

PART I B

Version 2.25: 2004/05/14 13:59:01.149 GMT-5

John Greiner
Ian Barland

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

Abstract

(Blank Abstract)

part I B

We've seen how to express concepts as precise formulas, and how some formulas ("tautologies") were true for all truth assignments, while others might be true or false, depending on the truth assignment. This leads to a question of *how* can we determine when a formula is a tautology? How can we tell if two different formulas are equivalent for all truth assignments? We'll look at three different methods of answering these questions:

- reasoning with truth tables (this section),
- reasoning with equivalences¹, and
- reasoning with inference rules².

1 Using Truth Tables

Is \rightarrow associative? In other words, is $(a \rightarrow (b \rightarrow c))$ always equivalent to $((a \rightarrow b) \rightarrow c)$? What is a *methodical* way of answering questions of this type? We can make a truth table with two output columns, one for each formula in question, and then just check whether those two columns are the same.

Exercise 1:

Use truth-tables to show that $(a \rightarrow (b \rightarrow c))$ and $((a \rightarrow b) \rightarrow c)$ aren't equivalent.

Solution:

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

¹<http://cnx.rice.edu/content/m10717/latest/>

²<http://cnx.rice.edu/content/m10718/latest/>

Truth table to check associativity of implication

a	b	c	$(a \rightarrow (b \rightarrow c))$	$((a \rightarrow b) \rightarrow c)$
false	false	false	true	false
false	false	true	true	true
false	true	false	true	false
false	true	true	true	true
true	false	false	true	true
true	false	true	true	true
true	true	false	false	false
true	true	true	true	true

By inspecting the two right-most columns, we see that the formulas are indeed not equivalent. They have different values for two truth-settings, those with a =false and c =false.

Thus we see that truth tables are a method for answering questions of the form "Is formula ϕ equivalent to formula ψ ?" We make a truth table, with a column for each of ϕ and ψ , and just inspect whether the two columns always agree. A bit of a brute-force solution, but certainly correct.

What about the related question, "Is formula θ a tautology?". Well, obviously truth tables can handle this as well: make a truth table for the formula, and inspect whether all entries are true. For example, in the above problem (Exercise 1), we could have made a truth table for the single formula $((a \rightarrow (b \rightarrow c)) \leftrightarrow ((a \rightarrow b) \rightarrow c))$. The original question of equivalence becomes, is this new formula a tautology?

The first approach is probably a tad easier to do by hand, though clearly the two approaches are equivalent. Another handy trick is to have *three* output columns you're computing: one for $\phi = (a \rightarrow (b \rightarrow c))$, one for $\psi = ((a \rightarrow b) \rightarrow c)$, and one for $(\phi \leftrightarrow \psi)$; filling out the first two helper columns makes it easier to fill out the last column.

TIP: When making a truth table for a large complicated WFF by hand, it's helpful to make columns for sub-WFFs; as you fill in a row, you can use the results of one column to help you calculate the entry for a later column.

Exercise 2:

Is it valid to replace the conditional

```
int do_something(int value1, int value2)
{
    bool a = ...;
    bool b = ...;

    if (a && b)
        return value1;
    else if (a || b)
        return value2;
    else
        return value1;
}
```

with this conditional?

```
int do_something(int value1, int value2)
{
    bool a = ...;
    bool b = ...;

    if ((a && !b) || (!a && b))
        return value2;
    else
        return value1;
}
```

After all, the later seems easier to understand, since it has only two cases, instead of three.

Solution:

In the original code, we return `value2` when the first case is false, but the second case is true. Using a WFF, when $(\neg(a \wedge b) \wedge (a \vee b))$. Is this equivalent to the WFF $((a \wedge \neg b) \vee (\neg a \wedge b))$? Here is a truth table:

Truth table for comparing conditionals' equivalence

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

Yes, looking at the appropriate two columns we see they are equivalent.

2 Are we done yet?

Are we done with propositional logic, now that we can test for equivalence and tautologies, using truth tables? Possibly. Truth tables can answer any question about propositional logic, but not always conveniently.

Consider the following code:

```
bool do_something(int value)
{
    bool a = ...;
    bool b = ...;

    if (a && !b)
        return true;
    else if (!a && !b)
        return false;
```

```

else if (a)
    return a;
else if (b)
    return false;
else
    return true;
}

```

Clearly, this is very ugly and should be simplified. But to what? We could build a truth table for the corresponding WFF, but so far we don't have any better way of finding a simpler equivalent formula than testing equivalence with whatever comes to mind. We need some way to generate formulas, given either an equivalent formula or a truth table.

There is another practical difficulty with truth table: they can get unwieldy.

Exercise 3:

How many rows are there in a truth table with 2 input variables? 3 inputs? 5? 10? n ?

Solution:

- 2 variables: As we're seen, 4 rows.
- 3 variables: 8 rows.
- 5 variables: 32 rows.
- 10 variables: 1024 rows.
- n variables: 2^n rows.

Exercise 4:

(*Optional*) Now, how many such boolean *functions* are possible, with 2 inputs? With 3? For fun, sit down and name all the possible two-input functions. You'll find that some of them are rather boring, such as the constant function "true", and many are just permutations on \rightarrow .

Solution:

- With 2 variables, we have 4 rows. How many different ways can we assign "true" and "false" to those 4 positions? If you write them all out, you should get 16 combinations.
- With 3 variables, we have 8 rows and a total of 256 different functions.
- With n variables, we have 2^n rows and a total of 2^{2^n} different functions. That's a lot!

When discussing a circuit with 100 wires (each corresponding to a single proposition), truth tables are clearly infeasible³. Modern processors have *millions* of wires and transistors. It is still an area of active research to cope with such a huge number of possibilities. (The key idea is to break things down into small sections, prove things about the small sections, and hopefully have a small set of sentences formally capturing the interface between sections.)

So truth tables are intractable for analyzing circuits of more than a few wires. But will they suffice for answering WaterWorld questions? Image a (large) table with all the neighbor propositions: A – has – 0, B – has – 0, ..., A – has – 1, B – has – 1, ... Now, determine which rows which entail B – safe. To answer this, we end up looking at rows involving many clearly-irrelevant propositions such as Z – has – 2.

³<http://www.cs.rice.edu/~ian/Misc/exponential-is-big.shtml>

ASIDE: Hmm, considering every possible board and then counting what proportion of boards entail B – safe – hmm, this is the brute-force definition of probability! Since such truth tables enumerates all possible boards, it's like looking for probability 1 the brute-force way.

Also, this method of playing WaterWorld via huge truth tables would be unsatisfying for another reason: it doesn't actually reflect our own reasoning. As a general principle of programming, your program should always reflect how *you* conceive of the problem. The same applies to logic.

ASIDE: Consider the difference between using truth tables and actually reasoning. The philosopher Bertrand Russell⁴, trying to pin down what exactly constitutes "knowledge", suggested that he knows that the last name of Britain's prime minister begins with a 'B'. While Tony Blair is prime minister, making Bertrand is correct, we hesitate to say he actually *knows* the fact – he wrote his example when the prime minister was Arthur Balfour⁵ (1902-1905). So while he is correct in a truth-table sense, his reasoning isn't, and we tend to say that he does not actually *know* the prime minister's last initial.

So, no: we're not yet finished with propositional logic. We want to look for (hopefully) more feasible ways to determine whether a formula is a tautology (or, whether two formulas are equivalent). As a clue, we'll try to discover methods which are based on the way we naively approach this. We'll look first at equivalences⁶, and then at inference rules⁷.

⁴<http://www.humanities.mcmaster.ca/~bertrand/>

⁵<http://www.britannia.com/gov/primes/prime39.html>

⁶<http://cnx.rice.edu/content/m10717/latest/>

⁷<http://cnx.rice.edu/content/m10718/latest/>