Further Remarks on PL Trees

Validity and Invalidity. An Argument *A* is invalid iff the set of premises Γ is true but the conclusion ϕ is false. To test validity, we check the consistency of Γ and ϕ under the assumption that the formulae in Γ are true but that ϕ is false. Hence, Γ logically entails a falsehood, so moving from the premises to the conclusion does not preserve truth. This is the guiding idea behind the tree method, which was developed in the 1950s by E. W. Beth.[1]

To illustrate the basic method, here is two examples of so-called ‘signed’ trees, where each formula gets a truth value of either ‘T’ or ‘F’. This is because we assume the *falsity* of the conclusion. The tree method is thus negative or indirect: it aims to refute certain assumptions. The task is to find out whether there is a valuation under which the premises are true but the conlusion is false. If this is not possible, *A* is not invalid, and so valid.

Example 1.[2] ‘~(*P* & *Q*)’, ‘(*P* & *R*)’ ⊧ ‘~(*R* ∨ *Q*)’

(1)	~(*P* & *Q*) [T]		Premise: must be true [T]
(2)	*P* & *R* [T]		Premise: must be true [T]
(3)	~(*R* ∨ *Q*) [F]		Conclusion: must be false [F]
(4)	*P* & *Q* [F]		1, remove ‘~’ and invert truth value *Rationale for (4).* If ~ϕ is true (cf. Line (2)), then ϕ is false. Removing ‘~’ changes the truth value.
(5)	*P* [T]		2, &A
(6)	*R* [T]		2, &A
(7)	*R* ∨ *Q* [T]		3, remove ‘~’ and invert truth value
(8)	*P* [F] ⋆	*Q* [F]	4, something like ‘&A’ *Rationale for (8).* The question is, what makes ‘*P* & *Q*’ *false*? A conjunction ϕ & ψ is *false* iff either ϕ or ψ are false (see *Notes and Exercises*, truth tables p. 15.) Hence the branching.
(9)		*R* [T] O *Q* [T] ⋆	7, ∨A

Result. There is an open branch. So there is a valuation such that the premises are true and the conclusion is false. The argument is invalid.

1 Beth, E. W. (1955). Semantic entailment and formal derivability. *Mededeelingen der Koninklijke Nederlandsche Akademie van Wetenschappen, Afd. Letterkunde; nieuwe reeks*, 18, 309–42. Amsterdam: Noord-Hollandsche Uitg. Mij.

2 After Smith, P. (2003), *An Introduction to Formal Logic* (p. 150). Cambridge: Cambridge University Press.

Example 2. '(*P* & (*Q* ∨ *R*))' ⊨ '((*P* & *Q*) ∨ (*P* & *R*))'

(1)	*P* & (*Q* ∨ *R*)	[T]	
(2)	(*P* & *Q*) ∨ (*P* & *R*)	[F]	
(3)	*P*	[T]	1, &A *Rationale for (3)*. ϕ & ψ is true iff both conjuncts are true.
(4)	*Q* ∨ *R*	[T]	1, &A *Note*. To avoid branching, we do not now continue with this formula.
(5)	*P* & *Q*	[F]	2, something like ∨A *Rationale for (5)*. ϕ ∨ ψ is *false* iff both ϕ and ψ are false.
(6)	*P* & *R*	[F]	2, something like ∨A
(7)	*P* [F] ★ / *Q* [F]		5, something like &A
(8)	(under *Q*:) *P* [F] ★ / *R* [F]		6, something like &A
(9)	(under *R*:) *Q* [T] ★ / *R* [T] ★		4, ∨A

Result. Every branch closes. The initial assumptions cannot be satisfied; the premise cannot be true but the conclusion false. So, the argument is valid.

However, it is simpler and more elegant to work with 'unsigned' trees, where *all* truth values are set to 'true'. Given the definition of the negation, we can say: if the conclusion ϕ is false (F), then ~ϕ is true (T). So, instead of assuming the falsity of the conclusion, we assume the truth of its negation. This strategy is at work in the following tree, which is Example 2 again. Is the tree different?

(1)	*P* & (*Q* ∨ *R*)	
(2)	~((*P* & *Q*) ∨ (*P* & *R*))	
(3)	*P*	1, &A
(4)	*Q* ∨ *R*	1, &A
(5)	~(*P* & *Q*)	2, ~∨A
(6)	~(*P* & *R*)	2, ~∨A
(7)	~*P* ★ / ~*Q*	5, ~&A
(8)	(under ~*Q*:) ~*P* ★ / ~*R*	6, ~&A
(9)	(under ~*R*:) *Q* ★ / *R* ★	4, ∨A