Tableau Proofs

Robert Keller

January 2011

Meaning

- “Tableau” (French for “table”)
- “Tableaux” plural
- As used here, more of a **tree** than 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.

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, therefore this is classical, rather than constructive logic). However, constructive variants are possible.

Algorithmic

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

- (For predicate logic, it is only semi-algorithmic).

Method

- **Negate** the formula to be proved (for which validity is to be checked).
- Then undertake the tree (tableau) construction to be described. Each step breaks down a working set of formulas into a possibly-larger set of **simpler** formulas.
- Note: This method can be used to **check** validity, even if some **other** proof method, such as Natural Deduction, is used to **present** a 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 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:

- α (Conjunctive, stacking): The form is one of:

  $A \wedge B$

  $\neg(A \vee B)$

  $\neg(A \rightarrow B)$

- β (Disjunctive, splitting): 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 an α also.

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

α 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 $\neg B$

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

- 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

Completion

- A tableau construction is **complete** if no further rule applications are 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
3. $\neg(q \rightarrow p)$

Tableau

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

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

Closed and Open Tableau

- A **path** from root to leaf in a tableau is **closed** if there is **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 complete tableau is closed.

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

Tableau

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

   X(4, 5) closed by p, $\neg p$

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.
(The original formula is thus 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. $(p \rightarrow q)$

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

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

2. $(p \rightarrow q)$

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

4. $\neg q$

5. $\neg\neg p$

Splitting is the Only Option Here

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

2. $(p \rightarrow q)$

3. $\neg(\neg q \rightarrow \neg p)$ ✓
4. $\neg q$
5. $\neg\neg p$ ✓
6. p

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

2. $(p \rightarrow q)$ ✓

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

4. $\neg q$

5. $\neg\neg p$ ✓

6. p

Split from line 2.

7. $\neg p$

X(6, 7)

8. q

X(4, 8)

Satisfiable Root

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

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

Example

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

2. $\neg(p \rightarrow q)$ ✓
4. p
5. $\neg q$

3. $\neg(q \rightarrow p)$ ✓
6. q
7. $\neg p$

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 may add a paths, but **preserves satisfiability** of the original formula along ***some*** path Γ.
- α **move** (stacking), such as $A \wedge B$:
  $\Gamma \cup \{A \wedge B\}$ is satisfiable iff $\Gamma \cup \{A, B\}$ is satisfiable.

- β **move** (splitting), 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.

- $\Gamma \cup \{p, \neg p\}$ 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

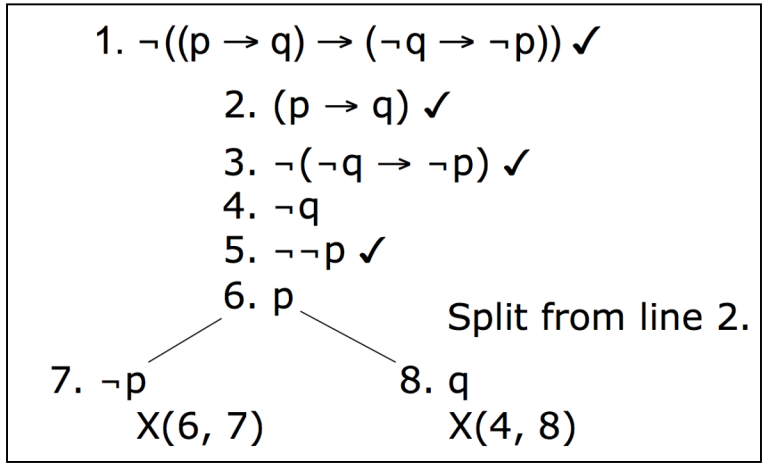
Block Tableaux

- Instead of using the tree formation, it is possible to carry on **each path** all un-checked formulas as a **set**.
- When an α rule is used, the formula is replaced **within the set** with the two constituent formulas.
- When a β rule is used, the **set splits**. The formula is replaced with one of the two constituent formulas in each set.

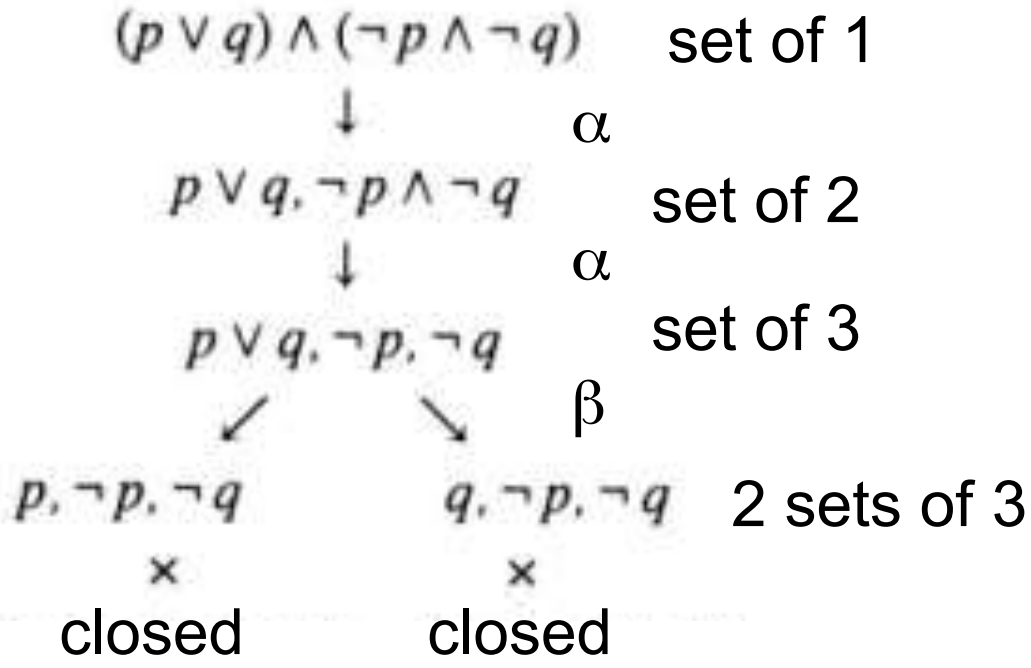
Block Tableau Example

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

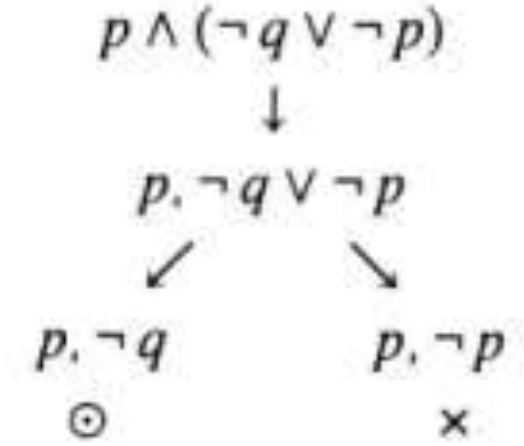
  X X



Example: Ben-Ari, p. 31



Example: Ben-Ari, p. 31



α

β

open

closed

Ben-Ari's Tableau Rules

α	α_1	α_2
$\neg\neg A_1$	A_1	
$A_1 \wedge A_2$	A_1	A_2
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$
$\neg(A_1 \rightarrow A_2)$	A_1	$\neg A_2$
$\neg(A_1 \uparrow A_2)$	A_1	A_2
$A_1 \downarrow A_2$	$\neg A_1$	$\neg A_2$
$A_1 \leftrightarrow A_2$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$
$\neg(A_1 \oplus A_2)$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$

β	β_1	β_2
$\neg(B_1 \wedge B_2)$	$\neg B_1$	$\neg B_2$
$B_1 \vee B_2$	B_1	B_2
$B_1 \rightarrow B_2$	$\neg B_1$	B_2
$B_1 \uparrow B_2$	$\neg B_1$	$\neg B_2$
$\neg(B_1 \downarrow B_2)$	B_1	B_2
$\neg(B_1 \leftrightarrow B_2)$	$\neg(B_1 \rightarrow B_2)$	$\neg(B_2 \rightarrow B_1)$
$B_1 \oplus B_2$	$\neg(B_1 \rightarrow B_2)$	$\neg(B_2 \rightarrow B_1)$

Why does the tableau construction terminate?

- 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 decrease 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 $\oplus$. ∎