Tableau Proofs

Robert Keller

February 2013

Meaning

- “Tableau” (French for “table”)
- Plural “Tableaux” or “Tableaus” (US)
- As used here, as much a **tree** as a table. Several variations exist.
- Variant terminology:

  Analytic tableau

  Semantic tableau

Syntactic or Semantic

- This is a syntactic proof method, with obvious connections to semantics.
- In addition to providing a proof when possible, it will provide a counterexample in the propositional case if the sequent is not provable.
- The version we present is strictly classical logic, although intuitionistic variants are possible.

Proof by Refutation

- Recall:

  A is valid iff ¬A is unsatisfiable.

- A refutation proof proves unsatisfiability.

- Thus it indirectly proves validity of the negation (assuming RAA).

Algorithmic

- For propositional logic, this method is algorithmic: guaranteed to find a proof.

- (For predicate logic, described in the future, it is only semi-algorithmic: If there is a proof, the method can find it. However, it might diverge if there is no proof.)

Method

- **Negate** the formula to be proved (for which validity is to be checked).
- Then perform the tree (tableau) construction to be described. Each step breaks down a working set of formulas into a possibly-larger set of **simpler** formulas, until no further break-down is possible.
- Note: This method can be used to **check** validity, even if some **other** proof method, such as Natural Deduction, is used to **present** the proof.

Why Not Used Universally?

- Although the tableau proof method is very effective, proofs are not usually **presented** this way informally. It is not as “natural”.

- Also, our tableau proofs are not intuitionistic, in that they have the LEM and other rules built-in, in effect.

Tableau Construction

- Start with the formula to be checked for satisfiability as the root.
- Successively “replace” formulas by new ones derived from sub-formulas.
- In some cases, the tree **branches** into two.
- In all cases, the construction will eventually stop.
- At the stopping point, satisfiability is readily determined “by inspection”.

Formulas are classified in one of four ways:

- **Stacking** (Conjunctive): The form is one of:

  $A \wedge B$

  $\neg(A \vee B)$

  $\neg(A \rightarrow B)$

- **Splitting** (Disjunctive): The form is one of:

  $A \vee B$

  $\neg(A \wedge B)$

  $A \rightarrow B$

- $\neg\neg A$: Replace with A. This is sometimes regarded as stacking.

- Other: Either a proposition symbol or its negation. Leave as is.

Stacking rules:
$A \wedge B$, $\neg(A \vee B)$, $\neg(A \rightarrow B)$

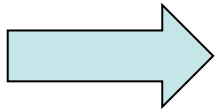
- The formula is **checked off** (removed from further consideration).

- The derived formulas are "**stacked**" in its place on the tree, as follows:

$A \wedge B$	stack A and B
$\neg(A \vee B)$	stack $\neg$A and $\neg$B
$\neg(A \rightarrow B)$	stack A and $\neg$B

Examples of Stacking

$(p \rightarrow q) \wedge (q \rightarrow r)$

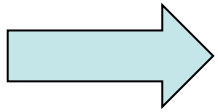


$(p \rightarrow q) \wedge (q \rightarrow r)$✔

$(p \rightarrow q)$

$(q \rightarrow r)$

$\neg(p \vee (q \wedge r))$

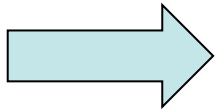


$\neg(p \vee (q \wedge r))$✔

$\neg p$

$\neg(q \wedge r)$

$\neg((p \vee q) \rightarrow r)$



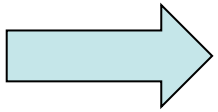
$\neg((p \vee q) \rightarrow r)$✔

$p \vee q$

$\neg r$

¬ ¬ Rule

¬ ¬(p ∨ (q ∧ r))



¬ ¬(p ∨ (q ∧ r))✔

p ∨ (q ∧ r)

Splitting rules: $A \vee B$, $A \rightarrow B$, $\neg(A \wedge B)$

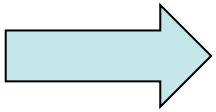
- The formula is **checked off** (removed from further consideration).

- The sub-formulas are "**split**" creating a branch of the tree for each, as follows:

$A \vee B$	split to A and B
$\neg(A \wedge B)$	split to $\neg A$ and $\neg B$
$A \rightarrow B$	split to $\neg A$ and B

Examples of Splitting

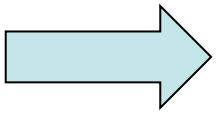
$(p \rightarrow q) \vee (q \rightarrow r)$



$(p \rightarrow q) \vee (q \rightarrow r)$✔

$(p \rightarrow q)$ $\quad$ $(q \rightarrow r)$

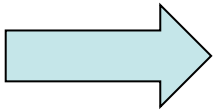
$\neg(p \wedge (q \vee r))$



$\neg(p \wedge (q \vee r))$✔

$\neg p$ $\quad$ $\neg(q \vee r)$

$((p \vee q) \rightarrow r)$



$((p \vee q) \rightarrow r)$✔

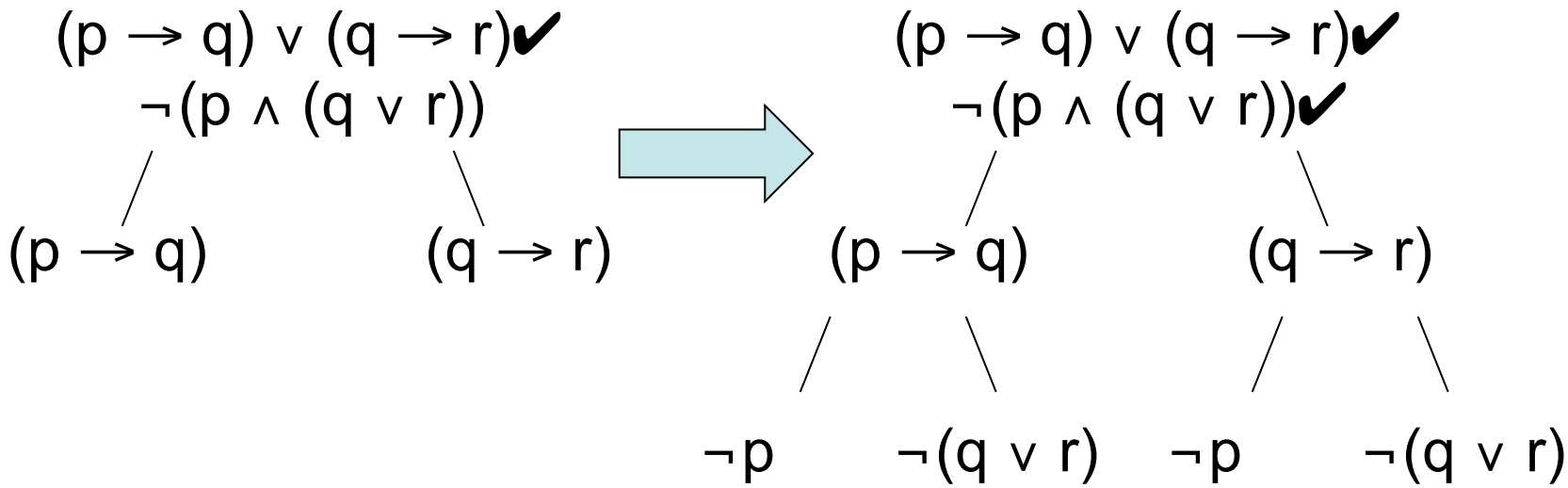
$\neg(p \vee q)$ $\quad$ r

Multiple Splitting

- When a formula splits, the unchecked formulas on the original branch are still active.

- If a formula on the original branch is split, its parts must be added to both new branches.

Multiple Splitting Example



Choosing Formulas

- The formulas can be chosen for stacking or splitting in any order.

- Prefer stacking over splitting, to reduce the complexity of the tree.

Completion

- A tableau construction is **complete** if no further rule application is possible.

Examples

- Check validity of: $p \rightarrow (q \rightarrow p)$
- Negate:

$$\neg(p \rightarrow (q \rightarrow p))$$

Tableau, Starting with Negated Formula

1. $\neg(p \rightarrow (q \rightarrow p))$

Tableau

1.	$\neg(p \rightarrow (q \rightarrow p))$	✓ (stack)
2.	p	1
3.	$\neg(q \rightarrow p)$	1

Tableau

1.	¬(p → (q →p))	✓ (stack)
2.	p	1
3.	¬(q →p)	1 ✓ (stack)
4.	q	3
5.	¬p	3

At this point, the tableau is **complete**.

Closed and Open Tableau

- A **path** from root to leaf in a tableau is **closed** if **both a formula and its negation** appearing on the path.
- We show closed paths by placing an X at the leaf.
- A **tableau** is closed if **all** paths from root to leaves are **closed**.
- A tableau is **open** iff it is not closed.

Main Result

- The root formula is **unsatisfiable** iff the completed tableau is closed.

- (Recall that the root formula is typically the negation of a formula to be proved valid.)

Tableau

1. ¬(p → (q →p))	✓ (stack)
2. p	1
3. ¬(q →p)	1 ✓ (stack)
4. q	3
5. ¬p	3
X(2, 5)	closed by p, ¬p

At this point, the tableau is complete.
There is only one path from root to leaf, and it is closed.
Therefore, the root formula is unsatisfiable.
(Thus the original formula is valid.)

Tableau Example: Negated Formula at Root

1. $\neg((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p))$

Prefer Stacking to Branching

1. $\neg((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p))$ ✓

2. $(p \rightarrow q)$ 1

3. $\neg(\neg q \rightarrow \neg p)$ 1

Stacking

1. $\neg((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p))$ ✓

    2. $(p \rightarrow q)$ 1

    3. $\neg(\neg q \rightarrow \neg p)$ 1✓

    4. $\neg q$ 3

    5. $\neg\neg p$ 3

¬ ¬

1. ¬((p → q) → (¬q → ¬p)) ✓

2. (p → q) 1

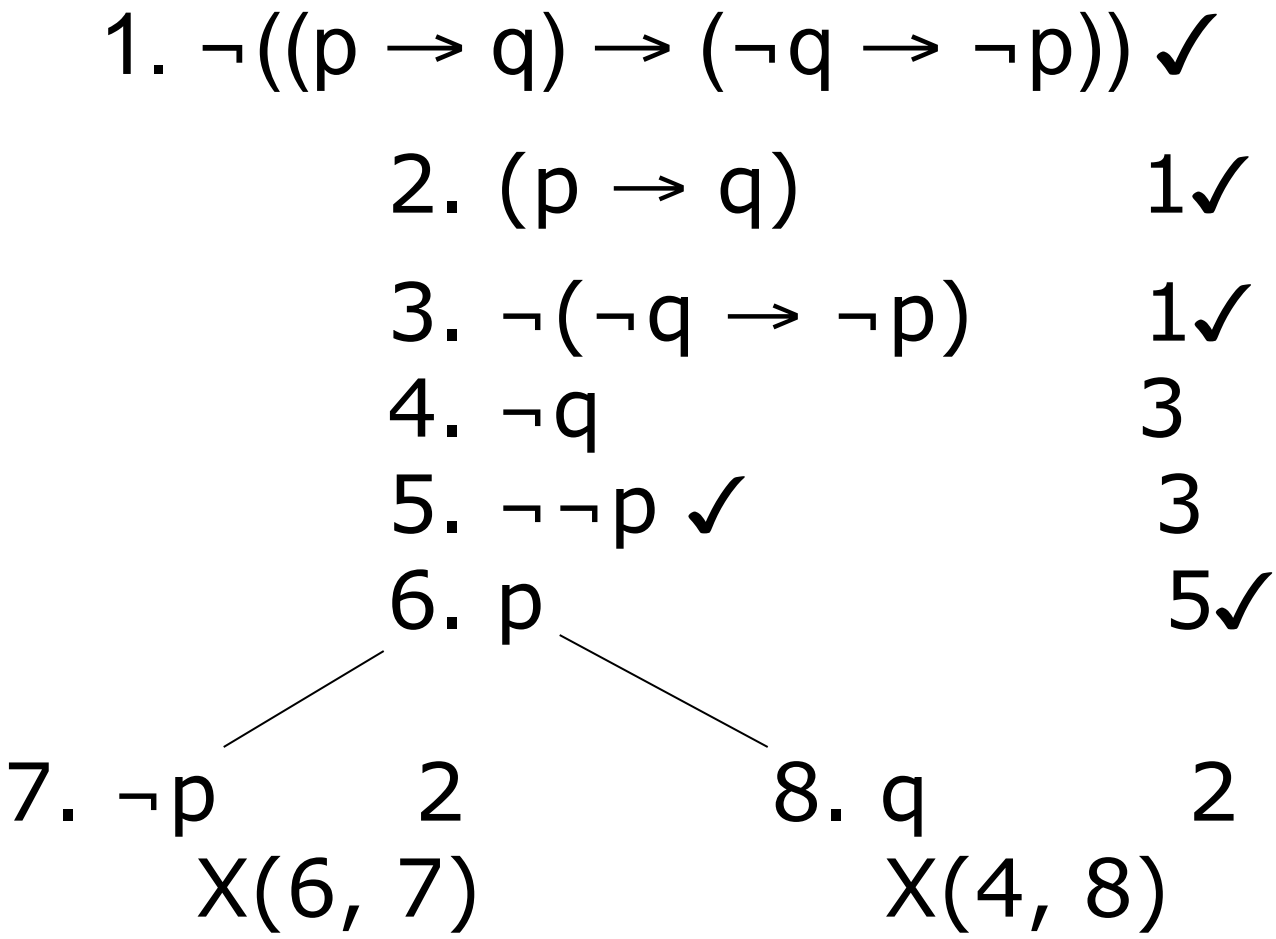
3. ¬(¬q → ¬p) 1✓

4. ¬q 3

5. ¬¬p 3✓

6. p 5

Must split now

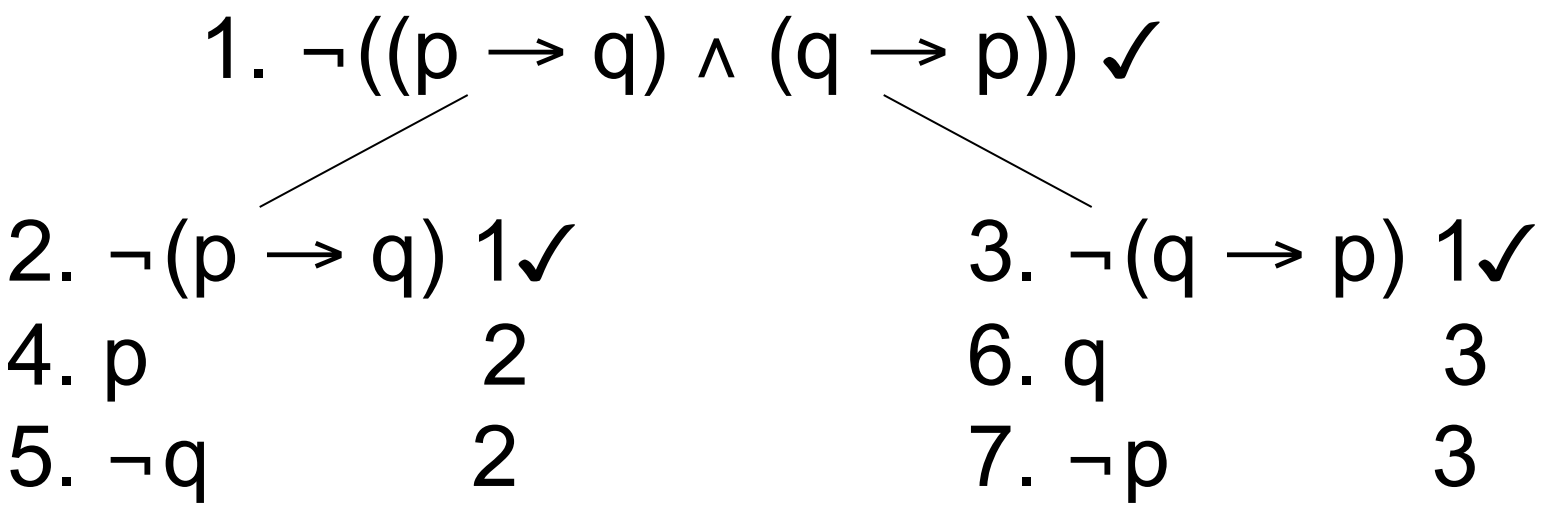


Satisfiable Root

- If the root formula ***is*** **satisfiable**, there is at least **one open path** in the complete tableau.

- Such a **path** can determine a **valuation** v that satisfies the formula:
  - If a proposition symbol occurs **bare**, then v assigns **true** to that symbol.
  - If a proposition symbol occurs **negated**, then v assigns **false** to that symbol.
  - If a proposition symbol does not occur, then its value does not matter in the valuation.

Example



The tableau is complete.
Both paths are open.

One satisfying valuation is $\nu(p) = T$, $\nu(q) = F$.

Another is $\nu(p) = F$, $\nu(q) = T$.

Why the Tableau Works

- Each move **preserves satisfiability** of the original formula along ***its parent*** path Γ.

- **stacking move**, such as $A \wedge B$:
  $\Gamma \cup \{A \wedge B\}$ is satisfiable iff $\Gamma \cup \{A, B\}$ is satisfiable.

- **splitting move**, such as $A \vee B$:
  $\Gamma \cup \{A \vee B\}$ is satisfiable iff $\Gamma \cup \{A\}$ is satisfiable **or** $\Gamma \cup \{B\}$ is satisfiable.

- $\Gamma \cup \{\neg\neg A\}$ is satisfiable iff $\Gamma \cup \{A\}$ is.

- $\Gamma \cup \{A, \neg A\}$ is not satisfiable (path closes).

$\Gamma \cup \{A \wedge B\}$ is satisfiable
iff $\Gamma \cup \{A, B\}$ is satisfiable

The following are equivalent:

- ν satisfies $\Gamma \cup \{A \wedge B\}$.
- ν satisfies Γ and ν satisfies $A \wedge B$.
- ν satisfies Γ and ν satisfies A and ν satisfies B.
- ν satisfies $\Gamma \cup \{A, B\}$.

$\Gamma \cup \{A \vee B\}$ is satisfiable
iff $\Gamma \cup \{A\}$ is satisfiable
or $\Gamma \cup \{B\}$ is satisfiable

The following are equivalent:

- ν satisfies $\Gamma \cup \{A \vee B\}$.
- ν satisfies Γ and ν satisfies $A \vee B$.
- ν satisfies Γ and (ν satisfies A or ν satisfies B).
- ν satisfies $\Gamma \cup \{A\}$ or ν satisfies $\Gamma \cup \{B\}$.

Why the Tableau Works

$A \wedge B$ satisfiable

A satisfiable
B satisfiable

$A \vee B$ satisfiable

$A \wedge B$ unsatisfiable

A unsatisfiable
B or unsatisfiable

$A \vee B$ unsatisfiable

Other rules than $\wedge$, $\vee$

The other rules are justified by applying DeMorgan's equivalences with the basic rules.

$\neg(A \vee B)$ is equivalent to $\neg A \wedge \neg B$ (stack)

$\neg(A \wedge B)$ is equivalent to $\neg A \vee \neg B$ (split)

$A \rightarrow B$ is equivalent to $\neg A \vee B$ (split)

$\neg(A \rightarrow B)$ is equivalent to $A \wedge \neg B$ (stack)

Block Tableaux

- Block tableau are based on a set of sets (“blocks”).
- Instead of using the **tree** formation, it is possible to carry on **each path** all un-checked formulas as a **block**.
- When a stacking rule is used, the formula is replaced **within the block** with the two constituent formulas.
- When a splitting rule is used, the **block itself splits**. The formula is replaced with one of the two constituent formulas in each block.
- If **each block** contains a formula and its negation, the original formula is unsatisfiable.

Block Tableaux Advantage

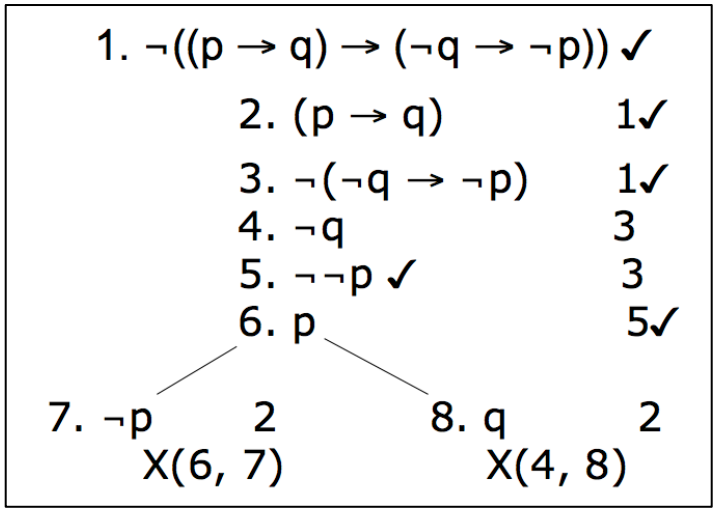
- Easier to program on a computer.
- Don’t need to maintain trees, just a set of sets.
- A connection with “Sequent Calculus” will be shown.

Block Tableau for an Earlier Example

- $\{\neg((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p))\}$ 1 block
- $\{p \rightarrow q, \neg(\neg q \rightarrow \neg p)\}$ (stack)
- $\{p \rightarrow q, \neg q, \neg\neg p\}$ (stack)
- $\{p \rightarrow q, \neg q, p\}$ (split)
- $\{\neg p, \neg q, p\}, \{q, \neg q, p\}$

X X

unsatisfiable



Another Block Tableau Example

- $\{(p \vee q) \wedge (\neg p \wedge \neg q)\}$ determine unsat
- $\{(p \vee q), (\neg p \wedge \neg q)\}$
- $\{(p \vee q), \neg p, \neg q\}$
- $\{p, \neg p, \neg q\}$ $\{q, \neg p, \neg q\}$

  X X

  unsatisfiable (all blocks closed)

Another Block Tableau Example

- $\{p \wedge (\neg q \vee \neg p)\}$ determine unsat
- $\{p, (\neg q \vee \neg p)\}$
- $\{p, \neg q\}$ $\{p, \neg p\}$
  open X

The formula is satisfied by p = T, q = F.

Proving that the tableau construction terminates (Ben-Ari)

- Each move deconstructs a formula on a path.

- We can create a **metric** W for a set that:

  has a **positive** integer value,

  (thus cannot decrease indefinitely)

  always decreases with each move

  W = 3*#binary operators + # negations

Metric Example

- $W(\{\neg((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p))\}) =$
  $3*3+3*1 = 12$

- $W(\{ (p \rightarrow q), \neg(\neg q \rightarrow \neg p)\} =$
  $2*3+3*1 = 9$

Ben-Ari, p 33

Proof: Let $\mathcal{T}$ be the tableau for formula A at any stage of its construction and let us assume for now that $\leftrightarrow$ and $\oplus$ do not occur in the formula A. For any leaf $l \in T$, let $b(l)$ be the number of binary operators in formulas in $U(l)$ and let $n(l)$ be the number of negations in $U(l)$. Define

$$W(l) = 3b(l) + n(l).$$

We claim that any step of the construction creates a new node l' or nodes l', l'' such that $W(l) > W(l')$ and $W(l) > W(l'')$. For example, if we apply the α rule to $\neg(A_1 \vee A_2)$ to obtain $\neg A_1$ and $\neg A_2$, then

$$W(l) = k + 3 \cdot 1 + 1 \; > \; k + 3 \cdot 0 + 2 = W(l'),$$

where k is the sum of the number of operators in A_1 and A_2. Obviously, $W(l) \geq 0$, so no branch of $\mathcal{T}$ can be extended indefinitely. We leave it to the reader to check the correctness of the claim for the other rules and to modify the definition of $W(l)$ in the case that A contains $\leftrightarrow$ or $\oplus$. ∎