

PY1003 Introduction to Logic
Lecture 4: Trees for Sentential Logic I

What is a truth tree?

- A diagram
- constructed according to certain strict rules
- which can be used to prove something
- about a sentence, a set of sentences, or an argument.

What can truth trees be used to prove?

- That a sentence is or isn't a contradiction
- That a sentence is or isn't a tautology
- That a sentence or set of sentences is or isn't consistent
- That an argument is or isn't valid
- NB: These things are related
 - o A is a tautology iff $\neg A$ is a contradiction
 - o $\{A, B\}$ is consistent iff $A \wedge B$ is consistent (i.e. isn't a contradiction)
 - o $\{A, \neg B\}$ is consistent iff $A \vdash B$ is invalid

But we can use truth tables to prove all those things!

- Truth tables can be inefficient and inelegant
- They involve lots of unnecessary working
- They get very big when we are dealing with lots of atomic sentences

What are the rules for constructing trees?

ROUGH IDEA:

Work downwards from your starting sentence(s), and go **from** a compound sentence **to** the ways in which that sentence could be true.

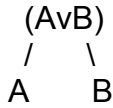
THE RULE FOR ' \wedge '

The only way for a compound sentence of the form $(A \wedge B)$ to be true is if A is true and B is also true. So the rule for $(A \wedge B)$ is:

$$\begin{array}{c} (A \wedge B) \\ | \\ A \\ B \end{array}$$

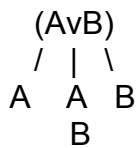
THE RULE FOR 'v'

A sentence of the form $(A \vee B)$ could be true either through A's being true or through B's being true. So the rule for $(A \vee B)$ is:



This is called a 'branching' rule. (The rule for $(A \wedge B)$ is a 'non-branching' rule.)

Query: But 'v' represents an inclusive or. So there are **three** ways for $(A \vee B)$ to be true: either A is true or B is true or they both are. So surely the rule should be:

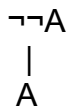


Answer: Not so!

The option represented by the middle branch of this diagram is **already covered** by the options represented in the original diagram.

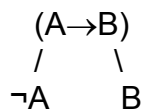
THE RULE FOR '¬¬':

A sentence of the form $\neg\neg A$ could only be true through $\neg A$'s being false. But $\neg A$ can only be false through A's being true. So the rule for $\neg\neg A$ is:



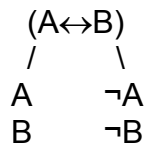
THE RULE FOR '→':

A sentence of the form $(A \rightarrow B)$ is truth-functionally equivalent to (has the same truth table as) a sentence of the form $(\neg A \vee B)$. And $(\neg A \vee B)$ could be true either through $\neg A$'s being true or through B's being true. So the rule for $(A \rightarrow B)$ is:



THE RULE FOR ' \leftrightarrow ':

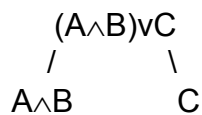
A sentence of the form $(A \leftrightarrow B)$ is truth-functionally equivalent to a sentence of the form $(A \wedge B) \vee (\neg A \wedge \neg B)$. This can be true either through A's being true and B's being true, or through $\neg A$'s being true and $\neg B$'s being true. So the rule for $(A \leftrightarrow B)$ is:



VERY IMPORTANT:
When building trees, you can only apply the rule for the **Main Connective** in the sentence you are dealing with.

Example:

Faced with $(A \wedge B) \vee C$, apply the rule for ' \vee '

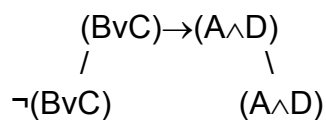


(Note that the rule is just the same, even when the sentences connected by the main connective are themselves compound sentences.)

DO NOT attempt to apply the rule for ' \wedge ' first when faced with this sentence!

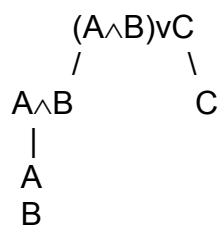
Example:

Faced with $(B \vee C) \rightarrow (A \wedge D)$, apply the rule for ' \rightarrow '

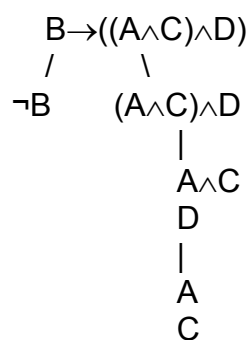


Building onto a tree diagram

Example:



Example:

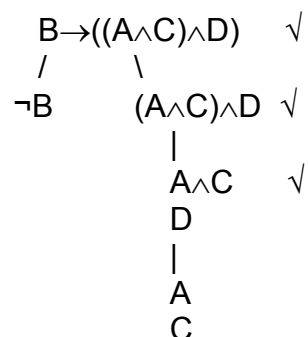


Ticking off 'used up' sentences

Why do this?

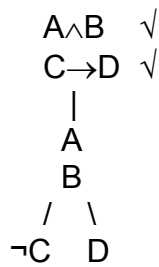
- It helps **you** remember which sentences you have 'used up'.
- It shows **anyone reading the tree** which sentences you have used up. (This will be important later.)

Example:



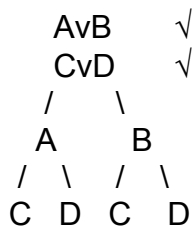
Starting with more than one sentence

Example:



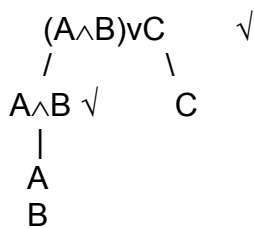
When applying the rule for a sentence, you **MUST** apply it to every branch on which that sentence appears (unless the branch is finished or 'closed', in a sense I'll describe below).

Example:



But this doesn't just mean "apply it to every branch".

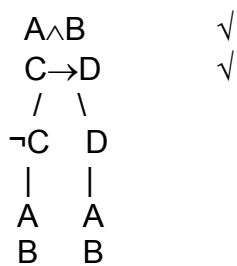
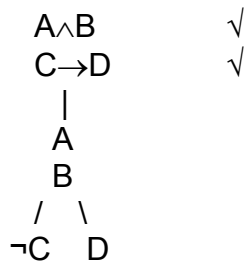
Example:



We don't apply the rule for 'A ∧ B' to the right hand branch, because the sentence 'A ∧ B' does not appear on that branch.

Hint: If faced with the option of applying a branching rule or a non-branching rule, choose the non-branching rule first!

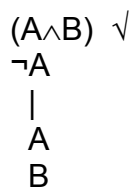
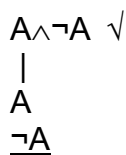
Examples:



What do the branches represent?

- Each branch represents a way for the starting sentence(s) to be true ...
- ... or rather, that is what it represents **assuming it does not close**.
- A closed branch is one that contains both a sentence and its negation.

We display a branch closure as follows:



A closure at the bottom of a branch shows that this branch does **not** represent a way for the starting sentence(s) to be true.

We do not write anything further on a branch once it has closed.

If every branch on a tree is closed
then there is no way for the set of sentences you started with all to be true.
I.e. there is no interpretation which makes them all come out true.

If you have applied all the rules you can and a branch is still open
then there is a way for the set of sentences you started with all to be true.
I.e. there is an interpretation which makes them all come out true.

It is often important to **show** that you have applied as many rules as you can, in order to **show** that there is a way for your starting sentences all to be true. That is why it is important to tick off used-up sentences. If you have ticked off every non-atomic sentence, you've shown that you have applied as many rules as you can.

Reading:
Colin Howson, *Logic With Trees*, chapter 2.