x}z + yz = xy + \overline{x}z$ is a theorem of *Boolean Algebra*.

x	y	z	$xy + \overline{x}z + yz$	$xy + \overline{x}z$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

Of course, checking Boolean identities via truth tables gets tedious by hand if three or more variables are involved, and if too many variables are involved, it becomes time consuming even for a computer check. But occasionally a shortcut is possible. That is the case here. Note that the two expressions differ only by the term yz. If $yz = 0$, then the two expressions are the same. On the other hand, if $yz = 1$, then both y and z must be 1. This amounts to having to check only the fourth and eighth rows in the above table, which is less work than filling out the entire table. Alternatively, when y and z are both 1, the two expressions reduce to $x + \overline{x} + 1$ and $x + \overline{x}$, both of which are 1, so they agree for these values, too.

The truth-table approach to justifying theorems in *Boolean Algebra* makes its set of identities decidable: it provides a mechanical procedure for determining whether an equation is true. It also makes "proving" certain theorems independent of creatively combining the axioms of *Boolean Algebra* in a sequential reasoning process to deduce a theorem. The *Corollary* to *Theorem* 2 is thus important from a mathematical point of view, but it does show that the elementary theory of *Boolean Algebra* is rather uninteresting from a proof-theoretic standpoint. Nevertheless, we did draw upon some significant mathematical results about Boolean expressions to justify the truth-table approach to developing *Boolean Algebra*, a connection many textbooks fail to spell out even when they adopt it to simplify establishing theorems.

Because of its relative simplicity and the ease of checking the truth of an identity, *Boolean Algebra* has been the focus of a fair bit of research by people interested in computerized theorem proving. Besides checking for identities in the system as we have set it up, researchers have explored using other operations, such as the NAND operator, developing and testing simple axiom systems for *Boolean Algebra* based upon those operators. Relatively simple axiom systems containing only two or even one axiom have been found using automated reasoning programs. Further information on this can be found in the literature on the topic.

Normal-Form Representation in Sentential Logic

The *Minterm Representation Theorem for Boolean Functions* yields a noteworthy consequence for *Sentential Logic*. There minterm expansions (canonical sums of products) correspond to what are called *normal disjunctive forms*.

DEFINITION* 7.5 - 3: *Normal-Disjunctive Form

A ***normal disjunctive form*** *in the variables* $P_1, P_2, \ldots, P_n$ *is a disjunction of distinct conjuncts* $M_1 \wedge M_2 \wedge \cdots \wedge M_n$*, where* M_i *is either* P_i *or* $\neg P_i$*.*

COROLLARY*: *Normal Disjunctive Form Representation

Every formula of Sentential Logic, regardless of the truth-functional connectives it involves, is logically equivalent to one in normal disjunctive form.

***Proof*:**

This is a direct application of the *Minterm Representation Theorem for Boolean Functions* to the case of *Sentential Logic*. Hence we have labeled it a corollary. ■

✠ **EXAMPLE 7.5 - 11**

Find a normal disjunctive form for $P \veebar Q$.

Solution

This is the XOR connective. We can express it using $\neg$, $\wedge$, and $\vee$ in the following way: $P \veebar Q \models\!\!\!\dashv (P \vee Q) \wedge \neg(P \wedge Q)$. This is not what's needed here, though it does show how XOR can be expressed using just these three more primitive connectives.

A second equivalent form is this: $P \veebar Q \models\!\!\!\dashv (P \wedge \neg Q) \vee (\neg P \wedge Q)$. This gives the normal disjunctive form.

If this latter form were overlooked, it could be generated from the first form by means of a series of SL equivalences. We will leave this as an exercise (see Exercise 37).

The normal disjunctive form for a *Sentential Logic* formula can be generated in a uniform way from the formula's truth table, much as we obtained a minterm expression for a Boolean function. Alternatively, if we already have an expression involving these basic connectives, we can use a procedure similar to generating a minterm expansion for a Boolean expression: conjoin each disjunct with $X \vee \neg X$ for a missing sentence variable X and proceed to equivalent forms. Occasionally this maneuver will be unnecessary, but not always.

That every formula of SL has a normal disjunctive form means that *every possible logical connective*, whether unary, binary, or n-ary, can be represented by a logically equivalent formula in the same sentence variables involving only negation, conjunction, and disjunction. Thus, the connectives $\neg$, $\wedge$, and $\vee$ form an *expressively complete* set of connectives. Anything anyone can say by means of a truth-functional connective of any complexity whatever can be said (in an equivalent form) using only $\neg$, $\wedge$, and $\vee$.

The expressive completeness of $\neg$, $\wedge$, and $\vee$ may not seem so surprising to you at this point, but it often strikes people as quite amazing when they first hear of it in the context of learning logic (see Section 2.1). Drawing upon our knowledge of logic (or *Boolean Algebra*), however, we can do even better. *De Morgan's Laws* warrant dropping either $\wedge$ or $\vee$ without losing expressive power, for $P \wedge Q \models\!\!\!\dashv \neg(\neg P \vee \neg Q)$ and $P \vee Q \models\!\!\!\dashv \neg(\neg P \wedge \neg Q)$. So two logical connectives suffice: take either $\neg$ and $\wedge$, or $\neg$ and $\vee$.

Having reduced the size of the number of required connectives from three to two, an obvious follow-up question suggests itself: does any single connective suffice? The answer to this, first discovered by the nineteenth-century American logician and mathematician C. S. Peirce, is "yes". The first published result along these lines, however, was due to Henry Sheffer in 1913. Neither $\wedge$ nor $\vee$ suffice because they fail at inversion, but combining either of these with $\neg$ does work. The Sheffer stroke, better known now as the NAND connective, suffices (see Exercise 1.3-31), as does the NOR connective (see Exercise 1.3-32). Naturally, restricting yourself to one connective, or even two or three, for that matter, makes logical life much more complicated for some tasks, though it simplifies it for others. The expressive completeness of the NAND connective, for instance, is what makes it possible to build logic circuits solely from NAND gates (see Exercises 7.4-41 – 7.4-43). It is also what allows people to use it as the sole operator for certain axiomatizations of *Boolean Algebra*.

The Principle of Duality in Boolean Algebra; Maxterm Expansions

In discussing lattices we noticed a certain symmetry or duality involving meets and joins. Properties that held for one held for the other as well (see the propositions in Section 7.2).

This duality was further developed by introducing the converse relation, which inverted the lattice (see Exercise 7.2-12); there meets turned into joins and conversely.

Boolean Algebra inherits this same duality. Properties true for addition are also true for multiplication and vice versa, though 0 and 1 must be interchanged as well if they are present (this can be thought of as related to lattice inversion: see Exercises 7.3-28). This phenomenon occurs because all the axioms of *Boolean Algebra* are present in dual forms. The dual of any proposition that follows from the axioms can thus be deduced from the axioms' duals (see Exercise 38).

This duality can be exploited here as well. Corresponding to minterms there are maxterms, which give rise to maxterm-expansion representations for Boolean functions and Boolean expressions. These are occasionally useful, as we will see, even when we want a minterm expansion. We will first state the relevant definition and illustrate its use before proving the obvious representation theorem.

DEFINITION* 7.5 - 4: *Maxterm, Maxterm Expansion

Let $\mathcal{B}$ be the standard Boolean algebra on $\{0,1\}$ and let $x_1, x_2, \ldots, x_n$ be defined on $\mathcal{B}$.

a) *A* ***maxterm*** *in the variables $x_1, x_2, \ldots, x_n$ is an n-fold sum $m_1 + m_2 + \cdots + m_n$, where for each i either $m_i = x_i$ or $m_i = \overline{x}_i$.*

b) *A* ***maxterm expansion*** *in the variables $x_1, x_2, \ldots, x_n$ is a product of distinct maxterms.*

The duals of the earlier acceptance propositions become rejection propositions. Note here again that *Propositions* 5 and 6 hold for *Boolean Algebra* in general, but the forward direction of *Proposition* 7 only holds for the algebra of 0 and 1.

PROPOSITION* 7.5 - 5: *Rejection for Complements

$\overline{x} = 0$ iff $x = 1$.

***Proof*:**

See Exercise 16. ■

PROPOSITION* 7.5 - 6: *Rejection for Sums

$x + y = 0$ iff $x = 0$ and $y = 0$.

***Proof*:**

See Exercise 17a. ■

PROPOSITION* 7.5 - 7: *Rejection for Products

For w and z in $\mathcal{B} = \{0,1\}$, $wz = 0$ iff $w = 0$ or $z = 0$.

***Proof*:**

See Exercise 18a. ■

✠ **EXAMPLE 7.5 - 12**

Determine the maxterm expansion for the Boolean function given by the following table.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	Maxterm
0	0	0	1	
0	0	1	0	$x_1 + x_2 + \overline{x}_3$
0	1	0	1	
0	1	1	1	
1	0	0	1	
1	0	1	1	
1	1	0	1	
1	1	1	0	$\overline{x}_1 + \overline{x}_2 + \overline{x}_3$

Solution

- For a maxterm expansion, we concentrate on the rows producing 0, using the dual process to that for minterms, this time starting from the top of the table (so our final result will again list variables before their complements): *add the input variables whose values are* 0 *to the complements of the input variables whose values are* 1*; then multiply the resulting sums together.*
  Doing this here gives $(x_1 + x_2 + \overline{x}_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3)$ as our maxterm expansion.
- We can determine the maxterm expansion in an alternative way as well.
  We begin by finding the *minterm expansion* for the *complement* $\overline{f}$ of the function f, where $\overline{f}(x_1, x_2, \ldots, x_n) = 1$ iff $f(x_1, x_2, \ldots, x_n) = 0$.
  This results in $x_1x_2x_3 + \overline{x}_1\overline{x}_2x_3$. Taking *its complement* gives the maxterm expansion for f:
  $f(x_1, x_2, x_3) = \overline{x_1x_2x_3 + \overline{x}_1\overline{x}_2x_3} = (\overline{x}_1+\overline{x}_2+\overline{x}_3)(x_1+x_2+\overline{x}_3) = (x_1+x_2+\overline{x}_3)(\overline{x}_1+\overline{x}_2+\overline{x}_3)$.

We can generalize either of the two methods used in the last example to prove the *Maxterm Representation Theorem*. The first method employs the dual argument of what we used earlier; we will leave this as an exercise (see Exercise 27). Instead, for variety, we will use a generalized version of the second method, proving our result via the earlier *Minterm Representation Theorem* and double complementation.

THEOREM* 7.5 - 3: *Maxterm Representation Theorem for Boolean Functions

Every Boolean function can be uniquely represented by a maxterm expansion.

***Proof*:**

Let f be a Boolean function in the variables $x_1, x_2, \ldots, x_n$, and let $\overline{f}$ denote its inversion: $\overline{f}(x_1, x_2, \ldots, x_n) = 1$ iff $f(x_1, x_2, \ldots, x_n) = 0$.
As a Boolean function in its own right, $\overline{f}$ has a minterm expansion $E(x_1, x_2, \ldots, x_n)$.
$\overline{E(x_1, x_2, \ldots, x_n)}$ will be the maxterm expansion for f, for complements of minterm expansions are maxterm expansions (see Exercise 24a) and the complement of a representation for $\overline{f}$ is a representation for f (see Exercise 26a).
Uniqueness of the maxterm expansion follows from the uniqueness of the minterm expansion. ■

Like the *Minterm Representation Theorem*, the *Maxterm Representation Theorem* has consequences for *Sentential Logic*, similar to what we saw earlier.

DEFINITION* 7.5 - 5: *Normal Conjunctive Form

A ***normal conjunctive form*** *in the variables $P_1, P_2, \ldots, P_n$ is a conjunction of distinct disjuncts $M_1 \vee M_2 \vee \cdots \vee M_n$, where M_i is either P_i or $\neg P_i$.*

COROLLARY*: *Normal Conjunctive Form for Sentential Logic

Every formula of Sentential Logic, regardless of the truth-functional connectives it involves, is logically equivalent to one in normal conjunctive form.

***Proof*:**

This follows immediately from the last theorem. ■

✠ **EXAMPLE 7.5 - 13**

Find the normal conjunctive form for $P \veebar Q$.

Solution

The table for $P \veebar Q$ is given by the following.

P	Q	$P \veebar Q$
0	0	0
0	1	1
1	0	1
1	1	0

The normal conjunctive form, therefore, is $(P \vee Q) \wedge (\neg P \vee \neg Q)$

EXERCISE SET 7.5

Problems 1 - 4: Boolean Functions and Boolean Expressions

The following problems explore the existence of particular Boolean functions.

1. *Example* 1
   Explain in your own words why $f(x) = x$ and $g(x) = x^2$ are identical as Boolean functions from $\mathcal{B}$ to $\mathcal{B}$, even though they are defined by different Boolean expressions.

2. *Example* 2
   a. Explain why there are four functions from $\mathcal{B}$ to $\mathcal{B}$, and verify that those given in Example 2 are all distinct and so must represent them.
   b. Which of the four functions in Example 2 are one-to-one functions? onto functions? bijections?

3. *Boolean Functions of Several Variables*
   a. Calculate the number of distinct functions from $\mathcal{B}^2$ to $\mathcal{B}$.
   b. How many of the functions in part *a* are one-to-one? onto? one-to-one and onto?
   c. Determine Boolean expressions $F(x_1, x_2) =$ _____ for all the distinct Boolean functions $f : \mathcal{B}^2 \to \mathcal{B}$.
   d. Calculate the number of distinct functions from $\mathcal{B}^n$ to $\mathcal{B}$.
   e. How many of the functions in part *d* are one-to-one? onto? one-to-one and onto? Assume that $n \geq 2$.
   f. Determine Boolean expressions $F(x_1, x_2, \ldots, x_n) =$ _____ for all the distinct Boolean functions $f : \mathcal{B}^n \to \mathcal{B}$.

*4. *Simplified Reduced Boolean Expressions*
   For each of the following Boolean expressions find an equivalent simplified reduced Boolean expression (a sum-of-products of literals).
   *a. $x^4 + x^3\overline{x} + \overline{x}^2$
   b. $x^{12} + \overline{x}^6(x^4\overline{x} + x)^5(x^2 + \overline{x}^3)^8$
   c. $(x^2 + xy^3 + \overline{x}\,y^2)\,\overline{x}^5\,\overline{y}^3$
   *d. $(x^3y^2 + \overline{x}\,y^2)\,(\overline{x^2 + x\,\overline{y}^3})$
   *e. $(x^4\overline{y}^2z + \overline{z})^2xyz + \overline{x}\,\overline{y}^2\,\overline{z}$
   f. $(x^4y + \overline{x}\,\overline{y}\,\overline{z}^5)(x^3y^2z + \overline{y}^2z^3) + (\overline{x}\,\overline{y}\,\overline{z} + \overline{x}^2x)(xy + z)^3$

Problems 5 - 8: Algebra of Boolean Functions

The following problems consider algebraic operations on Boolean functions, defined pointwise as follows.

Definition: *Let f and g be functions from $\mathcal{B}^n$ to $\mathcal{B}$. Define $f + g$, $f \cdot g$, and $\overline{f}$ by*

a) $(f + g)(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n) + g(x_1, x_2, \ldots, x_n)$
b) $(f \cdot g)(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n) \cdot g(x_1, x_2, \ldots, x_n)$
c) $\overline{f}\,(x_1, x_2, \ldots, x_n) = \overline{f\,(x_1, x_2, \ldots, x_n)}$

5. Explain why the above definition establishes $f + g$, $f \cdot g$, and $\overline{f}$ as Boolean functions from $\mathcal{B}^n$ to $\mathcal{B}$.

6. Prove that if f is given by $f(x_1, x_2, \ldots, x_n) = E(x_1, x_2, \ldots, x_n)$, where $E(x_1, x_2, \ldots, x_n)$ is a Boolean expression, then $\overline{f}(x_1, x_2, \ldots, x_n) = \overline{E(x_1, x_2, \ldots, x_n)}$.
7. Prove that the function $\overline{f}$ satisfies the following conditions, where 0 and 1 denote the constant functions with those values.
   a. $f + \overline{f} = 1$
   b. $f \cdot \overline{f} = 0$
   c. $\overline{\overline{f}} = f$
   d. Prove or disprove: $\overline{f}(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n) = f(x_1, x_2, \ldots, x_n)$.
8. Does the set of all Boolean functions $f : \mathcal{B}^n \to \mathcal{B}$ form a Boolean algebra under the pointwise function operations of addition, multiplication, and complementation defined above? Check whether all of the axioms of *Boolean Algebra* are satisfied. Explain your result.

Problems 9 - 12: True or False

Are the following statements true or false? Explain your answer.

9. A sum of Boolean expressions has value 1 iff the value of every component expression is 1.

*10. The formula $x_1 + x_2$ is a Boolean expression in the variables x_1, x_2, x_3.

*11. Truth tables can be used to check whether set-theoretic identities involving $\cup$ and $\cap$ hold true.

12. Any result that holds true of the Boolean algebra $\mathcal{B} = \{0, 1\}$ also holds true of all Boolean algebras. *Hint*: consider the statement $(\forall x)(x = 0 \vee x = 1)$. Does your answer contradict *Theorem* 7.5-2's *Corollary on Verifying Boolean Identities*? Explain.

Problems 13 - 18: Simple Propositions for Boolean Functions

Prove the following propositions, using what you know about Boolean Algebra and Boolean functions.

13. *Proposition* 1: *Acceptance for Complements*
    $\overline{x} = 1$ iff $x = 0$.

*14. *Proposition* 2: *Acceptance for Products*
    *a. $xy = 1$ iff $x = 1$ and $y = 1$.
    b. Generalize this proposition to the case of n variables and prove it using *Mathematical Induction*.

*15. *Proposition* 3: *Acceptance for Sums*
    *a. For w and z in $\mathcal{B} = \{0, 1\}$, $w + z = 1$ iff $w = 1$ or $z = 1$.
    b. Generalize this proposition to the case of n variables and prove it using *Mathematical Induction*.
    *c. Explain where and why this proposition fails for Boolean algebras in general. Hint: Consider $\mathcal{B}^2$.

16. *Proposition* 5: *Rejection for Complements*
    $\overline{x} = 0$ iff $x = 1$.

17. *Proposition* 6: *Rejection for Sums*
    a. $x + y = 0$ iff $x = 0$ and $y = 0$.
    b. Generalize this proposition to the case of n variables and prove it using *Mathematical Induction*.

18. *Proposition* 7: *Rejection for Products*
    a. For w and z in $\mathcal{B} = \{0, 1\}$, $wz = 0$ iff $w = 0$ or $z = 0$.
    b. Generalize this proposition to the case of n variables and prove it using *Mathematical Induction*.
    c. Explain where and why this proposition fails for Boolean algebras in general. Hint: Consider $\mathcal{B}^2$.

Problems 19 - 27: Minterm and Maxterm Expansions

The following problems relate to minterm and maxterm expansions for Boolean functions.

*19. Develop minterm expansions for the following Boolean expressions.
    a. $x + y$
    b. xy
    *c. $x + \overline{x}y$
    d. $x + \overline{y} + z$

e. $x(z + \overline{y}\,\overline{z}) + \overline{x}z$

f. Do you think 0 should be considered a minterm expansion (the null sum)? Why or why not?

20. Develop minterm expansions for Boolean functions that accept the following input strings.

a. $00, 10$

b. $01, 10, 11$

c. $010, 011, 101$

d. $000, 001, 100, 101$

21. *Counting Minterm Expansions*

a. How many distinct minterms in the variables $x_1, x_2, \ldots, x_n$ are there? Justify your answer.

b. Let $F(x_1, x_2, \ldots, x_n)$ denote the full minterm expansion for the constantly 1 function. How many minterms does F contain? Explain.

c. How many distinct minterm expansions (sums of minterms) in the variables $x_1, x_2, \ldots, x_n$ are there? Justify your answer.

*22. *Minterms and Boolean Function Values*

*a. How many distinct input strings are accepted by a minterm? Explain.

*b. How many distinct input strings are accepted by a minterm expansion? Explain.

*c. Explain why distinct minterm expansions represent different Boolean functions.

*23. *Minterm Expansions for Complements of Functions*

Knowing how the tables for a function and its complement are related, find the minterm expansion for $\overline{f}$, where f is given by the following Boolean expression.

a. $xy + \overline{x}\,y + \overline{x}\,\overline{y}$

*b. $xyz + x\,\overline{y}\,z + \overline{x}\,y\,\overline{z} + \overline{x}\,\overline{y}\,\overline{z}$

c. Explain how to find the minterm expansion for the complement $\overline{f}$ of a function f whose minterm expansion is given.

24. *Maxterm Expansions from Minterm Expansions*

a. Prove if $E(x_1, x_2, \ldots, x_n)$ is the minterm expansion for a Boolean function f, then $\overline{E(x_1, x_2, \ldots, x_n)}$ is the maxterm expansion for its complement function $\overline{f}$.

b. Use part *a* and Problem 23a to find the maxterm expansion for the function f given by $f(x, y) = xy + \overline{x}\,y + \overline{x}\,\overline{y}$.

*c. Use part *a* and Problem 23b to find the maxterm expansion for the function f given by $f(x, y, z) = xyz + x\,\overline{y}\,z + \overline{x}\,y\,\overline{z} + \overline{x}\,\overline{y}\,\overline{z}$.

25. *Maxterm Expansions for Complements of Functions*

Knowing how the tables for a function and its complement are related, find the maxterm expansion for $\overline{f}$, where f is given by the following Boolean expression.

a. $(x + \overline{y})(\overline{x} + y)$

b. $(x + y + \overline{z})(x + \overline{y} + z)(\overline{x} + \overline{y} + z)(\overline{x} + \overline{y} + \overline{z})$

c. Explain how to find the maxterm expansion for the complement $\overline{f}$ of a function f whose maxterm expansion is given.

26. *Minterm Expansions from Maxterm Expansions*

a. Prove that if $E(x_1, x_2, \cdots, x_n)$ is a maxterm expansion for a Boolean function f, then $\overline{E(x_1, x_2, \cdots, x_n)}$ is a minterm expansion for the function $\overline{f}$.

b. Use part *a* and Problem 25a to find the minterm expansion for the function f given by $f(x, y) = (x + \overline{y})(\overline{x} + y)$.

c. Use part *a* and Problem 25b to find the minterm expansion for the function f given by $f(x, y, z) = (x + y + \overline{z})(x + \overline{y} + z)(\overline{x} + \overline{y} + z)(\overline{x} + \overline{y} + \overline{z})$.

27. *Theorem* 3: *Maxterm Representation Theorem for Boolean Functions*

Prove that *every Boolean function can be uniquely represented by a maxterm expansion* by generalizing the first approach in Example 12; that is, use the dual argument of the proof for the *Minterm Representation Theorem.*

Problems 28 - 34: Proving Boolean Identities

Reprove the following theorems of Boolean Algebra, now using the method of truth-table verification.

28. *Proposition* 7.3-2: *Complements of Elements Laws*
    a. $\overline{0} = 1$
    b. $\overline{1} = 0$
    c. $\overline{\overline{x}} = x$

*29. *Proposition* 7.3-3: *Annihilation and Absorption Laws*
    a. $x \cdot 0 = 0$
    *b. $x + 1 = 1$
    *c. $x(x + y) = x$
    d. $x + xy = x$

30. *Proposition* 7.3-4: *Idempotence Laws*
    a. $x \cdot x = x$
    b. $x + x = x$

31. *Proposition* 7.3-5: *De Morgan's Laws*
    a. $\overline{xy} = \overline{x} + \overline{y}$
    b. $\overline{x + y} = \overline{x}\,\overline{y}$

*32. *Proposition* 7.3-7: *Redundancy Laws*
    *a. $x(\overline{x} + y) = xy$
    b. $x + \overline{x}y = x + y$

33. *Proposition* 7.3-8: *Consensus Laws*
    The first Consensus Law has been shown in Example 10.
    Prove the second one: $(x + y)(\overline{x} + z)(y + z) = (x + y)(\overline{x} + z)$

*34. Are the following equations Boolean identities or not? Explain.
    a. $x + x\overline{y} + \overline{x}y = x + \overline{y}$
    b. $x + yz = (x + y)z$
    *c. $x + xy + \overline{x}z = xy + x\overline{y} + z$
    d. $xy + x\,\overline{y}\,z = xy + xz$

Problems 35 - 37: Normal Forms in Sentential Logic

The following problems explore normal forms for sentences of Sentential Logic.

35. Determine equivalent normal disjunctive forms for the following SL connectives.
    a. P NAND Q
    b. P NOR Q
    c. P XNOR Q
    d. $\neg P$

36. Determine equivalent normal disjunctive forms for the following SL formulas.
    a. $P \wedge \neg(P \wedge Q)$
    b. $P \leftrightarrow (Q \leftrightarrow P)$
    c. $P \rightarrow (Q \wedge \neg R)$
    d. $P \vee (\neg Q \rightarrow P \wedge R)$

37. Beginning with the first equivalence of Example 10, $P \veebar Q \models\!\dashv (P \vee Q) \wedge \neg(P \wedge Q)$, transform this via various *Replacement Rules* and *Contraction Rules* (see Exercises 1.9-14 – 1.9-17) to show that $P \veebar Q \models\!\dashv (P \wedge \neg Q) \vee (\neg P \wedge Q)$, which gives the normal disjunctive form for the exclusive-or connective.

38. *Duality Principles for Boolean Algebra*

a. Show that interchanging x and $\overline{x}$ in each axiom of *Boolean Algebra* converts it into another axiom of *Boolean Algebra*.

b. Is the following *Duality Principle* legitimate? Interchanging $+$ and $\cdot$ and 0 and 1 transforms a true statement in *Boolean Algebra* into another true statement. Explain.

c. Is the following *Joint Duality Principle* legitimate? Interchanging x and $\overline{x}$, $+$ and $\cdot$, and 0 and 1 transforms a true statement in *Boolean Algebra* into another true statement. Explain.

d. Is the following transformation legitimized by a *Duality Principle*? Transform $\overline{xy} = \overline{x} + \overline{y}$ by replacing $\overline{xy}$ with xy and $\overline{x} + \overline{y}$ with $x + y$. Comment on what goes wrong here. Does this contradict part *c*?

e. Is the following transformation legitimized by a *Duality Principle*? Transform $\overline{xy} = \overline{x} + \overline{y}$ by replacing $\overline{xy}$ with $x + y$, and $\overline{x} + \overline{y}$ with xy. Comment on what goes wrong now. Does this contradict part *c*?

HINTS TO STARRED EXERCISES 7.5

4. See Example 3.

10. See Definition1.

11. See Example 2 from Section 7.3 and the *Corollary* to *Theorem* 2 from this section.

14. a. One direction is trivial. For the other direction, multiply the given equation by the variables involved (alternatively, multiply by their complements).

15. a. One direction is trivial. For the other direction, keep in mind that this only being asserted for the Boolean algebra $\mathcal{B} = \{0, 1, \}$ (see part c); then prove the statement in its contrapositive form.
    c. See the hint given.

19. c. You might work this in two ways to see which you prefer: you can use t6he method employed in Example 8, or you can write out the truth table for $x + \overline{x}y$ and use the method of Example 7.

22. a. See Example 7.
    b. See Example 7.
    c. Think about how minterm expansions are generated from Boolean function tables and also how such tables distinguish different Boolean functions. See Example 7 again.

23. b. Recall that $\overline{f}$ has output 1 for a string iff f has output 0 for that string.

24. c. Use Problems 23b and 24a, as mentioned in the problem.

29. Check the appropriate truth tables and apply the *Corollary* to *Theorem* 2.

32. Check the appropriate truth tables and apply the *Corollary* to *Theorem* 2.

34. c. Use a truth table to either prove or disprove this equality

7.6 Simplifying Boolean Functions

Boolean functions are of interest to computer scientists and engineers for a number of reasons, the most important from a practical standpoint being their connection with lolgic circuit design. We've already seen how circuits physically realize Boolean functions, and how *Boolean Algebra* can be used to determine equivalent Boolean expressions for a given function. In particular, we know that all Boolean functions/logic circuits can be represented using some standard set of basic Boolean operators/logic gates, and we have two uniform ways to do this (minterm expansions, maxterm expansions). However, these typically do not produce the simplest equivalent form, something which is a prime concern to those who use logic circuits. In this section we will explore ways to in which Boolean functions can be simplified.

✠ **EXAMPLE 7.6 - 1**

Show that both the minterm expansion and the maxterm expansion are more complex than the expression $f(x, y, z) = xy + xz + yz$, which represents the ternary majority function.

Solution

The minterm expansion for this function is $xyz + xy\overline{z} + x\overline{y}z + \overline{x}yz$ (see Example 7.5-7), while its maxterm expansion is $(x+y+z)(x+y+\overline{z})(x+\overline{y}+z)(\overline{x}+y+z)$. Neither of these is as simple as the given expression; both contain more components and more operations.

While the phrase "simplest equivalent form" is not completely unambiguous, such a form should probably minimize the number of terms and operators involved (see Exercises 1-6). In an engineering context, one might take this topic even further and look at ways to reduce the total cost or increase the speed of a circuit.

Ways to Simplify Boolean Functions and Expressions

The laws of *Boolean Algebra* provide us with one way to simplify a Boolean expression: apply the basic identities via substitution to reduce the numbers of terms or operations in the resulting equivalent expressions.

✠ **EXAMPLE 7.6 - 2**

Simplify the ternary majority function, whose minterm expansion is given by $f(x, y, z) = xyz + xy\overline{z} + x\overline{y}z + \overline{x}yz$.

Solution

This is the function of Example 1, picked up from the other end this time. Let's see how we can simplify it.

$$\begin{aligned} xyz + xy\overline{z} + x\overline{y}z + \overline{x}yz &= xyz + xy\overline{z} + xyz + x\overline{y}z + xyz + \overline{x}yz \\ &= xy(z+\overline{z}) + xz(y+\overline{y}) + yz(x+\overline{x}) \\ &= xy + xz + yz \;=\; x(y+z) + yz\,. \end{aligned}$$

Here we duplicated xyz twice (*Idempotence Law*) so it could be combined with each of the other terms, and then we simplified pairs of terms via a *Distributive Law*, a *Complementation Law*, and an *Identity Law*. Our final line gives two possible simplified conclusions.

It should be clear that while *Boolean Algebra* provides the tools for simplifying an expression, you are pretty much on your own for deciding which law to apply. What path to take and when to stop may require imagination and insight on your part; *Boolean Algebra* alone provides no priorities for how to proceed.

Consequently, people have explored more mechanical ways to simplify expressions instead of just applying Boolean identities that look promising. We will look at two such tools that will help us in this regard: Karnaugh Maps, and the Quine-McCluskey method.

Karnaugh Maps for Two-Variable Boolean Functions

Karnaugh maps, called K-maps for short, are due to Maurice Karnaugh, who introduced them in 1953 while working at Bell Labs. They provide a nice visual method for simplifying Boolean functions that have a small number of variables. We'll begin by looking at their application to Boolean functions of two variables; then we will extend it to functions with three or four variables.

The basic idea behind K-maps is to treat Boolean functions as compound operators; a K-map is based on a modified function table, which shows outputs for all input strings.

✠ **EXAMPLE 7.6 - 3**

Exhibit the K-map for the Boolean function $f(x, y) = x + \overline{x}y$.

Solution

Treating f as a Boolean operator that takes input pairs of 0s and 1s, we obtain the first table below: we place x-inputs on the side, y-inputs on the top, and the calculated outputs inside the table's cells. This is merely a condensed version of the function table for f.

$x \backslash y$	0	1
0	0	1
1	1	1

Function Table

	$\overline{y}$	y
$\overline{x}$		1
x	1	1

K-Map

The K-map on the right is a slight variation of f's operation table. It treats output cells as minterms associated with the products. We place a 1 in the cell or block if it is present in the formula and leave it blank otherwise. Here we put a 1 all along the x-row since it is a term, and we put another 1 in the $\overline{x}y$ cell since it is also present.* It is also customary to use 0s and 1s along the outside, as in the function table, to represent variables and their complements. We will do this as well (see below), but at times we will use the letters themselves to avoid confusion over which letter is being represented where.

Karnaugh Maps for Three-Variable Boolean Functions

To represent Boolean functions of three variables, it would be nice to use a three-dimensional cube, using one dimension for each input variable. Drawing this on flat paper isn't too practical, however, so instead we place the first variable along the left side of the table and the other two along the top, making a horizontal chart.

This can be done in several ways. Karnaugh's clever idea was to do it so that adjacent cells represent minterms with shared factors. The key here is to let the third column represent yz (11) instead of the numerically next value $y\overline{z}$ (10). In this way, the middle two columns represent z. The last two columns represent y, the first two represent $\overline{y}$, and the last and first columns (which we will consider adjacent since we can wrap the right edge around to join the left edge) represent $\overline{z}$. The following shows how we will do this, both with letters and binary labels.**

* This association is grounded in the *Acceptance Propositions* 1 - 3 of Section 7.5.

** There seems to be no universally accepted way to set up K-maps for Boolean functions. We're following the presentation based on the Gray Code procedure for listing binary numbers.

	$\overline{y}\,\overline{z}$	$\overline{y}z$	yz	$y\overline{z}$
$\overline{x}$				
x				

y $\quad$ z

	00	01	11	10
0				
1				

Karnaugh Maps for Three-Variable Boolean Functions

✠ **EXAMPLE 7.6 - 4**

Determine the minterm Boolean expansions represented by the following K-maps and then simplify them, comparing your final result to the K-map.

a)

	$\overline{y}\,\overline{z}$	$\overline{y}z$	yz	$y\overline{z}$
$\overline{x}$				
x	1			1

b)

	$\overline{y}\,\overline{z}$	$\overline{y}z$	yz	$y\overline{z}$
$\overline{x}$		1	1	
x		1	1	

c)

	00	01	11	10
0		1	1	
1	1	1	1	1

Solution

a) The minterm expansion represented by the first K-map is $x\overline{y}\,\overline{z} + xy\overline{z}$, which simplifies, after factoring, to $x\overline{z}$.
This can be read off the K-map, for the block of 1s here is the full intersection of the x row and the $\overline{z}$ columns.

b) The minterm expansion here is $\overline{x}\,\overline{y}z + \overline{x}yz + x\overline{y}z + xyz$. This simplifies to $\overline{y}z + yz$ (corresponding to adding the second and third columns), which further simplifies to z. This fits the K-map we're given; the z columns are completely filled with 1s.

c) This K-map is the union of the last two. The minterm expansion, therefore, is given by $\overline{x}\,\overline{y}z + \overline{x}yz + x\overline{y}\,\overline{z} + x\overline{y}z + xyz + xy\overline{z}$. The sum of the simplified expressions is $x\overline{z} + z$. However, this can be further simplified to $x + z$ using a *Redundancy Law*.
Alternatively, the complement of this function is $\overline{x}\,\overline{z}$, found by inspection of the K-map. The original function is thus its complement, $\overline{\overline{x}\,\overline{z}} = x + z$.
The simplified form $x + z$ can also be read off the K-map; the given set of 1s is the union/sum of the x row and the z columns.

It should be clear that blocks of one, two, or four adjacent cells represent products of three, two, or one literals respectively (see Exercise 37). To determine a simplified expression for a given Boolean function, then, we can decompose the patterned set of 1s into a small union of large blocks, possibly overlapping. This will generate a sum of products of literals to represent the function. (Note how products correspond to intersections here and sums to unions.) To formulate this all more precisely using standard terminology, we will first define some terms.

DEFINITION* 7.6 - 1: *Adjacent Cells, Blocks, Implicants, and Coverings

a) *Two cells in a K-map are* ***adjacent*** *iff they share a common edge or they are the first and last cells in a row or column (adjacent in a wrapped sense).*

b) *A* ***block*** *for a Boolean function is a rectangular set of* 2^k *adjacent cells in the function's K-map.*

c) *An* ***implicant*** *for a Boolean function is the associated product of literals corresponding to a block for that function.*

d) *A* ***prime implicant*** *for a Boolean function is an implicant associated with a maximal block of adjacent* 1s *for that function.*

e) *An* ***essential prime implicant*** *is a prime implicant whose block contains a* 1 *not in any other prime implicant's block.*

f) *A* ***covering*** *for a Boolean function is a set of blocks whose union is the set of* 1s *for that function's K-map.*

✠ **EXAMPLE 7.6 - 5**

Find all the implicants, prime implicants, and essential prime implicants for the function of Example 4c. Then relate a covering to the simplified expression for that function.

Solution

We'll repeat the K-map for this function for the sake of convenience.

	00	01	11	10
0		1	1	
1	1	1	1	1

Each of the six minterms $\overline{x}\,\overline{y}z$, $\overline{x}yz$, $x\overline{y}\,\overline{z}$, $x\overline{y}z$, xyz, and $xy\overline{z}$ correspond to filled blocks of size 1, so they are all implicants of the function.

Similarly, the products $\overline{x}z$, $x\overline{y}$, xz, xy, $x\overline{z}$, $\overline{y}z$, and yz are implicants; they correspond to filled blocks of size 2.

Finally, the trivial "products" x and z are implicants; they correspond to blocks of size 4. These are the only prime implicants. Furthermore, both are essential: each are required to cover some 1; neither covers all the 1s in the K-map by itself.

The covering associated with these two prime implicants consists of the bottom row of four cells and the middle square block of four cells. Given this covering we can generate the simplified representation $x + z$ for this function.

✠ **EXAMPLE 7.6 - 6**

Determine the prime implicants for the K-map of the ternary majority function to find a simplified expression for the function.

Solution

The K-map for the ternary majority function is as follows (see Example 2 above).

	00	01	11	10
0			1	
1		1	1	1

There are no blocks of size four here, but there are three blocks of size two (encircled in the K-map). They correspond to the following prime implicants: xz, xy, and yz. Each of these is essential. Adding these three together gives us our simplified expression for the ternary majority function: $f(x, y, z) = xy + xz + yz$.

The idea behind using K-maps to generate simplified expressions for Boolean functions is to decompose the patterns of 1s in the K-map into a minimal union of maximal blocks, find the products associated with the blocks, and add them together. You need to be able to recognize block patterns, particularly when they stretch from one side around to the other, but otherwise the process isn't very complicated.

Karnaugh Maps for Boolean Functions of Four Variables

What if a Boolean function has more than three variables? A slight variation on the K-map for functions of three variables works for functions of four variables. Just as we used the top of a K-map to represent two dimensions, so we can use the side for two more. We will work an extended example to illustrate the process.

✠ **EXAMPLE 7.6 - 7**

For the Boolean function $f(x_1, x_2, x_3, x_4) = x_1x_2\overline{x}_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3 + x_1x_3\overline{x}_4 + \overline{x}_1x_2x_4 + \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_3x_4 + \overline{x}_1\overline{x}_3x_4 + \overline{x}_2x_3\overline{x}_4$, exhibit a K-map and then use it to determine a simplified representation for this function.

Solution

The following are two ways to label and draw the K-map for this function (check that this is so by mapping out each term). Covering boxes have been drawn in on the second one for the function's prime implicants.

	$\overline{x}_3\overline{x}_4$	$\overline{x}_3x_4$	x_3x_4	$x_3\overline{x}_4$
$\overline{x}_1\overline{x}_2$	1	1	1	1
$\overline{x}_1x_2$		1	1	
x_1x_2	1			1
$x_1\overline{x}_2$	1	1		1

x_3 x_4 x_2 x_1

	00	01	11	10
00	1	1	1	1
01		1	1	
11	1			1
10	1	1		1

The essential prime implicants for this function are $\overline{x}_1x_4$ (middle top four-square), $x_1\overline{x}_4$ (bottom left-right four-square), and $\overline{x}_2\overline{x}_3$ (left top-bottom four-square). Taking these essential prime implicants, we're still missing cell 0010 (top right corner). Other prime implicants are $\overline{x}_1\overline{x}_2$ (top row) and $\overline{x}_2\overline{x}_4$ (the "adjacent" four corners). Either of these cover the missing cell. Thus we have two possible minimal expansions for our function: $f(x_1, x_2, x_3, x_4) = \overline{x}_1x_4 + x_1\overline{x}_4 + \overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2$, and $f(x_1, x_2, x_3, x_4) = \overline{x}_1x_4 + x_1\overline{x}_4 + \overline{x}_2\overline{x}_3 + \overline{x}_2\overline{x}_4$. No further simplifications are possible.

K-maps for Boolean functions of four variables will have boxes of size 1, $2 = 1 \times 2 = 2 \times 1$, $4 = 1 \times 4 = 4 \times 1 = 2 \times 2$, $8 = 2 \times 4 = 4 \times 2$, and $16 = 4 \times 4$. Each of these (except for the last, which represents the constant expression 1) corresponds to a product of literals. You will always be able to cover a given K-map by maximal boxes of these sizes; they will be associated with the prime implicants for the function. All the essential prime implicants must be used, but as in the last example, you may need to use additional prime implicants to cover the map and finish the representation, which may not be unique.

The Quine-McCluskey Method of Simplifying Boolean Functions

In practice, K-maps are used for Boolean functions of no more than six variables. Constructing a K-map for a function of more than four variables requires us to represent three variables along the top or side of the K-map. When three variables are involved in this way, the order $000, 001, 011, 010, 110, 111, 101, 100$ is used for labeling the two sides of the K-map. Functions can be expressed by minimal sum-of-products expressions as before, using maximal

boxes of size 1, 2, 4, 8, 16, 32, and 64 to cover the 1s in the K-map. Doing this can obviously get complicated for higher order K-maps, especially since boxes can wrap around and be overlooked. It would be good, therefore, to have a method that could be programmed to find a function's prime implicants and minimal expansions.

Such a procedure is available in the Quine-McCluskey method, proposed independently in the early to mid-1950s by the philosopher Willard van Orman Quine for logic and by the electrical engineer Edward McCluskey for Boolean functions. The procedure goes as follows.

Quine-McCluskey Method

0. *Minterm Expansion*
   Begin by getting a minterm representation for the function. This isn't really part of the method, but you need it to get started, so we'll call it stage 0.
1. *Reduced Sum-of-Products Expansion*
   Rewrite the function's minterm expansion as a sum involving fewer shorter products (implicants with fewer factors):
   a. Replace *each pair* of added minterms by a single shorter product, if possible; this is accomplished by factoring and simplifying, as in $xy + x\overline{y} = x(y + \overline{y}) = x$.
   b. Apply the process of part *a* to the resulting product sums, if possible.
   c. Repeat the process on each new set of shorter products until no further combining and simplifying can occur.
2. *Miminized Sum-of-Products Expansion*
   Stage 1 generates a set of prime implicants whose sum represents the function, but not all implicants may be either essential or needed to cover the function. Conclude by further reducing the expression to a minimal sum of essential and other prime implicants.

We will illustrate this method with two simple three-variable examples and then work one for a more complex function of four variables.

✠ **EXAMPLE 7.6 - 8**

Apply the Quine-McCluskey method to the ternary majority function.

Solution

This is essentially how we worked Example 2, but we'll work through it again to illustrate the steps of the Quine-McCluskey method.

0. The minterm expansion for this function is $f(x, y, z) = xyz + xy\overline{z} + x\overline{y}z + \overline{x}yz$.
1. We can combine and replace $xyz + xy\overline{z}$ by xy, $xyz + x\overline{y}z$ by xz, and $xyz + \overline{x}yz$ by yz; no other pairs combine. This gives $f(x, y, z) = xy + xz + yz$.
   These three new terms cannot be further combined in the required way; all the prime implicants of the function have been found.
2. In this case, we can't further reduce our sum: if any term is dropped, the resulting sum will no longer generate the function: some of the function's original implicants will no longer be covered. So the minimal expansion for the ternary majority function is given by $f(x, y, z) = xy + xz + yz$.

✠ **EXAMPLE 7.6 - 9**

Apply the Quine-McCluskey method to the function $f(x, y, z) = xyz + xy\overline{z} + x\overline{y}z + x\overline{y}\,\overline{z} + \overline{x}yz + \overline{x}\,\overline{y}z$ of Examples 4c and 5.

Solution

0. The minterm expansion for this function is already given.
1. We can combine and replace $xyz + xy\overline{z}$ by xy, $xyz + x\overline{y}z$ by xz, $xyz + \overline{x}yz$ by yz; $xy\overline{z} + x\overline{y}\,\overline{z}$ by $x\overline{z}$; $x\overline{y}z + x\overline{y}\,\overline{z}$ by $x\overline{y}$, $x\overline{y}z + \overline{x}\,\overline{y}z$ by $\overline{y}z$; and $\overline{x}yz + \overline{x}\,\overline{y}z$ by $\overline{x}z$.

All possible pairs of terms have been considered in this process, so the new representation is $f(x,y,z) = xy + xz + yz + x\overline{y} + x\overline{z} + \overline{x}z + \overline{y}z$.
The terms in this expansion can be further combined. We can replace $xy + x\overline{y}$ by x; $xz + x\overline{z}$ by x, $xz + \overline{x}z$ by z; and $yz + \overline{y}z$ by z. All terms have now been incorporated into a shorter product: the new representation is $f(x,y,z) = x + z$, omitting duplicates. No further combining can be done, so we have the prime implicants of our function (see Example 4c).

2. Since both terms are needed (neither one alone suffices), our minimal expansion is as just stated: $f(x,y,z) = x + z$.

In both of the above examples, stages one and two turned out to be fairly easy. This illustrates the essentials of the procedure without getting bogged down by complications, but it is definitely not typical. To illustrate this, we will rework the more complex Example 7. This also gives us a check on our work there (in fact, working Example 10 turned up a prime implicant that was overlooked the first time Example 7 was attempted!).

✠ **EXAMPLE 7.6 - 10**

Use the Quine-McCluskey method to find the prime implicants and a minimal expansion for Example 7's function: $f(x_1,x_2,x_3,x_4) = x_1x_2\overline{x}_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3 + x_1x_3\overline{x}_4 + \overline{x}_1x_2x_4 + \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_3x_4 + \overline{x}_1\overline{x}_3x_4 + \overline{x}_2x_3\overline{x}_4$.

Solution

0. The minterm expansion for this function is $f(x_1,x_2,x_3,x_4) = \overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1\overline{x}_2x_3\overline{x}_4 + \overline{x}_1\overline{x}_2x_3x_4 + \overline{x}_1x_2\overline{x}_3x_4 + \overline{x}_1x_2x_3x_4 + x_1\overline{x}_2\overline{x}_3\overline{x}_4 + x_1\overline{x}_2\overline{x}_3x_4 + x_1\overline{x}_2x_3\overline{x}_4 + x_1x_2\overline{x}_3\overline{x}_4 + x_1x_2x_3\overline{x}_4$.
1. To facilitate our work here, we will replace minterms with their bit strings, labeled by the associated binary number they represent, and group them in increasing order according to how many 1s appear in the representation. Strings in each group may be combinable with ones from the next group, since they differ in one place where one has the variable (1) and the other has its complement (0). This corresponds to cells being adjacent in a K-map. Combining these leads to new terms having one fewer factor; these are placed in the second column. For example, 0011 + 0111 simplifies to 0–11 (dashes here indicate a missing variable), because $\overline{x}_1\overline{x}_2x_3x_4 + \overline{x}_1x_2x_3x_4 = \overline{x}_1x_3x_4$. Used minterms are marked with a $*$ and are omitted from our final sum-of-products representation; thus, both minterm 3 and minterm 7 have an asterisk after them.

Minterms			Two-Cell Combinations			Four-Cell Combinations	
0	0000	$*$	(0, 1)	000–	$*$	(0, 1; 2, 3)	00––
1	0001	$*$	(0, 2)	00–0	$*$	(0, 1; 8, 9)	–00–
2	0010	$*$	(0, 8)	–000	$*$	~~(0, 2, 1, 3)~~	~~00––~~
8	1000	$*$	(1, 3)	00–1	$*$	(0, 2; 8, 10)	–0–0
3	0011	$*$	(1, 5)	0–01	$*$	~~(0, 8; 1, 9)~~	~~–00–~~
5	0101	$*$	(1, 9)	–001	$*$	~~(0, 8; 2, 10)~~	~~–0–0~~
9	1001	$*$	(2, 3)	001–	$*$	(1, 5; 3, 7)	0––1
10	1010	$*$	(2, 10)	–010	$*$	(8, 10; 12, 14)	1––0
12	1100	$*$	(8, 9)	100–	$*$	~~(8, 12; 10, 14)~~	~~1––0~~
7	0111	$*$	(8, 10)	10–0	$*$		
14	1110	$*$	(8, 12)	1–00	$*$		
			(3, 7)	0–11	$*$		
			(10, 14)	1–10	$*$		
			(12, 14)	11–0	$*$		

- The new combined product terms are labeled by which minterms gave rise to them; the result $0-11$ is labeled $(3,7)$ because it resulted from combining minterm 3 with minterm 7. It corresponds to covering the 3-cell and the 7-cell in a K-map.
- Once the entire second column of two-cell combined terms is generated, the same process is repeated with these strings. Two-cells that are adjacent are combined into a four-cell block and are marked. Additional columns are added as necessary; in this case, no new combinations are possible among the four-cell blocks.
- In the end, all those strings *not* marked will correspond to maximal blocks and so will yield prime implicants for the function. Here these all appear in the last column, but this will not always happen. The associated products are then used to form a sum-of-products representation for the function, duplicates being crossed out and omitted.
- The result of all this is: $f(x_1, x_2, x_3, x_4) = \overline{x}_1\overline{x}_2 + \overline{x}_2\overline{x}_3 + \overline{x}_2\overline{x}_4 + \overline{x}_1x_4 + x_1\overline{x}_4$.

2. Comparing the result of stage one with what we obtained in Example 7 using the function's K-map, we see that we have not yet achieved a minimal expansion. To determine which of the prime implicants are essential and then reduce this sum, if possible, to a minimal expansion, we will determine which of the original minterms are covered by which of these implicants. We will do this using a table for minimizing the list of prime implicants.

Prime Implicants	Minterms (Binary Cell Label)										
	0	1	2	3	5	7	8	9	10	12	14
$(0,1,2,3)$	X	X	X	X							
$(0,1,8,9)$	X	X					X	X			
$(0,2,8,10)$	X		X				X		X		
$(1,5,3,7)$		X		X	X	X					
$(8,10,12,14)$							X		X	X	X

We must cover all the minterms in the function's minterm expansion. From the chart we can see that 5 and 7 are only covered by the prime implicant $(1,5,3,7)$; 9 is only covered by the prime implicant $(0,1,8,9)$; and 12 and 14 are only covered by the prime implicant $(8,10,12,14)$. We've drawn horizontal lines through the Xs covered by these essential prime implicants. Together these three essential prime implicants cover 0, 1, 3, 5, 7, 8, 9, 10, 12, and 14: all but 2. We've drawn vertical lines to indicate this. We can cover the remaining minterm either by implicant $(0,1,2,3)$ or implicant $(0,2,8,10)$. Translating this into a literal expression, we get the following two sum-of-products representations (see the last table for the products associated with the four-cell combinations): $f(x_1, x_2, x_3, x_4) = \overline{x}_1\overline{x}_2 + \overline{x}_2\overline{x}_3 + \overline{x}_1x_4 + x_1\overline{x}_4$ and $f(x_1, x_2, x_3, x_4) = \overline{x}_2\overline{x}_3 + \overline{x}_2\overline{x}_4 + \overline{x}_1x_4 + x_1\overline{x}_4$. These agree with the expansions we obtained in Example 7.

Boolean Algebra: Theory and Practice

We've now had an opportunity to explore *Boolean Algebra* both theoretically and practically, both abstractly and concretely. We began this chapter by looking at the theory of posets and lattices. There we noted the surprising result that all finite Boolean lattices are essentially power-set lattices ordered by the subset relation. This gave us a concrete representation for an abstract notion. Since all Boolean lattices can be converted into Boolean algebras and conversely, this also gave us a strong connection between *Boolean Algebra* and *Set Theory*.

Boolean Algebra can be considered an abstract generalization of *Sentential Logic* as well as *Set Theory*. Thus, our work with Boolean functions focused on their connections to logic circuits. Here we noted that all Boolean functions/expressions were equivalent, even identical,

with certain normal representations, such as minterm expansions. This means practically that all Boolean logic circuits can be built using only the most basic logic gates (AND, OR, NOT). Theoretically, it means that the equational part of *Boolean Algebra* can be decided by truth tables, showing a strong connection between *Boolean Algebra* and *Sentential Logic*.

Besides using *Boolean Algebra* to work with logic circuits, we saw that we can use logic circuits to do ordinary computation. This forms the basis of twentieth-century computer science applications to mathematics.

Finally, in this last section we looked at ways to simplify Boolean expressions in sums-of-products form. Karnaugh maps provide a nice visual method for handling simple functions, while the Quine-McCluskey method gives a tabular technique that can be developed into an algorithm implemented on a computer. Simplification of Boolean expressions is obviously important to anyone who is designing complex logic circuits, both in engineering generally and computer science in particular.

Thus, while *Boolean Algebra* is quite abstract and theoretical, it is at the same time very concrete and practical, forming the theoretical basis of some of the most important technological developments of the last century. It thus provides a very nice example of the interplay that can occur between these varied aspects of mathematics, attracting people with many different interests and attitudes towards theoretical and practical mathematics. Given the very concrete applications it has, *Boolean Algebra* makes an excellent first introduction to advanced abstract mathematics. And, given its links to the areas of *Sentential Logic* and *Set Theory* studied earlier in the text, it is also satisfying for its ability to unify different aspects of foundational discrete mathematics, providing reinforcement and a theoretical context for those fields.

EXERCISE SET 7.6

Problems 1 - 6: Simplifying Boolean Functions Using Boolean Algebra

Simplify the following functions using Boolean Algebra. Explain, by comparing the number of literals and number of operations involved, why your answer is simpler than the given function.

1. $f(x,y) = x\overline{y} + \overline{x}\,\overline{y}$

*2. $f(x,y) = xy + x\overline{y} + \overline{x}y$

3. $f(x,y,z) = xy + x\overline{yz} + \overline{x}yz$

4. $f(x,y,z) = xy + x\overline{y}z + x\overline{z} + \overline{x}\,\overline{y}z + yz$

*5. $f(x,y,z) = xy + xyz + xy\overline{z} + \overline{x}yz + \overline{x}y\overline{z}$

6. $f(w,x,y,z) = wz + \overline{w}y\overline{z} + x\overline{y}\,\overline{z} + \overline{x}\,\overline{y}\,\overline{z}$

Problems 7 - 10: True or False

Are the following statements true or false? Explain your answer.

*7. The minterm expansion for a Boolean function yields a simplified expression for the function.

*8. An implicant for a Boolean function expressed as a minterm expansion is one of the minterms.

9. An essential prime implicant for a Boolean function is a term that must appear in a simplified sum-of-products representation of the function.

10. The only terms that appear in a simplified sum-of-products representation of a Boolean function are essential prime implicants

Problems 11 - 14: Determining Implicants for K-Maps

For the following K-maps, identify the following. Use alphabetic order for your variables.

a) *all the implicants of the associated Boolean function*;
b) *all the function's prime implicants*; *and*
c) *all the function's essential prime implicants.*

11.

	0	1
0	1	1
1	1	

*12.

	00	01	11	10
0	1		1	1
1	1			1

13.

	00	01	11	10
0	1			1
1	1	1	1	1

*14.

	00	01	11	10
00	1	1	1	1
01		1		
11		1	1	
10		1	1	1

Problems 15 - 18: Simplifying Boolean Functions Represented by K-maps

For each of the following K-maps, determine:

a) *the minterm expansion for the associated Boolean function, and*
b) *a simplified sum of products of literals representation for your minterm expansion.*

15. The K-map given in Exercise 11.

*16. The K-map given in Exercise 12.

17. The K-map given in Exercise 13.

*18. The K-map given in Exercise 14.

Problems 19 - 28: Simplifying Boolean Functions Using K-maps

Draw K-maps for the following Boolean functions. Then use these K-maps to determine simplified sums of products for the given functions.

*19. $f(x, y) = xy + \overline{x}$

20. $f(x, y) = xy + x\overline{y} + \overline{x}\,\overline{y}$

*21. $f(x, y, z) = xyz + xy\overline{z} + \overline{x}\,\overline{y}z + \overline{xy}z$

22. $f(x, y, z) = xy + x(\overline{y + z}) + \overline{xy}z + \overline{y}\,\overline{z}$

23. $f(x, y, z) = xyz + \overline{x}\,\overline{y}z + \overline{x}\,\overline{z} + yz + \overline{y}\,\overline{z}$

24. $f(x, y, z) = x\overline{y}\,\overline{z} + \overline{x}y\overline{z} + \overline{x}\,\overline{y}z + \overline{x}\,\overline{yz}$

25. $f(x,y,z) = xyz + xy\overline{z} + x\overline{y}z + x\overline{y}z + \overline{x}\,\overline{y}z + \overline{x}\,\overline{y}\,\overline{z}$

*26. $f(w,x,y,z) = wxz + w\overline{xyz} + xz + y\overline{z}$

27. $f(w,x,y,z) = w\overline{x}y\overline{z} + w\overline{x}\,\overline{y}\,\overline{z} + \overline{w}xyz + \overline{w}\,\overline{x}y\overline{z} + \overline{w}\,\overline{x}\,\overline{y}\,\overline{z}$

28. $f(w,x,y,z) = wxy + x\overline{y}z + wy\overline{z} + wxyz + \overline{w}x\overline{y}\,\overline{z}$

Problems 29 - 36: Quine-McCluskey Method

Use the Quine-McCluskey method to determine simplified representations for the following Boolean functions. For three-variable functions, follow the procedure of Examples 8 *and* 9; *for four-variable functions follow Example* 10*'s tabular method.*

*29. The Boolean function of Exercise 21.

30. The Boolean function of Exercise 22.

31. The Boolean function of Exercise 23.

32. The Boolean function of Exercise 24.

33. The Boolean function of Exercise 25.

EC 34. The Boolean function of Exercise 26.

35. The Boolean function of Exercise 27.

36. The Boolean function of Exercise 28.

Problems 37 - 39: Blocks in K-Maps

The following problems explore blocks of cells in K-maps.

*37. *Boolean Expressions for Blocks in a Three-Variable K-map.*

a. Explain, by considering all possibilities, why in a K-map for three-variable functions a rectangular block of two adjacent cells represents a product of two literals.

b. Explain, by considering all possibilities, why in a K-map for three-variable functions a rectangular block of four adjacent cells represents a single literal.

38. *Boolean Expressions for Blocks in a Four-Variable K-map.*

a. Explain, by considering all possibilities, why in a K-map for four-variable functions a rectangular block of two adjacent cells represents a product of three literals.

b. Explain, by considering all possibilities, why in a K-map for four-variable functions a rectangular block of four adjacent cells represents a product of two literals.

c. Explain, by considering all possibilities, why in a K-map for four-variable functions a rectangular block of eight adjacent cells represents a single literal.

39. *Boolean Expressions for Blocks in an N-Variable K-map.*

a. Generalize the results of Problems X and Y: in a K-Map for an n-variable function, a rectangular block of 2^k adjacent cells represents a product of _____ literals, for $k = 0, 1, \ldots, n-1$. What function does the full block of 2^n cells represent?

b. Explain why your formula in part *a* is correct.

Problems 40 - 42: Implicants for Boolean Functions

The following problems explore implicants for Boolean Functions.

*40. *Implicants*

Claim: *if an implicant for a function equals* 1*, then the function as a whole also equals* 1.

a. Show from the definition that xy is an implicant for $f(x,y,z) = x + z$; then use this to illustrate the given claim.

b. Argue the given claim in general.

c. Explain why the term *implicant* is an appropriate one.

d. Is the converse of the above claim also true? What if the implicant is a prime implicant? an essential prime implicant?

41. *Maximal Blocks and Implicants*
    a. Explain why maximal blocks in a K-map must correspond to prime implicants. Must maximal blocks also be essential prime implicants?
    b. Prove or give a counterexample: the sum a Boolean function's essential prime implicants provides a Boolean expression representation for the function.

42. *The Quine-McCluskey Method and Prime Implicants*
    Explain why all products that haven't been checked off in the tabular approach to the Quine-McCluskey method are prime implicants of the function.

43. *Exploration*: *Minimization Using Other Logic Gates*
    Show that while the processes studied in this section minimize the number of multiplications and additions of literals, they can be further minimized if other logic gates are allowed.

HINTS TO STARRED EXERCISES 7.6

2. Use the Boolean *Distributive Laws* to factor and expand.

5. Use the Boolean *Distributive Laws* and *Idempotence Laws* to simplify.

7. [No hint.]

8. [No hint.]

12. a. Determine expressions for all blocks in the K-map.
    b. Determine expressions for all maximal blocks in the K-map.
    c. Only include expressions for maximal blocks that cover a 1 not covered by any other maximal block.

14. a. Determine expressions for all blocks in the K-map.
    b. Determine expressions for all maximal blocks in the K-map.
    c. Only include expressions for maximal blocks that cover a 1 not covered by any other maximal block.

16. Use your work from #12 here.

18. Use your work from #14 here.

19. Follow the same procedure as in Example 3.

21. Follow the same procedure as in Example 6.

26. Follow the same procedure as in Example 7.

29. Follow the same procedure as in Example 9.

37. Use an exhaustive case analysis.

40. a. Review the meaning of implicant.
    b. Generalize your argument in part *a*.
    c. Think in terms of SL being a Boolean algebra.
    d. Consider the example of part *a*.