Introduction to Logic

Jake Chandler

Tableaux for Sentential Logic

LA TROBE UNIVERSITY

Introducing tableaux

Last week

- The idea of truth-functionality of connectives
- The concept of "expressive completeness"
- The issue of the truth table for conditionals
- The truth table method for determining validity of sentential forms

Too many atoms spoil the broth

- The truth table method:

  For each possible assignment of values for the atoms, check whether the rules for the connectives force the premises to be true and the conclusion false.
- But things quickly get out of hand: 8 atoms $\Rightarrow 2^8 = 256$ rows!!
- Observation: we waste time by having to consider many assignments of values that turn out not to yield countermodels.
- The semantic tableau, or tree, method avoids this:

  Assume the premises to be true and the conclusion to be false and check whether the rules for the connectives permit a compatible assignment of values for the atoms.
- This is a *top-down* rather than *bottom-up* approach.

Tableaux: the idea

- For tableaux, we in fact use the following equivalent procedure:

  Assume both the premises and the negation of the conclusion to be true and check whether the rules for the connectives permit a compatible assignment of values for the atoms.
- So how does this work?

Tableaux by example: first case

Checking whether or not $p \supset q, r \vee \sim q \models (p \vee q) \supset r$

- A tableau consists of branches that each correspond to an attempted valuation (counterexample)
- Writing sentence P on a branch represents the corresponding attempted valuation assigning 1 to P

Tableaux by example: first case

Checking whether or not $p \supset q, r \vee \sim q \models (p \vee q) \supset r$

- All branches will contain the premises and the negation of the conclusion
- We write these down at the root of the tableau

$$p \supset q$$
$$r \vee \sim q$$
$$\sim ((p \vee q) \supset r)$$

Tableaux by example: first case

Checking whether or not $p \supset q, r \vee \sim q \models (p \vee q) \supset r$

- Any branch that contains $\sim ((p \vee q) \supset r)$ will also contain both $p \vee q$ and $\sim r$
- We tick off $\sim ((p \vee q) \supset r)$ with ✓ to shows it's been processed and extend the tableau by adding $p \vee q$ and $\sim r$ to the branch

$$p \supset q$$
$$r \vee \sim q$$
$$\sim ((p \vee q) \supset r) \checkmark$$
$$|$$
$$p \vee q$$
$$\sim r$$

Tableaux by example: first case

Checking whether or not $p \supset q, r \vee \sim q \models (p \vee q) \supset r$

- Any branch that contains $r \vee \sim q$ will either contain r or contain $\sim q$
- We tick off $r \vee \sim q$ and extend the tableau by splitting off into two branches, one with r and one with $\sim q$

$p \supset q$
$r \vee \sim q$ ✓
$\sim ((p \vee q) \supset r)$ ✓
$p \vee q$
$\sim r$
r $\quad$ $\sim q$

Tableaux by example: first case

Checking whether or not $p \supset q, r \vee \sim q \models (p \vee q) \supset r$

- Note that one branch contains both r and $\sim r$
- But no valuation can assign 1 to both! This is a dead end.
- We say that the branch closes. Closed branches are marked by a × and represent failed attempts to construct counterexamples

$p \supset q$
$r \vee \sim q$ ✓
$\sim ((p \vee q) \supset r)$ ✓
$p \vee q$
$\sim r$
r $\quad$ $\sim q$
×

Tableaux by example: first case

Checking whether or not $p \supset q, r \vee \sim q \models (p \vee q) \supset r$

- Any branch that contains $p \supset q$ will either contain $\sim p$ or contain q
- We tick off $p \supset q$ and extend the tableau accordingly.

$p \supset q$ ✓
$r \vee \sim q$ ✓
$\sim ((p \vee q) \supset r)$ ✓
$p \vee q$
$\sim r$
r $\quad$ $\sim q$
×
$\sim p$ $\quad$ q

Tableaux by example: first case

Checking whether or not $p \supset q, r \vee \sim q \models (p \vee q) \supset r$

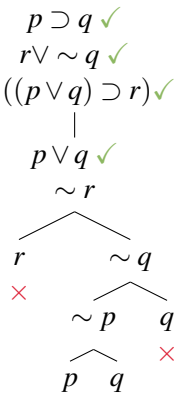
- Again, we have a dead end: one branch contains both $\sim q$ and q
- We close the branch with a ×

$p \supset q$ ✓
$r \vee \sim q$ ✓
$\sim ((p \vee q) \supset r)$ ✓
$p \vee q$
$\sim r$
r $\quad$ $\sim q$
×
$\sim p$ $\quad$ q
×

Tableaux by example: first case

Checking whether or not $p \supset q, r \vee \sim q \models (p \vee q) \supset r$

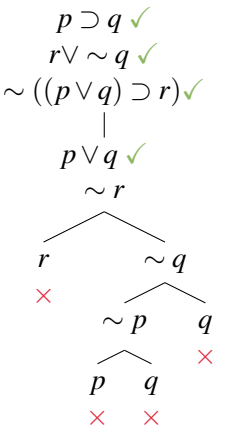
- Any branch that contains $p \vee q$ will either contain p or contain q
- We tick off $p \vee q$ and extend the tableau accordingly



Tableaux by example: first case

Checking whether or not $p \supset q, r \vee \sim q \models (p \vee q) \supset r$

- The remaining branches contain (i) $\sim p$ and p and (ii) $\sim q$ and q, respectively. They both close
- All branches are now closed: the tableau itself is said to be closed
- There is no valuation consistent with the truth of the premises and of the negation of the conclusion: the argument is *valid*



Tableaux by example: second case

Checking whether or not $\models ((p \supset q) \& q) \supset p$

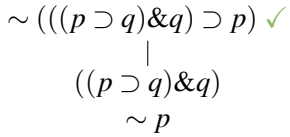
- All branches will contain $\sim(((p \supset q) \& q) \supset p)$, so we write it down

$\sim(((p \supset q) \& q) \supset p)$

Tableaux by example: second case

Checking whether or not $\models ((p \supset q) \& q) \supset p$

- Any branch that contains $\sim(((p \supset q) \& q) \supset p)$ will contain both $((p \supset q) \& q)$ and $\sim p$
- We tick off $\sim(((p \supset q) \& q) \supset p)$ and extend the tableau accordingly



Tableaux by example: second case

Checking whether or not $\models ((p \supset q)\&q) \supset p$

- Any branch that contains $((p \supset q)\&q)$ will contain both $p \supset q$ and q
- We tick off $((p \supset q)\&q)$ and extend the tableau accordingly

$\sim (((p \supset q)\&q) \supset p)$ ✓
|
$((p \supset q)\&q)$ ✓
$\sim p$
|
$p \supset q$
q

Tableaux by example: second case

Checking whether or not $\models ((p \supset q)\&q) \supset p$

- Any branch that contains $p \supset q$ will either contain $\sim p$ and contain q
- We tick off $p \supset q$ and extend the tableau accordingly

$\sim (((p \supset q)\&q) \supset p)$ ✓
|
$((p \supset q)\&q)$ ✓
$\sim p$
|
$p \supset q$ ✓
q
$\sim p$ q

Tableaux by example: second case

Checking whether or not $\models ((p \supset q)\&q) \supset p$

- We are left with two branches, both containing $\sim p$ and q and no complex sentences to process
- These branches are open. We mark them with a ↑
- There exists a valuation that satisfies the negation of the sentence: the sentence is *not a tautology*

$\sim (((p \supset q)\&q) \supset p)$ ✓
|
$((p \supset q)\&q)$ ✓
$\sim p$
|
$p \supset q$ ✓
q
$\sim p$ q
↑ ↑

Introduction to Logic

Jake Chandler

Tableaux for Sentential Logic

Tableaux: second pass

General terminology and concepts

- A tableau for a set of sentences Γ is an inverted tree, such that:
  (i) The members of Γ lie at at its root
  (ii) Any further sentence that is on one of its branches is derived from some other sentence further up that branch, in accordance with an appropriate resolving rule.
- A branch is closed iff it contains both a sentence and its negation.
- A branch is open iff it is not closed.
- A tableau is closed iff all of its branches are closed.
- A tableau is open iff at least one of its branches is open.
- A tableau is completed iff no resolving rule can be applied to any sentences on *open* branches.

Resolving rules: overview

- We have 4 'binary' connectives (connectives with 2 inputs)
- We actually need 2 rules for each of these.
  - one for sentences of the form $P \bullet Q$,
  - one for sentences of the form $\sim (P \bullet Q)$.
- That's 8 rules.
- We also have 1 'unary' connective (connective with 1 input)
- Grand total: 9 rules
- Warning: there are two *serious* typos in the textbook here!!

Resolving rules: conjunctions

- To resolve a sentence of the form $P\&Q$, extend any open branch in which the sentence occurs with a single branch segment containing both P and Q.

$$(P\&Q)$$
$$|$$
$$P$$
$$Q$$

Resolving rules: disjunctions

- To resolve a sentence of the form $P \vee Q$, extend any open branch in which the sentence occurs with two branch segments, one containing P and one containing Q.

$$(P \vee Q)$$
$$P \quad Q$$

Resolving rules: negated disjunctions

- Wrongly stated in the textbook!
- To resolve a sentence of the form $\sim (P \vee Q)$, extend any open branch in which the sentence occurs with a single branch segment containing both $\sim P$ and $\sim Q$.

$$\sim (P \vee Q)$$
$$\sim P$$
$$\sim Q$$

- Rationale: we exploit the equivalence
  $\sim (P \vee Q) \Leftrightarrow \sim P \& \sim Q$

Resolving rules: negated conjunctions

- To resolve a sentence of the form $\sim (P \& Q)$, extend any open branch in which the sentence occurs with two new branch segments, one containing $\sim P$ and one containing $\sim Q$.

$$\sim (P \& Q)$$
$$\sim P \quad \sim Q$$

- Rationale: we exploit the equivalence
  $\sim (P \& Q) \Leftrightarrow \sim P \vee \sim Q$

Resolving rules: conditionals

- To resolve a sentence of the form $P \supset Q$, extend any open branch in which the sentence occurs with two new branch segments, one containing $\sim P$ and one containing Q.

$$(P \supset Q)$$
$$\sim P \quad Q$$

- Rationale: we exploit the equivalence
  $P \supset Q \Leftrightarrow \sim P \vee Q$

Resolving rules: negated conditionals

- Wrongly stated in the textbook!
- To resolve a sentence of the form $\sim (P \supset Q)$, extend any open branch in which the sentence occurs with a single branch segment containing both P and $\sim Q$.

$$\begin{array}{c} \sim (P \supset Q) \\ | \\ P \\ \sim Q \end{array}$$

- Rationale: we exploit the equivalence
  $\sim (P \supset Q) \Leftrightarrow P \& \sim Q$

Resolving rules: biconditionals

- To resolve a sentence of the form $P \equiv Q$, extend any open branch in which the sentence occurs with two new branch segments, one containing both P and Q and one containing both $\sim P$ and $\sim Q$.

$$\begin{array}{cc} (P \equiv Q) & \\ \diagup \quad \diagdown & \\ P & \sim P \\ Q & \sim Q \end{array}$$

Resolving rules: negated biconditionals

- To resolve a sentence of the form $\sim (P \equiv Q)$, extend any open branch in which the sentence occurs with two new branch segments, one containing both $\sim P$ and Q and one containing both P and $\sim Q$.

$$\begin{array}{cc} \sim (P \equiv Q) & \\ \diagup \quad \diagdown & \\ \sim P & P \\ Q & \sim Q \end{array}$$

- Rationale: we exploit the equivalence
  $\sim (P \equiv Q) \Leftrightarrow (\sim P \& Q) \vee (P \& \sim Q)$

Resolving rules: double negations

- To resolve a sentence of the form $\sim\sim P$, extend any open branch in which the sentence occurs with a single branch segment containing P.

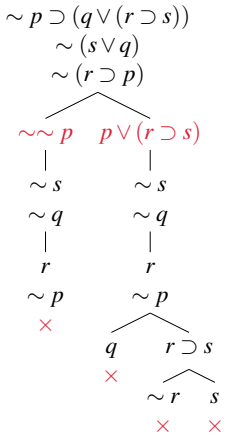
$$\begin{array}{c} \sim\sim Q \\ | \\ Q \end{array}$$

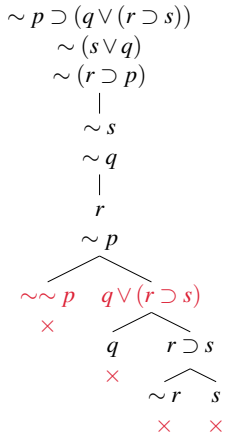
Rule application order

- Some good news:
  - (i) If you keep applying the rules, in *whatever order*, you are guaranteed to obtain a completed tableau in a finite number of steps.
  - (ii) Every completed tableau yields the same verdict: If one completed tableau for Γ is closed (open), then *every* completed tableau for Γ is closed (open).
- Some further good news:
  - (iii) You can reduce this finite number of steps by following some guidelines regarding the order of application of the rules.

Tip: early vs late branching

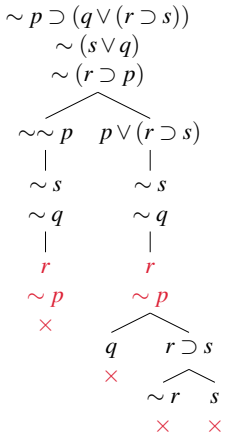
- Tip#1: prioritize non-branching rules.

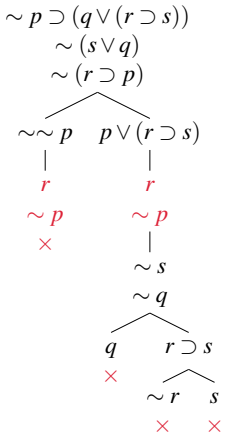




Tip: early vs late closure

- Tip #2: prioritize rules that immediately close branches.





Introduction to Logic

Jake Chandler

Tableaux for Sentential Logic

Soundness and completeness

Some notation

- There is a special symbol $\vdash$ to talk about tableaux for a set of sentences Γ.
- We write $\Gamma \vdash$ to indicate that there exists a closed completed tableau for Γ.
- We write $\Gamma \nvdash$ to indicate that there exists an open completed tableau for Γ.
- We write $\Gamma \vdash C$ to abbreviate $\Gamma \cup \{\sim C\} \vdash$.

  (Where $\Gamma \cup \{\sim C\}$ denotes the set whose members include those of Γ, plus $\sim C$.)
- We write $\Gamma \nvdash C$ to abbreviate $\Gamma \cup \{\sim C\} \nvdash$.

Introducing soundness and completeness

- The claim that the tableau proof method can be used to test for validity breaks down into two subclaims.

  Soundness: If $\Gamma \vdash C$, then $\Gamma \models C$
  If there exists a closed completed tableau for $\Gamma \cup \{\sim C\}$, then the argument from Γ to C is valid.

  Completeness: If $\Gamma \models C$, then $\Gamma \vdash C$
  If the argument from Γ to C is valid, then there exists a closed completed tableau for $\Gamma \cup \{\sim C\}$.
- Since we are also guaranteed a completed tableau in a finite number of steps, our method decides validity

Introducing soundness and completeness (ctd.)

- Note that I haven't given you a rigorous *proof* that either of these subclaims are true.
- I will give you a brief sketch of the proof for soundness.
- For the full version, as well as for the proof for completeness, see Restall, pp. 69–74.
- Regarding soundness, we want to show that if $\Delta \vdash$ then $\Delta \models$ (where $\Delta = \Gamma \cup \{\sim C\}$).
- Proof strategy: We assume, for sake of argument, that $\Delta \vdash$ but $\Delta \not\models$ and show that this would lead to a contradiction.
- Upshot: It cannot be the case that $\Delta \vdash$ but $\Delta \not\models$; in other words, if $\Delta \vdash$, then $\Delta \models$.

Soundness: proof sketch

(1) $\Delta \vdash$ but $\Delta \not\models$. *(Assumption)*

(2) There is a valuation, call it v, that satisfies all sentences in Δ. *(From the 2nd conjunct of (1))*

(3) If v satisfies a sentence P, then it satisfies the sentences in at least one of the branch segments resulting from the resolution of P *(From rules and tables for connectives)*

(4) Every completed tree for Δ has a branch, call it b, such that all sentences on b are satisfied by v. *(From (2) and (3))*

(5) There is no P such that $v(P) = v(\sim P) = 1$ *(From table for $\sim$)*

(6) Branch b is open. *(From (5))*

(7) Every completed tree for Δ has an open branch (namely b). *(From (4) and (6))*

(8) $\Delta \nvdash$, in contradiction with (1). *(From (7))*