

PART IIc: PROPOSITIONAL LOGIC

Version 2.30: 2003-02-09

John Greiner
Ian Barland

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License *

Abstract

(Blank Abstract)

Part IIc: Propositional Logic

1 Propositional Equivalences

What are the roots of $x^3 - 4x$? Well, in high-school algebra you learned how to deal with such numeric formulas:

$$x^3 - 4x \tag{1}$$

$$= x(x^2 - 4) \text{ [factor out x]} \tag{2}$$

$$= x(x - 2)(x + 2) \text{ [identity } (a^2 - b^2 = (a + b)(a - b)) \text{, with } a \text{ being } x \text{, and } b \text{ being } 2.] \tag{3}$$

(This last expression happens to be useful since it is in a form which lets us read off the roots 0, +2, -2.) The rules of algebra tell us that these three *different* formulas are all equivalent. We are distinguishing between **syntax** (the expression itself, as data), and **semantics** (what the expression means). Usually, when presented with syntax, one is supposed to bypass that and focus on its meaning (e.g., reading a textbook). However, in logic and postmodern literature alike, we are actually studying the interplay between syntax and semantics. The general gist is that in each step, you rewrite subparts of your formula according to certain rules ("replacing equals with equals").

Well, we can use a similar set of rules about rewriting formulas with equivalent ones, to answer the questions of whether two formulas are equal, or whether a formula is a tautology. George Boole¹ was the first to realize that true and false are just values in the way that numbers are, and he first codified the rules for manipulating them; thus **Boolean algebra** is named in his honor.

ASIDE: The "algebra²" comes from true, false, \wedge , \vee having some very specific properties similar to those of numbers, \times , $+$.

Again, each individual step consists of rewriting a formula according to certain rules. So, just what are the rules for manipulating Boolean values? We'll start with an example.

* <http://creativecommons.org/licenses/by/1.0>

¹ <http://kerryr.net/pioneers/boole.htm>

² <http://planetmath.org/encyclopedia/Algebra.html>



Figure 1: George Boole (1815-1864)

Example 1:

1. $((a \wedge \text{false}) \vee (b \wedge \text{true}))$
2. $\equiv (\text{false} \vee (b \wedge \text{true}))$ [Dominance of false over \wedge]
3. $\equiv ((b \wedge \text{true}) \vee \text{false})$ [Commutativity of \vee]
4. $\equiv (b \wedge \text{true})$ [Identity element for \vee is false]
5. $\equiv b$ [Identity element for \wedge is true]

Thus we have a series of equivalent formulas, with each step justified by citing a propositional equivalence³ (.ps)⁴. By and large, the equivalences are rather mundane. A couple are surprisingly handy; take a moment to consider DeMorgan's laws.

$$\boxed{\neg(\phi \wedge \psi) \equiv (\neg\phi \vee \neg\psi) \quad \neg(\phi \vee \psi) \equiv (\neg\phi \wedge \neg\psi)}$$

(Try ϕ being "leprechauns are green", and ψ being "Morgana Le Fay likes gold". Does these laws make sense?) Augustus DeMorgan⁵ was also an important figure in the formalization of logic.

Here are two more examples. The first is a proof of one of the laws from the given list, using others from the list.

Example 2:

Absorption of \vee

1. $((a \wedge b) \vee b)$
2. $\equiv ((a \wedge b) \vee (b \wedge \text{true}))$ [Identity of \wedge]
3. $\equiv ((b \wedge a) \vee (b \wedge \text{true}))$ [Commutativity of \wedge]

³<http://cnx.rice.edu/content/m10540/latest/>

⁴<http://www.teachLogic.org/Base/Printables/algebra-laws.ps>

⁵http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/De_Morgan.html



Figure 2: Augustus DeMorgan (1806-1871)

4. $\equiv (b \wedge (a \vee \text{true}))$ [Distributivity of \wedge over \vee]

5. $\equiv (b \wedge \text{true})$ [Dominance of \vee]

6. $\equiv b$ [Identity of \wedge]

Example 3:

Contrapositive

1. $(a \rightarrow b)$

2. $\equiv (\neg a \vee b)$ [Definition of \rightarrow]

3. $\equiv (b \vee \neg a)$ [Commutativity of \vee]

4. $\equiv (\neg \neg b \vee \neg a)$ [Double Complementation]

5. $\equiv (\neg b \rightarrow \neg a)$ [Definition of \rightarrow]

Exercise 1:

Show that the "Absorption of \wedge " equivalence holds, given the other equivalences. I.e., show $((a \vee b) \wedge b) \equiv b$.

Solution:

1. $((a \vee b) \wedge b)$

2. $\equiv ((a \vee b) \wedge (b \vee \text{false}))$ [Identity of \vee]

3. $\equiv ((b \vee a) \wedge (b \vee \text{false}))$ [Commutativity of \vee]

4. $\equiv (b \vee (a \wedge \text{false}))$ [Distributivity of \vee over \wedge]

5. $\equiv (b \vee \text{false})$ [Dominance of \wedge]

6. $\equiv b$ [Identity of \vee]

Compared to proofs using truth tables, Boolean algebra gives us much shorter proofs. But, determining *which* equivalence to use in the next step of a proof can be difficult. In this case, compare the solution for this exercise to the previous absorption proof. These two proofs have a special *dual* relationship described in the next section.

Exercise 2:

Show that the **modus ponens** rule, $((a \wedge (a \rightarrow b)) \rightarrow b)$ always holds. I.e., show that it is a tautology, and thus equivalent to true.

Solution:

1. $((a \wedge (a \rightarrow b)) \rightarrow b)$

2. $\equiv ((a \wedge (\neg a \vee b)) \rightarrow b)$ [Definition of \rightarrow]

3. $\equiv (((a \wedge \neg a) \vee (a \wedge b)) \rightarrow b)$ [Distributivity of \vee over \wedge]

4. $\equiv ((\text{false} \vee (a \wedge b)) \rightarrow b)$ [Complement]

5. $\equiv (((a \wedge b) \vee \text{false}) \rightarrow b)$ [Commutativity of \vee]

6. $\equiv ((a \wedge b) \rightarrow b)$ [Identity of \vee]

7. $\equiv (\neg (a \wedge b) \vee b)$ [Definition of \rightarrow]

8. $\equiv ((\neg a \vee \neg b) \vee b)$ [DeMorgan's law]

9. $\equiv (\neg a \vee (\neg b \vee b))$ [Associativity of \vee]

10. $\equiv (\neg a \vee (b \vee \neg b))$ [Commutativity of \vee]

11. $\equiv (\neg a \vee \text{true})$ [Complement]

12. \equiv true [Dominance of \vee]

So, what would it mean to use Boolean algebra as reasoning for WaterWorld? That is, if you wanted to show that $G - \text{safe}$ was true, how would you do that using Boolean algebra? It would mean starting with a big rule involving the conjunction of all the WaterWorld domain axioms (ρ), and the board's observed state (ψ), and showing that these were equivalent to $G - \text{safe}$ (\dots and the domain axioms and state). That is, $(\rho \wedge \psi) \equiv (G - \text{safe} \wedge \rho \wedge \psi)$.

1.1 Duals (optional)

Duals: a symmetry between \wedge , \vee mediated by \neg .

Looking at the provided propositional equivalences⁶ (.ps)⁷, you should notice a strong similarity between those for \vee and those for \wedge . Take any equivalence, swap \vee s and \wedge s, swap true and false, and you'll have another equivalence! For instance, there are two flavors of DeMorgan's law, which are just duals of each other:

$$\boxed{\neg(\phi \wedge \psi) \equiv (\neg\phi \vee \neg\psi) \quad \neg(\phi \vee \psi) \equiv (\neg\phi \wedge \neg\psi)}$$

ASIDE: In terms of circuit diagrams, we can change each AND gate to an OR gate and add negation-bubbles to each gate's inputs and outputs. The principle of duality asserts that this operation yields an equivalent circuit.

The idea of duality is more general⁸ than this. For example, polyhedra have a natural dual⁹ of interchanging the role of vertices and faces.

1.2 CNF, DNF, ... (ENuff already!)

In high school algebra, you saw that while $x^3 - 4x$ and $x(x - 2)(x + 2)$ are equivalent, the second form is particularly useful in letting you quickly know the roots of the equation. Similarly, in Boolean algebra there are certain canonical, or "normal", forms which have nice properties.

A formula in **Conjunctive Normal Form**, or **CNF**, is the conjunction of formulas ("clauses"). Each clause is in turn of a simple form: it must be a disjunction of possibly-negated propositions, and nothing more.

Example 4:

$((a \wedge b) \rightarrow c)$ is equivalent to $((a \vee \neg c) \wedge (b \vee \neg c))$; this latter formula is in CNF (it is the conjunction of disjunctions).

Another format, **Disjunctive Normal Form**, or **DNF** is the dual of conjunctive normal form. A DNF formula is the disjunction of clauses, where each clause is a conjunction of possibly-negated propositions.

Example 5:

$((a \wedge b) \rightarrow c)$ is equivalent to $(\neg a \vee \neg b \vee c)$ which is in DNF: three disjunctions, each being a clause with only one term. It is also equivalent to the more fleshed out DNF formula where we insist that each clause include all three variables. We

⁶<http://cnx.rice.edu/content/m10540/latest/>

⁷<http://www.teachLogic.org/Base/Printables/algebra-laws.ps>

⁸<http://carbon.cudenver.edu/~hgreenbe/glossary/duals.html>

⁹<http://www.georgehart.com/virtual-polyhedra/duality.html>

end up with a formula that includes each possible clause *except* $(a \wedge b \wedge \neg c)$: That is, the formula $((a \wedge b \wedge c) \vee (a \wedge \neg b \wedge c) \vee (a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge b \wedge c) \vee (\neg a \wedge b \wedge \neg c) \vee (\neg a \wedge \neg b \wedge c) \vee (\neg a \wedge \neg b \wedge \neg c))$

Any Boolean function can be represented in CNF and in DNF. One way to obtain CNF and DNF formulas is based upon the truth table for the function.

- A DNF formula results from looking at a truth table, and focusing on the rows where the function is true: As if saying "I'm in this row, or in this row, or ...": For each row where the function is true, form a conjunction of the propositions. (E.g., for the row where a is false and b is true, form $(\neg a \wedge b)$.) Now, form the disjunction of all those conjunctions.
- A CNF formula is the pessimistic approach, focusing on the rows where the function is false: "I'm not in this row, and not in this row, and ...". For each row where the function is false, create a formula for "not in this row": (E.g., if in this row a is false and b is true; form $\neg(\neg a \wedge b)$; then notice that by DeMorgan's law, this is $(a \vee \neg b)$ – a disjunct. Now, form the conjunction of all those disjunctions.

Example 6:

Truth table example

a	b	c	Unknown function
false	false	false	false
false	false	true	false
false	true	false	true
false	true	true	true
true	false	false	false
true	false	true	true
true	true	false	false
true	true	true	false

For CNF, the false rows give us the following five clauses:

- $(a \vee b \vee c)$
- $(a \vee b \vee \neg c)$
- $(\neg a \vee b \vee c)$
- $(\neg a \vee \neg b \vee c)$
- $(\neg a \vee \neg b \vee \neg c)$

and the full formula is the conjunction of these. Essentially, each clause rules out one row as being true.

For DNF, the true rows give us the following three clauses:

- $(\neg a \wedge b \wedge \neg c)$
- $(\neg a \wedge b \wedge c)$
- $(a \wedge \neg b \wedge c)$

and the full formula is the disjunction of these. Essentially, each clause allows one row to be true.

This shows that, for any arbitrarily complicated WFF, we can find an equivalent WFF in CNF or DNF. These provide us with two very regular and relatively uncomplicated forms to use.

Exercise 3:

The above example (Example 6) (.pdf)¹⁰ (.ps)¹¹ produced CNF and DNF formulas for a Boolean function, but they are *not* the simplest such formulas. For fun, can you find simpler ones?

Solution:

- CNF: $((a \vee b) \wedge (\neg a \vee b \vee c) \wedge (\neg a \vee \neg b))$
- DNF: $((\neg a \wedge b) \vee (a \wedge \neg b \wedge c))$

ASIDE: There is a general technique, Karnaugh maps¹², for finding minimal CNF and DNF formulas, which works when only a small number of variables are involved. We won't worry about minimizing formulas ourselves, though.

1.2.1 Notation for DNF, CNF

Sometimes you'll see the form of CNF and DNF expressed in a notation with subscripts.

- DNF is $i\varphi_i$, where each clause φ_i is $j\lambda_i$, where each λ is a propositional variable (Prop), or a negation of one (\neg Prop).
- CNF is $i\psi_i$, where each clause ψ_i is $j\lambda_i$, where each λ is again a propositional variable (Prop), or a negation of one (\neg Prop).

For example, in the CNF formula $((a \vee b) \wedge (\neg a \vee b \vee c) \wedge (\neg a \vee \neg b))$ we have $\text{phiv}_2 = (\neg a \vee b \vee c)$; within that clause we have $\lambda_2 = \neg a$.

One question this notation brings up:

- What is the disjunction of a single clause? Well, it's reasonable to say that $\psi \equiv \psi$. Note that this is also equivalent to $(\psi \vee \text{false})$.
- What is the disjunction of zero clauses? Well, if we start with $\psi \equiv (\psi \vee \text{false})$ and remove the ψ , that leaves us with false! Alternately, imagine writing a function which takes a list of booleans, and returns the \vee of all of them – the natural base case for this recursive list-processing program turns out to be false. Indeed, this is the accepted definition of the empty disjunction; It follows from false being the identity element for \vee . (Correspondingly, a conjunction of no clauses is true.)

Actually, that subscript notation above isn't *quite* correct: it forces each clause to be the same length, which isn't actually required for CNF or DNF. For fun, you can think about how to patch it up. (Hint: double-subscripting.)

Note that often one of these forms might be more concise than the other. Here are two equivalently verbose ways of encoding true, in CNF and DNF respectively: $((a \vee \neg a) \wedge (b \vee \neg b))$

¹⁰<http://www.teachLogic.org/Base/Printables/partIIc.pdf>

¹¹<http://www.teachLogic.org/Base/Printables/partIIc.ps>

¹²<http://www.ee.surrey.ac.uk/Projects/Labview/minimisation/karnaugh.html>

$\neg b) \wedge \dots \wedge (z \vee \neg z)$) is equivalent to $((a \wedge b \wedge c \wedge \dots \wedge y \wedge z) \vee (a \wedge b \wedge c \wedge \dots \wedge y \wedge \neg z) \vee (a \wedge b \wedge c \wedge \dots \wedge \neg y \wedge z) \vee \dots \vee (\neg a \wedge \neg b \wedge \dots \wedge \neg y \wedge \neg z))$. The first version corresponds to enumerating the choices for each location of a WaterWorld board; it has 26 2-variable clauses. The second version corresponds to enumerating all possible WaterWorld boards explicitly: there are 64 billion 26-variable clauses!

2 Are we done yet?

We have shown (for both truth-tables and for Boolean algebra), how to solve two problems:

- Equivalence: Show whether or not two WFFs ϕ and ψ are equivalent;
- Truth: Show whether or not a given WFF θ is true.

Exercise 4:

Which of these two problems seems harder than the other? That is, suppose if you have a friend who can solve *any* (say) Equivalence problem efficiently. But you want to open a business which will solve *any* Truth problem efficiently. Can you open your business and, by subcontracting out specific Equivalence problems to your friend, really solve any Truth problem brought to you? (This question is sometimes phrased as "Does Truth **reduce** to Equivalence?") Or, does it work the other way – does Equivalence reduce to Truth?

Solution:

We can indeed reduce the question of Truth to the question of Equivalence: if somebody asks you whether θ is true, you can just turn around and ask your friend whether the following two formulas are equivalent: θ , and true. Your friend's answer for this variant question will be your answer to your customer's question about θ . Thus, the Truth problem isn't particularly harder than the Equivalence problem.

But also, Truth can be reduced to Equivalence: if somebody asks you whether ϕ is equivalent to ψ , you can construct a new formula $((\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi))$. This formula is true exactly when ϕ and ψ are equivalent. So, you ask your friend about its truth, and you then have your answer. Thus, the Equivalence problem isn't particularly harder than the Truth problem!

Given these two facts (that the problems each reduce to each other), we realize that really they are essentially the same problem, in disguise.

But we have a more fundamental question to ask, about the method of using propositional equivalences to prove something: Where does the initial list of allowable rules come from, and how do we know they're valid? They can all be checked by truth tables!

Exercise 5:

Using a truth table, show the validity of the Redundancy of \wedge : $(\phi \wedge (\neg\phi \vee \psi)) \equiv (\phi \wedge \psi)$

Solution:

Compare the last two columns in the following:

Truth table to prove Redundancy valid

a	b	$(\neg a \vee b)$	$(a \wedge (\neg a \vee b))$	$(a \wedge b)$
false	false	true	false	false
false	true	true	false	false
true	false	false	false	false
true	true	true	true	true

This is called **soundness**: If, using Boolean algebra, we derive that ϕ and ψ are equivalent, then truly they are equivalent. (Whew!) By the way, there is one subtle point: our truth table tells us that $(a \wedge b)$ and $(b \wedge a)$ are equivalent. But then suddenly we generalize this to saying that for *any* formulas ϕ and ψ , $(\phi \wedge \psi)$ and $(\psi \wedge \phi)$ are also equivalent. What lets us justify that step? It's because any given formula will be either true or false, so we can reduce the entire formula to a single true/false proposition.

Is Boolean algebra enough? Could I have given you a homework problem you couldn't manipulate? Hmm, good question! The property we desire here is called the **completeness** of Boolean algebra: any equivalence which is true can be proved.

It turns out that, given any two formulas which really are equivalent, Boolean algebra *is* indeed sufficiently powerful to show that. Put both formulas into CNF (or, DNF); if the truth tables are equal then the CNF formulas will be equal. (Well, there are a few details to take care of: you have to order the clauses alphabetically, eliminate any duplicate clauses, and include all variables in each clause. This might be tedious, but not difficult.) Thus, Boolean algebra is complete, since (we state without proof) this procedure can always be carried out.

The concepts of soundness and completeness can be generalized to any system.

Definition 1: soundness

If the system (claims to) prove something is true, it really is true.

Definition 2: completeness

If something really is true, the system is capable of proving it.