4. Truth Trees

4. Truth Trees

a. Why Trees?

Why Trees?

- Truth tables become a bit tedious to work with

Why Trees?

- Truth tables become a bit tedious to work with

- We would like a more streamlined method for checking INconsistency (i.e. UNsatisfiability) and validity

Why Trees?

- Truth tables become a bit tedious to work with
- We would like a more streamlined method for checking INconsistency (i.e. UNsatisfiability) and validity
- Trees are often faster and have less 'irrelevant' information

Why not 'Partial Truth Tables' (PTTs)?

- Couldn't we just look at the 'relevant' lines in a truth table?

Why not ‘Partial Truth Tables’ (PTTs)?

- Couldn’t we just look at the ‘relevant’ lines in a truth table?
- e.g. are the following sentences consistent?

Why not 'Partial Truth Tables' (PTTs)?

- Couldn't we just look at the 'relevant' lines in a truth table?
- e.g. are the following sentences consistent?
- $P \,\&\, Q$; $\sim(P \supset \sim Q)$; $(S \,\&\, R) \vee \sim J$

Why not 'Partial Truth Tables' (PTTs)?

- Couldn't we just look at the 'relevant' lines in a truth table?
- e.g. are the following sentences consistent?
- $P \,\&\, Q$; $\sim(P \supset \sim Q)$; $(S \,\&\, R) \vee \sim J$
- What about $P \,\&\, Q$; $\sim(P \supset \sim Q)$; $(S \,\&\, R) \vee \sim J$; $\sim(S \vee R) \,\&\, J$

Why not 'Partial Truth Tables' (PTTs)?

- Couldn't we just look at the 'relevant' lines in a truth table?
- e.g. are the following sentences consistent?
- $P \& Q$; $\sim(P \supset \sim Q)$; $(S \& R) \vee \sim J$
- What about $P \& Q$; $\sim(P \supset \sim Q)$; $(S \& R) \vee \sim J$; $\sim(S \vee R) \& J$
    - Showing this formally would require a 32 row truth table...

Why not ‘Partial Truth Tables’ (PTTs)?

- Couldn’t we just look at the ‘relevant’ lines in a truth table?
- e.g. are the following sentences consistent?
- $P \,\&\, Q$; $\sim(P \supset \sim Q)$; $(S \,\&\, R) \vee \sim J$
- What about $P \,\&\, Q$; $\sim(P \supset \sim Q)$; $(S \,\&\, R) \vee \sim J$; $\sim(S \vee R) \,\&\, J$
  - Showing this formally would require a 32 row truth table...
  - But the answer is clear from reasoning about truth conditions

Where Partial Truth Tables shine

The method of partial truth tables is already rigorous in cases where a single truth value assignment (TVA) suffices for an answer:

1. Consistent set of sentences:

Where Partial Truth Tables shine

The method of partial truth tables is already rigorous in cases where a single truth value assignment (TVA) suffices for an answer:

1. Consistent set of sentences:
    - Sufficient to find one TVA on which each sentence is true

Where Partial Truth Tables shine

The method of partial truth tables is already rigorous in cases where a single truth value assignment (TVA) suffices for an answer:

1. Consistent set of sentences:
    - Sufficient to find one TVA on which each sentence is true
2. Invalid argument:

Where Partial Truth Tables shine

The method of partial truth tables is already rigorous in cases where a single truth value assignment (TVA) suffices for an answer:

1. Consistent set of sentences:
    - Sufficient to find one TVA on which each sentence is true
2. Invalid argument:
    - Find one TVA where premises are true but conclusion is false

Where Partial Truth Tables shine

The method of partial truth tables is already rigorous in cases where a single truth value assignment (TVA) suffices for an answer:

1. Consistent set of sentences:
    - Sufficient to find one TVA on which each sentence is true
2. Invalid argument:
    - Find one TVA where premises are true but conclusion is false
3. Logically inequivalent sentences:

Where Partial Truth Tables shine

The method of partial truth tables is already rigorous in cases where a single truth value assignment (TVA) suffices for an answer:

1. Consistent set of sentences:
    - Sufficient to find one TVA on which each sentence is true
2. Invalid argument:
    - Find one TVA where premises are true but conclusion is false
3. Logically inequivalent sentences:
    - Suffices to find one TVA where they differ in truth value

Where Partial Truth Tables Falter

So far, our only rigorous method for answering some questions *requires* a complete truth table, even when the answer is 'obvious'

1. INconsistent set of sentences:

Where Partial Truth Tables Falter

So far, our only rigorous method for answering some questions *requires* a complete truth table, even when the answer is 'obvious'

1. INconsistent set of sentences:
    - Need to show that NO TVA makes every sentence true

Where Partial Truth Tables Falter

So far, our only rigorous method for answering some questions *requires* a complete truth table, even when the answer is ‘obvious’

1. INconsistent set of sentences:
    - Need to show that NO TVA makes every sentence true
2. Valid argument:

Where Partial Truth Tables Falter

So far, our only rigorous method for answering some questions *requires* a complete truth table, even when the answer is 'obvious'

1. INconsistent set of sentences:
    - Need to show that NO TVA makes every sentence true
2. Valid argument:
    - Need to show there is NO TVA where the premises are true but the conclusion is false

Where Partial Truth Tables Falter

So far, our only rigorous method for answering some questions *requires* a complete truth table, even when the answer is ‘obvious’

1. INconsistent set of sentences:
    - Need to show that NO TVA makes every sentence true
2. Valid argument:
    - Need to show there is NO TVA where the premises are true but the conclusion is false
3. Logically equivalent sentences:

Where Partial Truth Tables Falter

So far, our only rigorous method for answering some questions *requires* a complete truth table, even when the answer is 'obvious'

1. INconsistent set of sentences:
    - Need to show that NO TVA makes every sentence true
2. Valid argument:
    - Need to show there is NO TVA where the premises are true but the conclusion is false
3. Logically equivalent sentences:
    - Need to show that the sentences agree on EVERY TVA

Motivation for Trees (System STD)

- Trees formalize our pattern of thinking when making PTTs

Motivation for Trees (System STD)

- Trees formalize our pattern of thinking when making PTTs
  - For proofs of consistency, invalidity, or inequivalence, there is really little difference

Motivation for Trees (System STD)

- Trees formalize our pattern of thinking when making PTTs
  - For proofs of consistency, invalidity, or inequivalence, there is really little difference
  - Trees basically 'keep track of our mental work'

Motivation for Trees (System STD)

- Trees formalize our pattern of thinking when making PTTs
  - For proofs of consistency, invalidity, or inequivalence, there is really little difference
  - Trees basically 'keep track of our mental work'

- But for validity, inconsistency, and logical equivalence, we really get something new:

Motivation for Trees (System STD)

- Trees formalize our pattern of thinking when making PTTs
  - For proofs of consistency, invalidity, or inequivalence, there is really little difference
  - Trees basically 'keep track of our mental work'

- But for validity, inconsistency, and logical equivalence, we really get something new:
- Trees formalize our 'shortcut' arguments without needing to consider every TVA to the relevant atomic sentences

Motivation for Trees (System STD)

- Trees formalize our pattern of thinking when making PTTs
  - For proofs of consistency, invalidity, or inequivalence, there is really little difference
  - Trees basically 'keep track of our mental work'

- But for validity, inconsistency, and logical equivalence, we really get something new:
- Trees formalize our 'shortcut' arguments without needing to consider every TVA to the relevant atomic sentences
- (Although to be fair, the rigor of this 'shortcut' is beholden to our soundness result. But that's work you do once and then have FOREVER—much like a diploma!)

Similarities with Truth Tables

- Like truth tables, trees are *mechanical*:

Similarities with Truth Tables

- Like truth tables, trees are *mechanical*:
  - No insight or creativity required (wooooooooo!)

Similarities with Truth Tables

- Like truth tables, trees are *mechanical*:
  - No insight or creativity required (woooooooo!)
  - Just execute the resolution 'algorithm'

Similarities with Truth Tables

- Like truth tables, trees are *mechanical*:
  - No insight or creativity required (woooooooo!)
  - Just execute the resolution ‘algorithm’
- Trees give equivalent answers to truth tables

Similarities with Truth Tables

- Like truth tables, trees are *mechanical*:
  - No insight or creativity required (woooooooo!)
  - Just execute the resolution 'algorithm'
- Trees give equivalent answers to truth tables
  - Will prove this soon (soundness and completeness of STD)

What we will do (soon!) to Demonstrate 'Rigor'

- By proving that our tree system is *sound*, we show that these shortcut arguments are rigorous (they never lead us astray)

What we will do (soon!) to Demonstrate 'Rigor'

- By proving that our tree system is *sound*, we show that these shortcut arguments are rigorous (they never lead us astray)
  - **Sound**: Single turnstile entails Double Turnstile

What we will do (soon!) to Demonstrate ‘Rigor’

- By proving that our tree system is *sound*, we show that these shortcut arguments are rigorous (they never lead us astray)
  - **Sound**: Single turnstile entails Double Turnstile
  - (syntactic to semantic: i.e. we chose ‘good’ rules!)

What we will do (soon!) to Demonstrate ‘Rigor’

- By proving that our tree system is *sound*, we show that these shortcut arguments are rigorous (they never lead us astray)
  - **Sound**: Single turnstile entails Double Turnstile
  - (syntactic to semantic: i.e. we chose ‘good’ rules!)

- By proving that our tree system is *complete*, we will show that we never need truth tables: trees suffice

What we will do (soon!) to Demonstrate ‘Rigor’

- By proving that our tree system is *sound*, we show that these shortcut arguments are rigorous (they never lead us astray)
  - **Sound**: Single turnstile entails Double Turnstile
  - (syntactic to semantic: i.e. we chose ‘good’ rules!)
- By proving that our tree system is *complete*, we will show that we never need truth tables: trees suffice
  - **Complete**: Double Turnstile entails Single turnstile

What we will do (soon!) to Demonstrate 'Rigor'

- By proving that our tree system is *sound*, we show that these shortcut arguments are rigorous (they never lead us astray)
  - **Sound**: Single turnstile entails Double Turnstile
  - (syntactic to semantic: i.e. we chose 'good' rules!)

- By proving that our tree system is *complete*, we will show that we never need truth tables: trees suffice
  - **Complete**: Double Turnstile entails Single turnstile
  - (semantic notions are fully covered by our syntactic rules)

What we will do (soon!) to Demonstrate ‘Rigor’

- By proving that our tree system is *sound*, we show that these shortcut arguments are rigorous (they never lead us astray)
  - **Sound**: Single turnstile entails Double Turnstile
  - (syntactic to semantic: i.e. we chose ‘good’ rules!)
- By proving that our tree system is *complete*, we will show that we never need truth tables: trees suffice
  - **Complete**: Double Turnstile entails Single turnstile
  - (semantic notions are fully covered by our syntactic rules)
  - (Means: we wrote down *enough* rules!)

4. Truth Trees

b. Tree Rules (trees rule!)

Sentential Tree Derivations (STD)

- The method of trees is our first derivation system

Sentential Tree Derivations (STD)

- The method of trees is our first derivation system
- To distinguish it from our later natural deduction (ND) systems, we will call this system 'Sentential Tree Derivations' (STD)!

Sentential Tree Derivations (STD)

- The method of trees is our first derivation system
- To distinguish it from our later natural deduction (ND) systems, we will call this system ‘Sentential Tree Derivations’ (STD)!
- As a proof system, it comes with its very own single turnstile:

$$\vdash_{STD}$$

Sentential Tree Derivations (STD)

- The method of trees is our first derivation system
- To distinguish it from our later natural deduction (ND) systems, we will call this system 'Sentential Tree Derivations' (STD)!
- As a proof system, it comes with its very own single turnstile:

$$\vdash_{STD}$$

- As a proof system, it is *purely syntactic*: it is defined entirely in terms of legal rules, with no explicit mention of truth or falsity

Whence these rules?

- For negation, we have a single rule: double negation elimination

Whence these rules?

- For negation, we have a single rule: double negation elimination
- For each binary connective, we have two rules:
  one for an unnegated sentence; one for a negated sentence

Whence these rules?

- For negation, we have a single rule: double negation elimination
- For each binary connective, we have two rules:
  one for an unnegated sentence; one for a negated sentence
- Where do these rules come from?

Whence these rules?

- For negation, we have a single rule: double negation elimination
- For each binary connective, we have two rules:
  one for an unnegated sentence; one for a negated sentence
- Where do these rules come from?
- They come from the truth conditions for the connectives

Whence these rules?

- For negation, we have a single rule: double negation elimination
- For each binary connective, we have two rules:
  one for an unnegated sentence; one for a negated sentence
- Where do these rules come from?
- They come from the truth conditions for the connectives
- If you think about what it means for a formula to be true, you can always derive the rules

Stacking vs. Splitting (aka ‘branching’)

- There are two basic kinds of rules,
  coming from conjunction and disjunction:

Stacking vs. Splitting (aka ‘branching’)

- There are two basic kinds of rules,
  coming from conjunction and disjunction:
- Stacking: resolve $A \& B$ into ‘A stacked on B’

Stacking vs. Splitting (aka ‘branching’)

- There are two basic kinds of rules,
  coming from conjunction and disjunction:
- Stacking: resolve $A \& B$ into ‘A stacked on B’
  - Idea: for a conjunction to be true, both conjuncts must be true

Stacking vs. Splitting (aka 'branching')

- There are two basic kinds of rules,
  coming from conjunction and disjunction:
- Stacking: resolve $A \,\&\, B$ into 'A stacked on B'
  - Idea: for a conjunction to be true, both conjuncts must be true
- Splitting: resolve $A \vee B$ into an A−branch and a B−branch

Stacking vs. Splitting (aka ‘branching’)

- There are two basic kinds of rules,
  coming from conjunction and disjunction:
- Stacking: resolve $A \& B$ into ‘A stacked on B’
  - Idea: for a conjunction to be true, both conjuncts must be true
- Splitting: resolve $A \vee B$ into an A−branch and a B−branch
  - Idea: for a disjunction to be true, either disjunct must be true

Stacking vs. Splitting (aka ‘branching’)

- There are two basic kinds of rules,
  coming from conjunction and disjunction:
- Stacking: resolve $A \& B$ into ‘A stacked on B’
  - Idea: for a conjunction to be true, both conjuncts must be true
- Splitting: resolve $A \vee B$ into an A–branch and a B–branch
  - Idea: for a disjunction to be true, either disjunct must be true
- Atomic formulae and their negations can’t be further resolved

Your Very First Tree

- Use a tree to determine whether the following sentences are consistent (i.e. jointly satisfiable):

$$\sim A \,\&\, \sim D;\ C \,\&\, (B \vee A);\ \sim B \vee C;\ \sim D \,\&\, \sim F$$

Your Very First Tree

- Use a tree to determine whether the following sentences are consistent (i.e. jointly satisfiable):

$$\sim A \,\&\, \sim D;\ C \,\&\, (B \vee A);\ \sim B \vee C;\ \sim D \,\&\, \sim F$$

- Some things to NEVER forget!

- Use a tree to determine whether the following sentences are consistent (i.e. jointly satisfiable):

$$\sim A \,\&\, \sim D;\; C \,\&\, (B \vee A);\; \sim B \vee C;\; \sim D \,\&\, \sim F$$

- Some things to NEVER forget!
  - Each formula gets its own line, replete with line NUMBER

Your Very First Tree

- Use a tree to determine whether the following sentences are consistent (i.e. jointly satisfiable):

$$\sim A \,\&\, \sim D;\ C \,\&\, (B \vee A);\ \sim B \vee C;\ \sim D \,\&\, \sim F$$

- Some things to NEVER forget!
  - Each formula gets its own line, replete with line NUMBER
  - Justify new 'nodes' by citing the line number of a formula you are 'resolving'; note the rule you applied in the far right 'column'

Your Very First Tree

- Use a tree to determine whether the following sentences are consistent (i.e. jointly satisfiable):

$$\sim A \mathbin{\&} \sim D;\ C \mathbin{\&} (B \vee A);\ \sim B \vee C;\ \sim D \mathbin{\&} \sim F$$

- Some things to NEVER forget!
  - Each formula gets its own line, replete with line NUMBER
  - Justify new ‘nodes’ by citing the line number of a formula you are ‘resolving’; note the rule you applied in the far right ‘column’
  - Put a check ‘✓’ next to a formula once you resolve it

Your Very First Tree

- Use a tree to determine whether the following sentences are consistent (i.e. jointly satisfiable):
$$\sim A \,\&\, \sim D;\; C \,\&\, (B \vee A);\; \sim B \vee C;\; \sim D \,\&\, \sim F$$
- Some things to NEVER forget!
  - Each formula gets its own line, replete with line NUMBER
  - Justify new 'nodes' by citing the line number of a formula you are 'resolving'; note the rule you applied in the far right 'column'
  - Put a check '✓' next to a formula once you resolve it
  - (Perhaps put a colon ':' before each justification, in order to get used to what *Carnap* requires for natural deduction, e.g. :3 &)

The Rules in Themselves

Let's go through the rules!

1. Double negation (elimination)

The Rules in Themselves

Let's go through the rules!

1. Double negation (elimination)
2. Conjunction and Negated conjunction

The Rules in Themselves

Let's go through the rules!

1. Double negation (elimination)
2. Conjunction and Negated conjunction
3. Disjunction and Negated disjunction

The Rules in Themselves

Let's go through the rules!

1. Double negation (elimination)
2. Conjunction and Negated conjunction
3. Disjunction and Negated disjunction
4. Conditional and Negated conditional

The Rules in Themselves

Let's go through the rules!

1. Double negation (elimination)
2. Conjunction and Negated conjunction
3. Disjunction and Negated disjunction
4. Conditional and Negated conditional
5. Biconditional and Negated biconditional

The Rules in Themselves

Let's go through the rules!

1. Double negation (elimination)
2. Conjunction and Negated conjunction
3. Disjunction and Negated disjunction
4. Conditional and Negated conditional
5. Biconditional and Negated biconditional
   - Motivation: $A \,\&\, B$ branch or $\sim A \,\&\, \sim B$ branch

Double Negation ($\sim$)

$$
\begin{array}{lll}
\text{m.} & \sim\sim\Phi & \\
 & \vdots & \\
\vdots & \Big| & \\
\text{j.} & \Phi & \text{m} \sim
\end{array}
$$

Conjunction and Negated Conjunction

Conjunction (&)

m.	$\Phi \,\&\, \Psi$	
⋮	⋮	
j.	Φ	m &
j+1.	Ψ	

Negated Conjunction (∼ &)

m.	$\sim(\Phi \,\&\, \Psi)$		
⋮	⋮		
j.	$\sim\Phi$	$\sim\Psi$	m ∼ &

Disjunction and Negated Disjunction

Disjunction ($\vee$)

m.	$\Phi \vee \Psi$	
$\vdots$	$\vdots$	
j.	$\Phi \quad \Psi$	m $\vee$

Negated Disjunction ($\sim\vee$)

m.	$\sim(\Phi \vee \Psi)$	
$\vdots$	$\vdots$	
j.	$\sim\Phi$	m $\sim\vee$
j+1.	$\sim\Psi$	

Conditional and Negated Conditional

Conditional ($\supset$)

m.	$\Phi \supset \Psi$		
$\vdots$	$\vdots$		
j.	$\sim\Phi$	Ψ	m $\supset$

Negated Conditional ($\sim\supset$)

m.	$\sim(\Phi \supset \Psi)$	
$\vdots$	$\vdots$	
j.	Φ	m $\sim\supset$
j+1.	$\sim\Psi$	

Biconditional and Negated Biconditional

Biconditional ($\equiv$)

m.	$\Phi \equiv \Psi$		
⋮	⋮		
j.	Φ	$\sim\Phi$	m $\equiv$
j+1.	Ψ	$\sim\Psi$	

Negated Biconditional ($\sim\equiv$)

m.	$\sim(\Phi \equiv \Psi)$		
⋮	⋮		
j.	Φ	$\sim\Phi$	m $\sim\equiv$
j+1.	$\sim\Psi$	Ψ	

4. Truth Trees

c. Grow your own Trees!

Planting Trees

The Root of the Tree

You begin a tree by writing a list of formula(s)

Planting Trees

The Root of the Tree

You begin a tree by writing a list of formula(s)

- Every formula sits on its own line number

Planting Trees

The Root of the Tree

You begin a tree by writing a list of formula(s)

- Every formula sits on its own line number
- This list is called the **root** of the tree

Planting Trees

The Root of the Tree

You begin a tree by writing a list of formula(s)

- Every formula sits on its own line number
- This list is called the **root** of the tree

(The list is finite and non–empty; contains at least one formula)

Planting Trees

The Root of the Tree

You begin a tree by writing a list of formula(s)

- Every formula sits on its own line number
- This list is called the **root** of the tree

(The list is finite and non–empty; contains at least one formula)

- Grow your tree by adding **nodes**

Planting Trees

The Root of the Tree

You begin a tree by writing a list of formula(s)

- Every formula sits on its own line number
- This list is called the **root** of the tree

(The list is finite and non–empty; contains at least one formula)

- Grow your tree by adding **nodes**
- We count the root as a ‘node’ as well

Planting Trees

The Root of the Tree

You begin a tree by writing a list of formula(s)

- Every formula sits on its own line number
- This list is called the **root** of the tree

(The list is finite and non–empty; contains at least one formula)

- Grow your tree by adding **nodes**
- We count the root as a 'node' as well
- Each non–root node contains *one or two* wffs (per the rules)

each new node is connected to the node above it

Planting Trees

The Root of the Tree

You begin a tree by writing a list of formula(s)

- Every formula sits on its own line number
- This list is called the **root** of the tree

(The list is finite and non–empty; contains at least one formula)

- Grow your tree by adding **nodes**
- We count the root as a 'node' as well
- Each non–root node contains *one or two* wffs (per the rules)

  each new node is connected to the node above it
- Introduce a new line number for every new (row of) sentence(s)

  (line numbers correspond to (rows of) sentence(s), NOT nodes)

Where do the non-root nodes come from?

- We **resolve** sentences!
  (i.e. break them down, one main connective or negated formula at a time—much like our New Year's Resolutions)

Where do the non-root nodes come from?

- We **resolve** sentences!
  (i.e. break them down, one main connective or negated formula at a time—much like our New Year's Resolutions)
- We start by resolving the sentences in the root

Where do the non-root nodes come from?

- We **resolve** sentences!
  (i.e. break them down, one main connective or negated formula at a time—much like our New Year's Resolutions)
- We start by resolving the sentences in the root
- We then resolve any new nodes created, and so on

Where do the non-root nodes come from?

- We **resolve** sentences!
  (i.e. break them down, one main connective or negated formula at a time—much like our New Year's Resolutions)
- We start by resolving the sentences in the root
- We then resolve any new nodes created, and so on
- Until we hit rock bottom! i.e. atomic formulae or their negations

Where do the non-root nodes come from?

- We **resolve** sentences!
  (i.e. break them down, one main connective or negated formula at a time—much like our New Year's Resolutions)
- We start by resolving the sentences in the root
- We then resolve any new nodes created, and so on
- Until we hit rock bottom! i.e. atomic formulae or their negations
- (Sometimes you can stop before resolving all sentences)

Closed vs. Open Branches

- A **branch** is a path taking you from the root through a series of nodes, each connected to the one above it

Closed vs. Open Branches

- A **branch** is a path taking you from the root through a series of nodes, each connected to the one above it
- A branch is **closed** if a wff and its negation appear in its nodes

Closed vs. Open Branches

- A **branch** is a path taking you from the root through a series of nodes, each connected to the one above it
- A branch is **closed** if a wff and its negation appear in its nodes
  - **We write a** '$\times$' beneath each closed branch

Closed vs. Open Branches

- A **branch** is a path taking you from the root through a series of nodes, each connected to the one above it
- A branch is **closed** if a wff and its negation appear in its nodes
  - **We write a** ‘$\times$’ beneath each closed branch
  - Closure results from ANY wff and its negation (not just atomic)

Closed vs. Open Branches

- A **branch** is a path taking you from the root through a series of nodes, each connected to the one above it
- A branch is **closed** if a wff and its negation appear in its nodes
  - **We write a** '$\times$' beneath each closed branch
  - Closure results from ANY wff and its negation (not just atomic)
- Otherwise, a branch is **open**

Closed vs. Open Branches

- A **branch** is a path taking you from the root through a series of nodes, each connected to the one above it
- A branch is **closed** if a wff and its negation appear in its nodes
  - **We write a** '$\times$' beneath each closed branch
  - Closure results from ANY wff and its negation (not just atomic)
- Otherwise, a branch is **open**
  - As you grow your tree, any branch that is not closed is open

Closed vs. Open Branches

- A **branch** is a path taking you from the root through a series of nodes, each connected to the one above it
- A branch is **closed** if a wff and its negation appear in its nodes
  - **We write a** '$\times$' beneath each closed branch
  - Closure results from ANY wff and its negation (not just atomic)
- Otherwise, a branch is **open**
  - As you grow your tree, any branch that is not closed is open
  - We are particularly interested in branches that *remain open* even after every formula has been resolved

Closed vs. Open Branches

- A **branch** is a path taking you from the root through a series of nodes, each connected to the one above it

- A branch is **closed** if a wff and its negation appear in its nodes
  - **We write a** ‘$\times$’ beneath each closed branch
  - Closure results from ANY wff and its negation (not just atomic)

- Otherwise, a branch is **open**
  - As you grow your tree, any branch that is not closed is open
  - We are particularly interested in branches that *remain open* even after every formula has been resolved
  - These branches are **complete** and **open**

Completing a Branch

- A branch is **complete** when every resolvable formula appearing on it has been resolved (you've then 'completed' the branch)

Completing a Branch

- A branch is **complete** when every resolvable formula appearing on it has been resolved (you've then 'completed' the branch)
- i.e., we have applied the tree rules until nothing but atomic formulae or their negations appear (remember: inclusive 'or'!)

Completing a Branch

- A branch is **complete** when every resolvable formula appearing on it has been resolved (you've then 'completed' the branch)
- i.e., we have applied the tree rules until nothing but atomic formulae or their negations appear (remember: inclusive 'or'!)
- In a **complete open branch**, it is

Completing a Branch

- A branch is **complete** when every resolvable formula appearing on it has been resolved (you've then 'completed' the branch)
- i.e., we have applied the tree rules until nothing but atomic formulae or their negations appear (remember: inclusive 'or'!)
- In a **complete open branch**, it is
  - (i) not the case that both a sentence and its negation appear in its nodes (i.e. no contradictions along the branch)

Completing a Branch

- A branch is **complete** when every resolvable formula appearing on it has been resolved (you've then 'completed' the branch)
- i.e., we have applied the tree rules until nothing but atomic formulae or their negations appear (remember: inclusive 'or'!)
- In a **complete open branch**, it is
  - (i) not the case that both a sentence and its negation appear in its nodes (i.e. no contradictions along the branch)
  - and (ii) all formulae on the nodes have been resolved (so there's no possibility of a contradiction appearing later)

Completing a Branch

- A branch is **complete** when every resolvable formula appearing on it has been resolved (you've then 'completed' the branch)
- i.e., we have applied the tree rules until nothing but atomic formulae or their negations appear (remember: inclusive 'or'!)
- In a **complete open branch**, it is
  - (i) not the case that both a sentence and its negation appear in its nodes (i.e. no contradictions along the branch)
  - and (ii) all formulae on the nodes have been resolved (so there's no possibility of a contradiction appearing later)
  - **We write an** '$\uparrow$' beneath each complete open branch

Completing a Branch

- A branch is **complete** when every resolvable formula appearing on it has been resolved (you've then 'completed' the branch)
- i.e., we have applied the tree rules until nothing but atomic formulae or their negations appear (remember: inclusive 'or'!)
- In a **complete open branch**, it is
  - (i) not the case that both a sentence and its negation appear in its nodes (i.e. no contradictions along the branch)
  - and (ii) all formulae on the nodes have been resolved (so there's no possibility of a contradiction appearing later)
  - **We write an** '↑' beneath each complete open branch
- *Psssst, semantic point!*: a complete open branch indicates a TVA that makes each of the sentences in the root true

Returning to our Earlier Examples

Let's use trees!

- e.g. are the following sentences consistent?

Returning to our Earlier Examples

Let's use trees!

- e.g. are the following sentences consistent?

- $P \,\&\, Q$; $\sim(P \supset \sim Q)$; $(S \,\&\, R) \vee \sim J$

Returning to our Earlier Examples

Let's use trees!

- e.g. are the following sentences consistent?
- $P \,\&\, Q$; $\sim(P \supset \sim Q)$; $(S \,\&\, R) \vee \sim J$
- What about $P \,\&\, Q$; $\sim(P \supset \sim Q)$; $(S \,\&\, R) \vee \sim J$; $\sim(S \vee R) \,\&\, J$

Returning to our Earlier Examples

Let's use trees!

- e.g. are the following sentences consistent?
- $P \& Q$; $\sim(P \supset \sim Q)$; $(S \& R) \vee \sim J$
- What about $P \& Q$; $\sim(P \supset \sim Q)$; $(S \& R) \vee \sim J$; $\sim(S \vee R) \& J$
- Question (for later): can we reinterpret this second result as a valid argument? What are the premises? What is the conclusion?

Warning: No Logical Equivalencies!

- Keep in mind that the rules of STD are entirely syntactic

Warning: No Logical Equivalencies!

- Keep in mind that the rules of STD are entirely syntactic
- You can use ONLY these nine rules and no others

Warning: No Logical Equivalencies!

- Keep in mind that the rules of STD are entirely syntactic
- You can use ONLY these nine rules and no others
- In particular, STD itself knows nothing about logical equivalence

Warning: No Logical Equivalencies!

- Keep in mind that the rules of STD are entirely syntactic
- You can use ONLY these nine rules and no others
- In particular, STD itself knows nothing about logical equivalence
- So you CANNOT replace a sentence willy-nilly with a logically equivalent one, unless this is sanctioned by one of our rules

Warning: No Logical Equivalencies!

- Keep in mind that the rules of STD are entirely syntactic
- You can use ONLY these nine rules and no others
- In particular, STD itself knows nothing about logical equivalence
- So you CANNOT replace a sentence willy-nilly with a logically equivalent one, unless this is sanctioned by one of our rules
- Likewise, you can close a branch only if some wff Φ and its negation $\sim\Phi$ appear in the branch

4. Truth Trees

d. Using Trees

Syntactic equivalents of our Semantic Notions

- Recall that we are interested in assessing various semantic properties of sentences of SL:

Syntactic equivalents of our Semantic Notions

- Recall that we are interested in assessing various semantic properties of sentences of SL:

    1.) Contradiction? Tautology? Logically Contingent?

Syntactic equivalents of our Semantic Notions

- Recall that we are interested in assessing various semantic properties of sentences of SL:

  1.) Contradiction? Tautology? Logically Contingent?
  2.) Equivalent? Inequivalent?

Syntactic equivalents of our Semantic Notions

- Recall that we are interested in assessing various semantic properties of sentences of SL:

  1.) Contradiction? Tautology? Logically Contingent?
  2.) Equivalent? Inequivalent?
  3.) Consistent? Inconsistent?

Syntactic equivalents of our Semantic Notions

- Recall that we are interested in assessing various semantic properties of sentences of SL:

  1.) Contradiction? Tautology? Logically Contingent?
  2.) Equivalent? Inequivalent?
  3.) Consistent? Inconsistent?
  4.) Valid argument/entailment? Invalid argument/not entailed?

Syntactic equivalents of our Semantic Notions

- Recall that we are interested in assessing various semantic properties of sentences of SL:

  1.) Contradiction? Tautology? Logically Contingent?
  2.) Equivalent? Inequivalent?
  3.) Consistent? Inconsistent?
  4.) Valid argument/entailment? Invalid argument/not entailed?

- For each semantic notion, there is a corresponding syntactic property of a tree
  - (Although one has to prove this correspondence exists)

Tree-contradictions and Tree-tautologies

Tree-contradiction

- A sentence Φ is a tree-contradiction if there is a tree that starts with Φ that has only closed branches

Tree-contradictions and Tree-tautologies

Tree-contradiction

- A sentence Φ is a tree-contradiction if there is a tree that starts with Φ that has only closed branches
- (definition only requires the existence of a single such tree)

Tree-contradictions and Tree-tautologies

Tree-contradiction

- A sentence Φ is a tree-contradiction if there is a tree that starts with Φ that has only closed branches
- (definition only requires the existence of a single such tree)
- (it says nothing about *all* trees starting with Φ)

Tree-contradictions and Tree-tautologies

Tree-contradiction

- A sentence Φ is a tree-contradiction if there is a tree that starts with Φ that has only closed branches
- (definition only requires the existence of a single such tree)
- (it says nothing about *all* trees starting with Φ)

Tree-tautology

- A sentence Ψ is a tree-tautology if there is a tree that starts with $\sim\Psi$ that has only closed branches

Tree-contradictions and Tree-tautologies

Tree-contradiction

- A sentence Φ is a tree-contradiction if there is a tree that starts with Φ that has only closed branches
- (definition only requires the existence of a single such tree)
- (it says nothing about *all* trees starting with Φ)

Tree-tautology

- A sentence Ψ is a tree-tautology if there is a tree that starts with $\sim\Psi$ that has only closed branches
- i.e., provided that $\sim\Psi$ is a tree-contradiction

Tree-contradictions and Tree-tautologies

Tree-contradiction

- A sentence Φ is a tree-contradiction if there is a tree that starts with Φ that has only closed branches
- (definition only requires the existence of a single such tree)
- (it says nothing about *all* trees starting with Φ)

Tree-tautology

- A sentence Ψ is a tree-tautology if there is a tree that starts with $\sim\Psi$ that has only closed branches
- i.e., provided that $\sim\Psi$ is a tree-contradiction
- In this case, we write '$\vdash_{STD} \Psi$'

Tree-contradictions and Tree-tautologies

Tree-contradiction

- A sentence Φ is a tree-contradiction if there is a tree that starts with Φ that has only closed branches
- (definition only requires the existence of a single such tree)
- (it says nothing about *all* trees starting with Φ)

Tree-tautology

- A sentence Ψ is a tree-tautology if there is a tree that starts with $\sim\Psi$ that has only closed branches
- i.e., provided that $\sim\Psi$ is a tree-contradiction
- In this case, we write '$\vdash_{STD} \Psi$'
- (again: this definition requires the existence of single such tree)

A limitation of these syntactic definitions

- Notice that the definitions of ‘tree-contradiction’ and ‘tree-tautology’ leave open the possibility that a sentence could be both a tree-contradiction and a tree-tautology

A limitation of these syntactic definitions

- Notice that the definitions of ‘tree-contradiction’ and ‘tree-tautology’ leave open the possibility that a sentence could be both a tree-contradiction and a tree-tautology
- This is because each definition requires the existence of only a single tree with a given syntactic property

A limitation of these syntactic definitions

- Notice that the definitions of ‘tree-contradiction’ and ‘tree-tautology’ leave open the possibility that a sentence could be both a tree-contradiction and a tree-tautology
- This is because each definition requires the existence of only a single tree with a given syntactic property
- Obviously, if system STD is any good, this won’t be possible! But we’ll need to prove this (perhaps on PS 5)!

A limitation of these syntactic definitions

- Notice that the definitions of ‘tree-contradiction’ and ‘tree-tautology’ leave open the possibility that a sentence could be both a tree-contradiction and a tree-tautology
- This is because each definition requires the existence of only a single tree with a given syntactic property
- Obviously, if system STD is any good, this won’t be possible! But we’ll need to prove this (perhaps on PS 5)!
- Taking for granted that system STD is ‘good’, any tree-contradiction is a contradiction, and any tree-tautology is a tautology

Tree-consistency

Question: when is a set Γ of wffs consistent (i.e. jointly satisfiable)?

- Construct a tree whose root is all sentences in Γ

Tree-consistency

Question: when is a set Γ of wffs consistent (i.e. jointly satisfiable)?

- Construct a tree whose root is all sentences in Γ
- Next, apply the tree-rules until either

Tree-consistency

Question: when is a set Γ of wffs consistent (i.e. jointly satisfiable)?

- Construct a tree whose root is all sentences in Γ
- Next, apply the tree-rules until either

1.) **Each branch closes**, in which case the argument is **tree-inconsistent**

Tree-consistency

Question: when is a set Γ of wffs consistent (i.e. jointly satisfiable)?

- Construct a tree whose root is all sentences in Γ
- Next, apply the tree-rules until either

1.) **Each branch closes**, in which case the argument is **tree-inconsistent**

2.) You have a complete open branch, in which case the argument is **tree-consistent**

Tree-consistency

Question: when is a set Γ of wffs consistent (i.e. jointly satisfiable)?

- Construct a tree whose root is all sentences in Γ
- Next, apply the tree-rules until either

  1.) **Each branch closes**, in which case the argument is **tree-inconsistent**

  2.) You have a complete open branch, in which case the argument is **tree-consistent**

- (Pssst semantic aside: the complete open branch indicates a truth value assignment that makes each sentence in Γ true)

Connections between consistency and validity

- Recall from long ago: an argument is **valid** if and only if the premise set and the negation of the conclusion is **inconsistent** (i.e. if the premises are true, the conclusion is not false)

Connections between consistency and validity

- Recall from long ago: an argument is **valid** if and only if the premise set and the negation of the conclusion is **inconsistent** (i.e. if the premises are true, the conclusion is not false)

- An argument is invalid if and only if the premise set and the negation of the conclusion is consistent (i.e. the premises and the negated–conclusion are satisfiable)

Connections between consistency and validity

- Recall from long ago: an argument is **valid** if and only if the premise set and the negation of the conclusion is **inconsistent** (i.e. if the premises are true, the conclusion is not false)
- An argument is invalid if and only if the premise set and the negation of the conclusion is consistent (i.e. the premises and the negated-conclusion are satisfiable)
- These connections motivate our definitions of tree-validity and tree-invalidity

Tree-valid vs. Tree-invalid

- Consider an argument with premises given by a set Γ of wffs and conclusion Φ (i.e. Γ could be multiple sentences)

Tree-valid vs. Tree-invalid

- Consider an argument with premises given by a set Γ of wffs and conclusion Φ (i.e. Γ could be multiple sentences)
- Construct a tree with the following root: all sentences in Γ along with $\sim\Phi$ (i.e. the NEGATION of the conclusion)

Tree-valid vs. Tree-invalid

- Consider an argument with premises given by a set Γ of wffs and conclusion Φ (i.e. Γ could be multiple sentences)
- Construct a tree with the following root: all sentences in Γ along with $\sim\Phi$ (i.e. the NEGATION of the conclusion)
- Next, apply the tree-rules until either

Tree-valid vs. Tree-invalid

- Consider an argument with premises given by a set Γ of wffs and conclusion Φ (i.e. Γ could be multiple sentences)
- Construct a tree with the following root: all sentences in Γ along with $\sim\Phi$ (i.e. the NEGATION of the conclusion)
- Next, apply the tree-rules until either

1.) **Each branch closes**, in which case the argument is **tree-valid**

- In this case, we write $\Gamma \vdash_{STD} \Phi$

Tree-valid vs. Tree-invalid

- Consider an argument with premises given by a set Γ of wffs and conclusion Φ (i.e. Γ could be multiple sentences)
- Construct a tree with the following root: all sentences in Γ along with $\sim\Phi$ (i.e. the NEGATION of the conclusion)
- Next, apply the tree-rules until either

  1.) **Each branch closes**, in which case the argument is **tree-valid**

  - In this case, we write $\Gamma \vdash_{STD} \Phi$

  2.) You have a complete open branch, in which case the argument is **tree-invalid**

  - In this case, we write $\Gamma \nvdash_{STD} \Phi$ (at least, we can write this once we've shown STD is sound)

Using trees to check for Validity

Since most homework problems follow this pattern, let's make it really explicit!

1. Add each premise to the root (number each line)

Don't forget to **justify each new node** by citing the line you are resolving and the rule you are applying

Using trees to check for Validity

Since most homework problems follow this pattern, let's make it really explicit!

1. Add each premise to the root (number each line)
2. Add the **NEGATION** of the conclusion to the root

Don't forget to **justify each new node** by citing the line you are resolving and the rule you are applying

Using trees to check for Validity

Since most homework problems follow this pattern, let's make it really explicit!

1. Add each premise to the root (number each line)
2. Add the **NEGATION** of the conclusion to the root
3. Resolve sentences until either:

Don't forget to **justify each new node** by citing the line you are resolving and the rule you are applying

Using trees to check for Validity

Since most homework problems follow this pattern, let's make it really explicit!

1. Add each premise to the root (number each line)
2. Add the **NEGATION** of the conclusion to the root
3. Resolve sentences until either:
    - **Each branch closes**, in which case the argument is **valid**

Don't forget to **justify each new node** by citing the line you are resolving and the rule you are applying

Using trees to check for Validity

Since most homework problems follow this pattern, let's make it really explicit!

1. Add each premise to the root (number each line)
2. Add the **NEGATION** of the conclusion to the root
3. Resolve sentences until either:
    - **Each branch closes**, in which case the argument is **valid**
    - You have a complete open branch $\Rightarrow$ the argument is **invalid**

Don't forget to **justify each new node** by citing the line you are resolving and the rule you are applying

Using trees to check for Validity

Since most homework problems follow this pattern, let's make it really explicit!

1. Add each premise to the root (number each line)
2. Add the **NEGATION** of the conclusion to the root
3. Resolve sentences until either:
    - **Each branch closes**, in which case the argument is **valid**
    - You have a complete open branch $\Rightarrow$ the argument is **invalid**

Don't forget to **justify each new node** by citing the line you are resolving and the rule you are applying

Using trees to check for Validity

Since most homework problems follow this pattern, let's make it really explicit!

1. Add each premise to the root (number each line)
2. Add the **NEGATION** of the conclusion to the root
3. Resolve sentences until either:
    - **Each branch closes**, in which case the argument is **valid**
    - You have a complete open branch $\Rightarrow$ the argument is **invalid**

Don't forget to **justify each new node** by citing the line you are resolving and the rule you are applying

Remember that a branch closes whenever a sentence and its negation appear in its nodes (these need not be atomic sentences)

Using trees to check Tautologies

Likewise for whether a sentence is a tautology:

1. Add the **NEGATION** of the sentence to the root

Using trees to check Tautologies

Likewise for whether a sentence is a tautology:

1. Add the **NEGATION** of the sentence to the root
2. Resolve sentences until either:

Using trees to check Tautologies

Likewise for whether a sentence is a tautology:

1. Add the **NEGATION** of the sentence to the root
2. Resolve sentences until either:
    - **Each branch closes**, in which case the sentence is **tautologous** (semantic aside: it's impossible to make the sentence false)

Using trees to check Tautologies

Likewise for whether a sentence is a tautology:

1. Add the **NEGATION** of the sentence to the root
2. Resolve sentences until either:
   - **Each branch closes**, in which case the sentence is **tautologous** (semantic aside: it's impossible to make the sentence false)
   - You have a complete open branch, in which case the sentence is NOT a tautology
     (semantic aside: it is possible to satisfy the sentence's negation, so it's possible to make the sentence in question false)

Never forget to justify! Always remember!

- Just to repeat something you are liable to forget to do:

Never forget to justify! Always remember!

- Just to repeat something you are liable to forget to do:
- Never forget to **justify each new node** by

Never forget to justify! Always remember!

- Just to repeat something you are liable to forget to do:

- Never forget to **justify each new node** by
  1.) citing the line you are resolving (e.g. '3') and

Never forget to justify! Always remember!

- Just to repeat something you are liable to forget to do:

- Never forget to **justify each new node** by

  1.) citing the line you are resolving (e.g. '3') and

  2.) citing the rule you are applying (e.g. '$\sim\vee$')

Never forget to justify! Always remember!

- Just to repeat something you are liable to forget to do:

- Never forget to **justify each new node** by

  1.) citing the line you are resolving (e.g. ‘3’) and

  2.) citing the rule you are applying (e.g. ‘$\sim\vee$’)

  These justifications go in the ‘rightmost column’

Another Eternal Memory!

- If your tree has branched and you are resolving a wff, you have to put the result under EVERY branch connected to the wff

Another Eternal Memory!

- If your tree has branched and you are resolving a wff, you have to put the result under EVERY branch connected to the wff
- More precisely: To resolve a sentence, you have to extend **ALL** of the open branches *running through the node* of the given wff

Another Eternal Memory!

- If your tree has branched and you are resolving a wff, you have to put the result under EVERY branch connected to the wff
- More precisely: To resolve a sentence, you have to extend **ALL** of the open branches *running through the node* of the given wff
- Example: consider a root with $A \vee B$ and $P \,\&\, Q$. Check what happens when you resolve '$A \vee B$' first, followed by '$P \,\&\, Q$'

Another Eternal Memory!

- If your tree has branched and you are resolving a wff, you have to put the result under EVERY branch connected to the wff
- More precisely: To resolve a sentence, you have to extend **ALL** of the open branches *running through the node* of the given wff
- Example: consider a root with $A \vee B$ and $P \& Q$. Check what happens when you resolve '$A \vee B$' first, followed by '$P \& Q$'
- If a branch is already closed, you don't have to worry about it

4. Truth Trees

e. Topical Topiary Tips

How to Sculpt a Tree

- With trees, as with life, you’ve got options!

How to Sculpt a Tree

- With trees, as with life, you’ve got options!
- You can resolve sentences in any order you please

How to Sculpt a Tree

- With trees, as with life, you've got options!
- You can resolve sentences in any order you please
- But some resolution orders will be faster/easier/more convenient than others (they'll at least involve 'less ink', and someone is paying for that ink!)

How to Sculpt a Tree

- With trees, as with life, you’ve got options!
- You can resolve sentences in any order you please
- But some resolution orders will be faster/easier/more convenient than others (they’ll at least involve ‘less ink’, and someone is paying for that ink!)
- Corporate America and BigPharma want you to SAVE INK!

Rules of thumb for Green thumbs

1. Temporally favor ‘stacking’ rules over ‘splitting’ rules

Rules of thumb for Green thumbs

1. Temporally favor ‘stacking’ rules over ‘splitting’ rules

2. Given a choice, resolve sentences that do not lead to new open branches:

Rules of thumb for Green thumbs

1. Temporally favor ‘stacking’ rules over ‘splitting’ rules
2. Given a choice, resolve sentences that do not lead to new open branches:
    - Save splitting until you’ve closed as many branches as possible

Rules of thumb for Green thumbs

1. Temporally favor ‘stacking’ rules over ‘splitting’ rules
2. Given a choice, resolve sentences that do not lead to new open branches:
   - Save splitting until you’ve closed as many branches as possible
   - Otherwise, you can end up with a lot of branches!

Rules of thumb for Green thumbs

1. Temporally favor ‘stacking’ rules over ‘splitting’ rules
2. Given a choice, resolve sentences that do not lead to new open branches:
   - Save splitting until you’ve closed as many branches as possible
   - Otherwise, you can end up with a lot of branches!

     $\Rightarrow$ and that’s bad topiary!

An Example to Work Through

Let us illustrate these morals, since one burnt by the flame fears fire for life:

- Is the argument from $C \supset P$, $P \vee D$, $\sim(Q \equiv C)$ to D valid?

An Example to Work Through

Let us illustrate these morals, since one burnt by the flame fears fire for life:

- Is the argument from $C \supset P$, $P \vee D$, $\sim(Q \equiv C)$ to D valid?
- In the worst tree, resolve $C \supset P$, then $\sim(Q \equiv C)$, and then $P \vee D$

An Example to Work Through

Let us illustrate these morals, since one burnt by the flame fears fire for life:

- Is the argument from $C \supset P$, $P \vee D$, $\sim(Q \equiv C)$ to D valid?
- In the worst tree, resolve $C \supset P$, then $\sim(Q \equiv C)$, and then $P \vee D$
- In the best tree, resolve $P \vee D$, then $C \supset P$, and then $\sim(Q \equiv C)$

An Example to Work Through

Let us illustrate these morals, since one burnt by the flame fears fire for life:

- Is the argument from $C \supset P$, $P \vee D$, $\sim(Q \equiv C)$ to D valid?
- In the worst tree, resolve $C \supset P$, then $\sim(Q \equiv C)$, and then $P \vee D$
- In the best tree, resolve $P \vee D$, then $C \supset P$, and then $\sim(Q \equiv C)$
- Often we should take the road most traveled, and that will make all the difference

Our Running Example

Sarah lives in Chicago or Erie.
Amir lives in Chicago unless he enjoys hiking.
If Amir lives in Chicago, Sarah doesn't.
Neither Sarah nor Amir enjoy hiking.
$\therefore$ Sarah lives in Erie.

Our Running Example

Sarah lives in Chicago or Erie.
Amir lives in Chicago unless he enjoys hiking.
If Amir lives in Chicago, Sarah doesn't.
Neither Sarah nor Amir enjoy hiking.
∴ Sarah lives in Erie.

$$C \vee E$$
$$A \vee M$$
$$A \supset {\sim}C$$
$${\sim}S \,\&\, {\sim}M$$
$$\therefore E$$

Our Running Example

Sarah lives in Chicago or Erie.
Amir lives in Chicago unless he enjoys hiking.
If Amir lives in Chicago, Sarah doesn't.
Neither Sarah nor Amir enjoy hiking.
∴ Sarah lives in Erie.

$$C \vee E$$
$$A \vee M$$
$$A \supset \sim C$$
$$\sim S \,\&\, \sim M$$
$$\therefore E$$

Let us tree!

Our Running Example (simplified)

Recall that to handle this with a truth-table, we simplified the last premise (to eliminate 'S' and avoid a 32 row truth table):

Sarah lives in Chicago or Erie.

Amir lives in Chicago unless he enjoys hiking.

If Amir lives in Chicago, Sarah doesn't.

Amir doesn't enjoy hiking.

∴ *Sarah lives in Erie.*

$$C \vee E$$

$$A \vee M$$

$$A \supset \sim C$$

$$\sim M$$

$$\therefore E$$

A	C	E	M	$C \vee E$	$A \vee M$	$A \supset \sim C$	$\sim M$	E
T	T	T	T					
T	T	T	F					
T	T	F	T					
T	T	F	F					
T	F	T	T					
T	F	T	F					
T	F	F	T					
T	F	F	F					
F	T	T	T					
F	T	T	F					
F	T	F	T					
F	T	F	F					
F	F	T	T					
F	F	T	F					
F	F	F	T					
F	F	F	F					

A	C	E	M	$C \vee E$		$A \vee M$		$A \supset \sim C$		$\sim M$	E
T	T	T	T	T	T	T	T	T	T	T	T
T	T	T	F	T	T	T	F	T	T	F	T
T	T	F	T	T	F	T	T	T	T	T	F
T	T	F	F	T	F	T	F	T	T	F	F
T	F	T	T	F	T	T	T	T	F	T	T
T	F	T	F	F	T	T	F	T	F	F	T
T	F	F	T	F	F	T	T	T	F	T	F
T	F	F	F	F	F	T	F	T	F	F	F
F	T	T	T	T	T	F	T	F	T	T	T
F	T	T	F	T	T	F	F	F	T	F	T
F	T	F	T	T	F	F	T	F	T	T	F
F	T	F	F	T	F	F	F	F	T	F	F
F	F	T	T	F	T	F	T	F	F	T	T
F	F	T	F	F	T	F	F	F	F	F	T
F	F	F	T	F	F	F	T	F	F	T	F
F	F	F	F	F	F	F	F	F	F	F	F

A	C	E	M	C	$\vee$	E	A	$\vee$	M	A	$\supset$	$\sim$	C	$\sim$	M	E
T	T	T	T	T	T	T	T	T	T	T	F	F	T	F	T	T
T	T	T	F	T	T	T	T	T	F	T	F	F	T	T	F	T
T	T	F	T	T	T	F	T	T	T	T	F	F	T	F	T	F
T	T	F	F	T	T	F	T	T	F	T	F	F	T	T	F	F
T	F	T	T	F	T	T	T	T	T	T	T	T	F	F	T	T
T	F	T	F	F	T	T	T	T	F	T	T	T	F	T	F	T
T	F	F	T	F	F	F	T	T	T	T	T	T	F	F	T	F
T	F	F	F	F	F	F	T	T	F	T	T	T	F	T	F	F
F	T	T	T	T	T	T	F	T	T	F	T	F	T	F	T	T
F	T	T	F	T	T	T	F	F	F	F	T	F	T	T	F	T
F	T	F	T	T	T	F	F	T	T	F	T	F	T	F	T	F
F	T	F	F	T	T	F	F	F	F	F	T	F	T	T	F	F
F	F	T	T	F	T	T	F	T	T	F	T	T	F	F	T	T
F	F	T	F	F	T	T	F	F	F	F	T	T	F	T	F	T
F	F	F	T	F	F	F	F	T	T	F	T	T	F	F	T	F
F	F	F	F	F	F	F	F	F	F	F	T	T	F	T	F	F

4.e.6

A	C	E	M	C	$\vee$	E	A	$\vee$	M	A	$\supset$	$\sim$	C	$\sim$	M	E
T	T	T	T	T	T	T	T	T	T	T	F	F	T	F	T	T
T	T	T	F	T	T	T	T	T	F	T	F	F	T	T	F	T
T	T	F	T	T	T	F	T	T	T	T	F	F	T	F	T	F
T	T	F	F	T	T	F	T	T	F	T	F	F	T	T	F	F
T	F	T	T	F	T	T	T	T	T	T	T	T	F	F	T	T
T	F	T	F	F	T	T	T	T	F	T	T	T	F	T	F	T
T	F	F	T	F	F	F	T	T	T	T	T	T	F	F	T	F
T	F	F	F	F	F	F	T	T	F	T	T	T	F	T	F	F
F	T	T	T	T	T	T	F	T	T	F	T	F	T	F	T	T
F	T	T	F	T	T	T	F	F	F	F	T	F	T	T	F	T
F	T	F	T	T	T	F	F	T	T	F	T	F	T	F	T	F
F	T	F	F	T	T	F	F	F	F	F	T	F	T	T	F	F
F	F	T	T	F	T	T	F	T	T	F	T	T	F	F	T	T
F	F	T	F	F	T	T	F	F	F	F	T	T	F	T	F	T
F	F	F	T	F	F	F	F	T	T	F	T	T	F	F	T	F
F	F	F	F	F	F	F	F	F	F	F	T	T	F	T	F	F

A	C	E	M	$C \vee E$	$A \vee M$	$A \supset {\sim} C$	${\sim} M$	E
T	T	T	T	T T T	T T T	T F F T	F T	T
T	T	T	F	T T T	T T F	T F F T	T F	T
T	T	F	T	T T F	T T T	T F F T	F T	F
T	T	F	F	T T F	T T F	T F F T	T F	F
T	F	T	T	F T T	T T T	T T T F	F T	T
T	F	T	F	F T T	T T F	T T T F	T F	T
T	F	F	T	F F F	T T T	T T T F	F T	F
T	F	F	F	F F F	T T F	T T T F	T F	F
F	T	T	T	T T T	F T T	F T F T	F T	T
F	T	T	F	T T T	F F F	F T F T	T F	T
F	T	F	T	T T F	F T T	F T F T	F T	F
F	T	F	F	T T F	F F F	F T F T	T F	F
F	F	T	T	F T T	F T T	F T T F	F T	T
F	F	T	F	F T T	F F F	F T T F	T F	T
F	F	F	T	F F F	F T T	F T T F	F T	F
F	F	F	F	F F F	F F F	F T T F	T F	F

Every valuation makes at least one premise false, or makes the conclusion true: the argument is valid.

4.e.6

A fun Question

- Question: how can you use a single tree to show that two sentences are logically equivalent?

A fun Question

- Question: how can you use a single tree to show that two sentences are logically equivalent?
- Answer is on next slide! No spoilers!

A fun Question

- Question: how can you use a single tree to show that two sentences are logically equivalent?
- Answer is on next slide! No spoilers!
- Example: show that $P \vee Q$ is logically equivalent to $\sim\Big((\sim(P \equiv Q)) \equiv (P \,\&\, Q)\Big)$

A fun Question

- Question: how can you use a single tree to show that two sentences are logically equivalent?
- Answer is on next slide! No spoilers!
- Example: show that $P \vee Q$ is logically equivalent to $\sim\Big((\sim(P \equiv Q)) \equiv (P \,\&\, Q)\Big)$
- (where this represents $(x + y) + (x \cdot y)$ in our Boolean algebra)

Answer to our ‘fun Question’

- Question: how can you use a single tree to show that two sentences are logically equivalent?

Answer to our ‘fun Question’

- Question: how can you use a single tree to show that two sentences are logically equivalent?
- Answer:

Answer to our ‘fun Question’

- Question: how can you use a single tree to show that two sentences are logically equivalent?
- Answer:
- Consider whether the biconditional of the two formulae is a tautology

Answer to our 'fun Question'

- Question: how can you use a single tree to show that two sentences are logically equivalent?
- Answer:
- Consider whether the biconditional of the two formulae is a tautology
- So make a tree whose root is the negation of this biconditional

Answer to our ‘fun Question’

- Question: how can you use a single tree to show that two sentences are logically equivalent?
- Answer:
- Consider whether the biconditional of the two formulae is a tautology
- So make a tree whose root is the negation of this biconditional
- If all branches close, then this negation is unsatisfiable, i.e. is a contradiction. In which case the biconditional is a tautology. In which case the two wffs are logically equivalent.

4. Truth Trees

f. Practice with Proofs

Two Questions about (semantic) Entailment

Two questions we never got around to answering:

> Recall: '$\Gamma \nvDash \psi$' means that the (set of) sentence(s) Γ does not semantically entail ψ, i.e. an argument from Γ to ψ is invalid.

Two Questions about (semantic) Entailment

Two questions we never got around to answering:

> Recall: '$\Gamma \nvDash \Psi$' means that the (set of) sentence(s) Γ does not semantically entail Ψ, i.e. an argument from Γ to Ψ is invalid.

1. True or False? If $\Phi \vDash \Psi$, then $\sim\Phi \nvDash \Psi$

Two Questions about (semantic) Entailment

Two questions we never got around to answering:

> Recall: '$\Gamma \nvDash \Psi$' means that the (set of) sentence(s) Γ does not semantically entail Ψ, i.e. an argument from Γ to Ψ is invalid.

1. True or False? If $\Phi \vDash \Psi$, then $\sim\Phi \nvDash \Psi$

2. True or False? If $\Gamma \vDash \Phi$ and $\Delta, \Phi \vDash \Psi$, then $\Gamma, \Delta \vDash \Psi$?

And now with trees!

Let's answer syntactic analogs of these questions in system STD:

Recall: '$\Gamma \nvdash_{STD} \Psi$' means that arguing from Γ to Ψ is NOT tree-valid (and with soundness, this means it is tree-invalid)

And now with trees!

Let's answer syntactic analogs of these questions in system STD:

> Recall: '$\Gamma \nvdash_{STD} \Psi$' means that arguing from Γ to Ψ is NOT tree-valid (and with soundness, this means it is tree-invalid)

1. True or False? If $\Phi \vdash_{STD} \Psi$, then $\sim\Phi \nvdash_{STD} \Psi$

And now with trees!

Let's answer syntactic analogs of these questions in system STD:

> Recall: '$\Gamma \nvdash_{STD} \Psi$' means that arguing from Γ to Ψ is NOT tree-valid (and with soundness, this means it is tree-invalid)

1. True or False? If $\Phi \vdash_{STD} \Psi$, then $\sim\Phi \nvdash_{STD} \Psi$

2. True or False? If $\Gamma \vdash_{STD} \Phi$, then $\Gamma, \Delta \vdash_{STD} \Phi$

And now with trees!

Let's answer syntactic analogs of these questions in system STD:

> Recall: '$\Gamma \nvdash_{STD} \Psi$' means that arguing from Γ to Ψ is NOT tree-valid (and with soundness, this means it is tree-invalid)

1. True or False? If $\Phi \vdash_{STD} \Psi$, then $\sim\Phi \nvdash_{STD} \Psi$
2. True or False? If $\Gamma \vdash_{STD} \Phi$, then $\Gamma, \Delta \vdash_{STD} \Phi$
3. True or False? If $\Gamma \vdash_{STD} \Phi$ and $\Delta, \Phi \vdash_{STD} \Psi$, then $\Gamma, \Delta \vdash_{STD} \Psi$?

An Induction example because...why not?

Gotta stay sharp!

Prove the following by induction. Don't forget to explicitly state the base case and the induction step!

3. If a wff doesn't contain any binary connectives, then it is contingent.
   (hint: say that a wff is *baller* if it either contains a binary connective or is contingent. Use induction to show that every wff is baller.)