Deduction

by Daniel Bonevac

Chapter 3 Truth Trees

Truth trees

- Truth trees provide an alternate decision procedure for assessing validity, logical equivalence, satisfiability and other logical properties of sentences and arguments.
- Truth trees are superior to truth tables only in that they are much less tedious for humans to use.
- The basic idea behind a truth tree can be captured by explaining a shorter way of using a truth table to assess validity.

Thinking backwards

- You already know how to use a truth table to determine whether a sentence like this is valid.
- ((p→r) & ¬p) → ¬r
- But you can shorten the task by recalling that the only way for a conditional to be false is for the premises to be true and the conclusion false.
- This means that you really only need to look at the lines of the table that could possibly produce that result. To do this, you simply assign F to the conclusion and work backwards to see what assignments of T to the premises are consistent with the assignment of F to the conclusion.
- In the case of the above formula, we begin by observing that the only way it can be false is if ¬r is F, meaning that r must be T.
- We now look for ways to assign F to antecedent ((p→r) & ¬p) that is compatible with r being T. We know from the truth table for → that if r is T, then (p→r) must be true. So, the only way that the formula ((p→r) & ¬p will be true is if ¬p is F, meaning that p is true.
- But this one interpretation gives us an assignment that renders the formula not valid.

Example of working backwards 1

- Here's how that reasoning works out in a truth table. First assign F to the conclusion.

p	r		((p	→	r)	&	¬ p)	→	¬ r)
									F

Example of working backwards 2

- This means that r has to be T

p	r		((p	→	r)	&	¬ p)	→	¬ r)
	T				T				F

Example of working backwards 3

- This means that (p → r) has to be T

p	r		((p	→	r)	&	¬ p)	→	¬ r)
	T			T	T				F

Example of working backwards 4

- So in order for the conjunction to be T, ¬p must be T, meaning that p is F.

p	r		((p	→	r)	&	¬ p)	→	¬ r)
F	T		F	T	T		T		F

Example of working backwards 5

- Which gives you an F under the →, meaning the formula is not valid.

p	r		((p	→	r)	&	¬ p)	→	¬ r)
F	T		F	T	T		T	F	F

Truth tree terminology

- Truth trees are tree-like structures with nodes of the trees corresponding to particular formulas.
  - Here are some basic facts about truth trees.
    - The branches of the trees develop in accord with specific rules that we will learn.
    - The rules are applied to the formulas on the nodes of the tree.
    - When a rule has been applied, the formula is dispatched with a √ mark. This means that you are done operating on it. A formula without a √ mark is called live.

More truth tree terminology

- We say that a branch of a tree is closed when a formula and it's negation both appear on the branch. We mark a closed branch with an X. Otherwise we say the branch is open.

- We say that a branch is finished when it is (1) closed or (2) only atomic formulas or their negations are live on it.

Using trees to test for validity

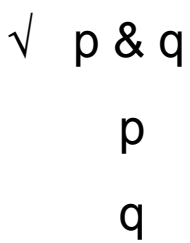
- The most common use of a truth tree is to test an argument for validity. The idea behind this test is simple. Just as with working backwards by the truth table method, we assume the conclusion is false. With the truth tree method, we do not assign the letter F, but rather just negate the conclusion.
- So, if we were going to test the following argument for validity:

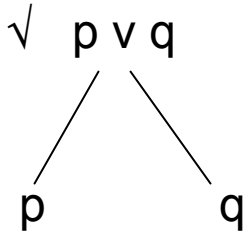
  p v q
  ¬p
  ∴q
- We would start the tree like this.

  p v q
  ¬p
  ∴¬q
- From the definition of validity, you know that if the original argument is valid, then negating the conclusion will produce a contradiction. In truth table language this means that every possible interpretation of the corresponding conditional will be false. In truth tree language it means that every single branch of the corresponding tree will be closed.

Rules for truth trees

- The rules for making trees exploit the difference between conjunctions and disjunctions.
- Statements of the form ($\mathcal{A}$ & $\mathcal{B}$) are true if and only if both $\mathcal{A}$ and $\mathcal{B}$ are true. So the rule for growing a branch with a conjunction is to write both $\mathcal{A}$ and $\mathcal{B}$ on the same branch.
- Statements of the form ($\mathcal{A}$ v $\mathcal{B}$) are true when just one or both of the conjuncts are true. So the rule for growing a branch with a conjunction is to write $\mathcal{A}$ and $\mathcal{B}$ on different branches.





Other truth tree rules

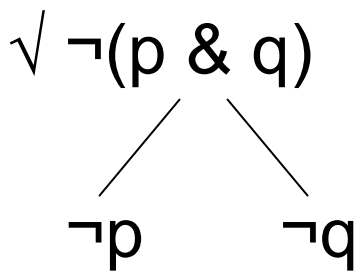
- Almost all of the other rules for truth trees are based on the logical equivalence of formulas with those involving v and &.
- The only exception is the rule for double negation (¬¬) which is simply:

√ ¬¬p

p

The negated conjunction rule ¬&

- A negated conjunction has the form
  - ¬($\mathcal{A}$ & $\mathcal{B}$)
- and it is logically equivalent to
  - ¬$\mathcal{A}$ v ¬ $\mathcal{B}$
- This is intuitive, but you can also prove it to yourself using the truth table method.
- Because of this equivalence, the truth tree rule for negated conjunction is:



The negated disjunction rule ¬v

- A negated disjunction has the form
  - ¬($\mathcal{A}$ v $\mathcal{B}$)
- and it is logically equivalent to
  - ¬$\mathcal{A}$ & ¬ $\mathcal{B}$

- Because of this equivalence, the truth tree rule for negated conjunction is:

√ ¬(p v q)
¬p
¬q

A first truth tree proof.

- Let's use the truth tree method to determine whether the following argument is valid.

  ¬(¬ p v ¬ q)

  ∴ p & q

- We begin by negating the conclusion.

  ¬(¬ p v ¬ q)

  ¬(p & q)

- We can now proceed by applying rules to either one of the formulas. Practically speaking it is always best to work on the non branching formulas first. In this case, that means we apply the rule of ¬v to the premise.

First truth tree proof 2

√ ¬(¬ p v ¬ q)

¬(p & q)

¬¬p

¬¬q

∴ After dispatching the original formula, we now apply the ¬¬ rule to both resulting formulas. We do this now, because ¬¬ is also a non branching rule.

First truth tree proof 3

√ ¬(¬ p v ¬ q)

¬(p & q)

√ ¬¬p

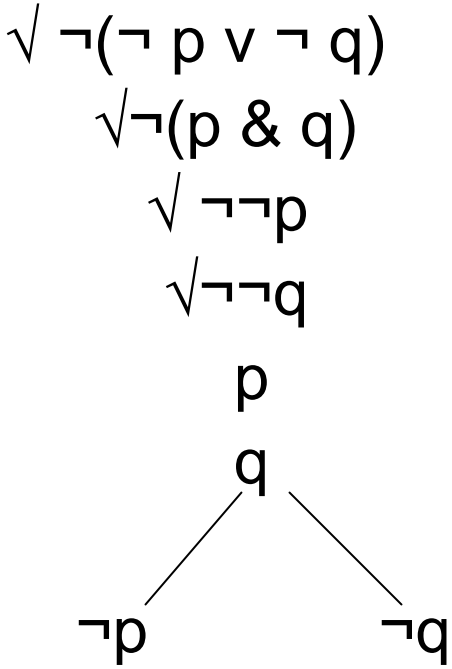
√¬¬q

p

q

∴ Now the only thing left to do is apply ¬& to the negated conclusion.

First truth tree proof 4



∴

∴ We have now dispatched everything but atomic and negated formulas. The only thing left to do is check and see if the branches are closed or open.

Example continued

√ ¬(¬ p v ¬ q)

√¬(p & q)

√ ¬¬p

√¬¬q

p

q

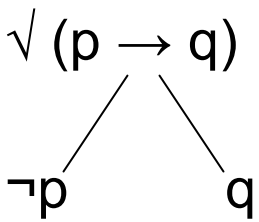
∴ ¬p ¬q

∴ x x

∴ And it's easy to see that in this example both branches close, since tracing backwards, you find contradictory formulas in both cases. Because both branches close, the original argument is valid.

Conditional →

- There are four more rules we need to learn, all related to conditionals and biconditionals.
- The rule for a conditional (→) is based on the fact that
  - $(\mathcal{A} \rightarrow \mathcal{B})$
- is logically equivalent to
  - $\neg\mathcal{A}$ v $\mathcal{B}$
- This is intuitive, as we saw in the previous chapter.
- Because of this equivalence, the truth tree rule for ¬is:



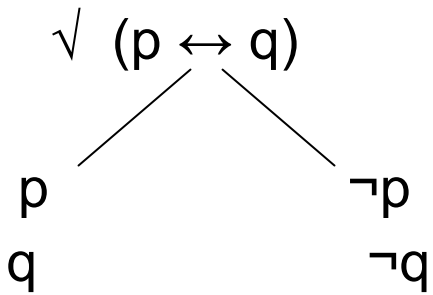
Negated conditional ¬→

- A negated conditional (¬→) has the form
  - $\neg(\mathcal{A} \rightarrow \mathcal{B})$
- and it is logically equivalent to
  - $\mathcal{A}$ & ¬ $\mathcal{B}$
- This is not at all intuitive, but we won't dwell on that now. You can prove it to yourself using the truth table method, or by simply recalling that the only way for a conditional ($\mathcal{A} \rightarrow \mathcal{B}$) to be false (i.e. negated) is for $\mathcal{A}$ to be true and $\mathcal{B}$ to be false.
- Because of this equivalence, the truth tree rule for ¬→ is:

√ ¬(p → q)
p
¬q

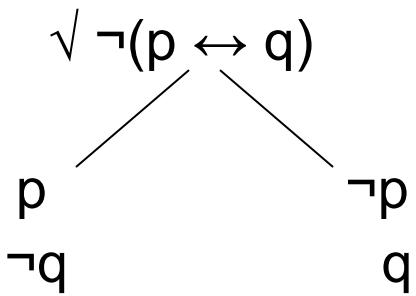
Biconditional ↔

- The rule for the biconditional (↔) based on the slightly more complicated fact that
  - $(\mathcal{A} \leftrightarrow \mathcal{B})$
- is logically equivalent to
  - $(\mathcal{A}$ & $\mathcal{B})$ v $(\neg\mathcal{A}$ & $\neg\mathcal{B})$
- Since the equivalent expression is a disjunction of conjunctions, this is a branching rule with the conjunctions hanging off the ends of the branches like this:
-



Negated biconditional ¬↔

- Our last rule is negated biconditional ¬↔. It is based on the fact that
  - $\neg(\mathcal{A} \leftrightarrow \mathcal{B})$
- is logically equivalent to
  - $(\mathcal{A} \,\&\, \neg\, \mathcal{B})$ v $(\neg\mathcal{A} \,\&\, \mathcal{B})$
- Since the equivalent expression is a disjunction of conjunctions, this is a branching rule with the conjunctions hanging off the ends of the branches like this:
-



Two more details

- There are just a couple of more things you need to know about doing truth trees property:
- (1) When you are decomposing a formula according to a truth tree rule, you must do so on every open branch under the formula. For example, if there are four open branches under the formula, you must apply the rule 4 times. (This is why we always try to apply non branching rules first.)
- (2) Once a contradiction appears on a branch you can close it off. You don't need to continue working on it. This saves a lot of work as well.

More truth tree proofs

- Let's do some more trees in a different environment.