5. Metalogic for STD

5. Metalogic for STD

a. Big Picture Stuff

Trees as a Shortcut, provided that...

- As we have seen, trees provide a shortcut for demonstrating that a set of sentences is inconsistent (i.e. unsatisfiable): construct a tree whose root is these sentences s.t. all branches close

Trees as a Shortcut, provided that...

- As we have seen, trees provide a shortcut for demonstrating that a set of sentences is inconsistent (i.e. unsatisfiable): construct a tree whose root is these sentences s.t. all branches close
- Underwrites further shortcuts for demonstrating:

Trees as a Shortcut, provided that...

- As we have seen, trees provide a shortcut for demonstrating that a set of sentences is inconsistent (i.e. unsatisfiable): construct a tree whose root is these sentences s.t. all branches close
- Underwrites further shortcuts for demonstrating:
  1.) that an argument is valid
  (its premises and negated conclusion are unsatisfiable)

Trees as a Shortcut, provided that...

- As we have seen, trees provide a shortcut for demonstrating that a set of sentences is inconsistent (i.e. unsatisfiable): construct a tree whose root is these sentences s.t. all branches close
- Underwrites further shortcuts for demonstrating:
  1.) that an argument is valid
  (its premises and negated conclusion are unsatisfiable)
  2.) that a sentence is a tautology (its negation is unsatisfiable)

Trees as a Shortcut, provided that...

- As we have seen, trees provide a shortcut for demonstrating that a set of sentences is inconsistent (i.e. unsatisfiable): construct a tree whose root is these sentences s.t. all branches close
- Underwrites further shortcuts for demonstrating:
  1.) that an argument is valid
  (its premises and negated conclusion are unsatisfiable)
  2.) that a sentence is a tautology (its negation is unsatisfiable)
  3.) that two sentences are logically equivalent

Trees as a Shortcut, provided that...

- As we have seen, trees provide a shortcut for demonstrating that a set of sentences is inconsistent (i.e. unsatisfiable): construct a tree whose root is these sentences s.t. all branches close
- Underwrites further shortcuts for demonstrating:
  1.) that an argument is valid
  (its premises and negated conclusion are unsatisfiable)
  2.) that a sentence is a tautology (its negation is unsatisfiable)
  3.) that two sentences are logically equivalent
- But our shortcuts are justified only if system STD is *sound*

Trees as a Shortcut, provided that...

- As we have seen, trees provide a shortcut for demonstrating that a set of sentences is inconsistent (i.e. unsatisfiable): construct a tree whose root is these sentences s.t. all branches close
- Underwrites further shortcuts for demonstrating:
  1.) that an argument is valid
     (its premises and negated conclusion are unsatisfiable)
  2.) that a sentence is a tautology (its negation is unsatisfiable)
  3.) that two sentences are logically equivalent
- But our shortcuts are justified only if system STD is *sound*
- And guaranteed to have a shortcut only if system STD is *complete*

A Tale of Two Turnstiles: the syntactic one

- Recall that the single turnstile '$\vdash_{STD}$' stands for provability (aka derivability) within our proof system STD

A Tale of Two Turnstiles: the syntactic one

- Recall that the single turnstile ‘$\vdash_{STD}$’ stands for provability (aka derivability) within our proof system STD
- “$\Gamma \vdash_{STD} \Theta$” means that we can derive Θ from Γ, within STD. The argument with premises Γ and conclusion Θ is ‘**tree-valid**’

A Tale of Two Turnstiles: the syntactic one

- Recall that the single turnstile '$\vdash_{STD}$' stands for provability (aka derivability) within our proof system STD
- "$\Gamma \vdash_{STD} \Theta$" means that we can derive Θ from Γ, within STD. The argument with premises Γ and conclusion Θ is '**tree-valid**'
- Equivalently, this means that $\Gamma \cup \{\sim\Theta\}$ is **tree-inconsistent**: There is a tree with this set as the root s.t. **all branches close**

A Tale of Two Turnstiles: the syntactic one

- Recall that the single turnstile '$\vdash_{STD}$' stands for provability (aka derivability) within our proof system STD
- "$\Gamma \vdash_{STD} \Theta$" means that we can derive Θ from Γ, within STD. The argument with premises Γ and conclusion Θ is '**tree-valid**'
- Equivalently, this means that $\Gamma \cup \{\sim\Theta\}$ is **tree-inconsistent**: There is a tree with this set as the root s.t. **all branches close**
- Throughout, 'Γ' is a *finite* set of SL sentences

A Tale of Two Turnstiles: the semantic one

- Recall that the double turnstile '$\vDash$' stands for semantic entailment (aka logical consequence) within (classical) sentential logic SL.

A Tale of Two Turnstiles: the semantic one

- Recall that the double turnstile '$\vDash$' stands for semantic entailment (aka logical consequence) within (classical) sentential logic SL.
- "$\Gamma \vDash \Theta$" means that Γ logically entails Θ
  Whenever the premises in Γ are true, the conclusion Θ is true

A Tale of Two Turnstiles: the semantic one

- Recall that the double turnstile '$\vDash$' stands for semantic entailment (aka logical consequence) within (classical) sentential logic SL.
- "$\Gamma \vDash \Theta$" means that Γ logically entails Θ
  Whenever the premises in Γ are true, the conclusion Θ is true
- Equivalently: there is no truth–value assignment (TVA) s.t.
  Γ is satisfied while Θ is false

A Tale of Two Turnstiles: the semantic one

- Recall that the double turnstile ‘$\vDash$’ stands for semantic entailment (aka logical consequence) within (classical) sentential logic SL.
- “$\Gamma \vDash \Theta$” means that Γ logically entails Θ
  Whenever the premises in Γ are true, the conclusion Θ is true
- Equivalently: there is no truth-value assignment (TVA) s.t. Γ is satisfied while Θ is false
- Equivalently, this means that $\Gamma \cup \{\sim\Theta\}$ is logically inconsistent: no TVA satisfies the premises and negated conclusion

Promises Made, Promises Kept

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

Promises Made, Promises Kept

- By proving that our tree system is *sound*, we show that these shortcut arguments are rigorous (they never lead us astray)
  - **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$

Promises Made, Promises Kept

- By proving that our tree system is *sound*, we show that these shortcut arguments are rigorous (they never lead us astray)
  - **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
  - (syntactic to semantic: i.e. we chose 'good' rules!)

Promises Made, Promises Kept

- By proving that our tree system is *sound*, we show that these shortcut arguments are rigorous (they never lead us astray)
  - **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
  - (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

Promises Made, Promises Kept

- By proving that our tree system is *sound*, we show that these shortcut arguments are rigorous (they never lead us astray)
  - **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
  - (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**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD} \Theta$

Promises Made, Promises Kept

- By proving that our tree system is *sound*, we show that these shortcut arguments are rigorous (they never lead us astray)
  - **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
  - (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**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD} \Theta$
  - (semantic notions are fully covered by our syntactic rules)

Promises Made, Promises Kept

- By proving that our tree system is *sound*, we show that these shortcut arguments are rigorous (they never lead us astray)
  - **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
  - (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**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD} \Theta$
  - (semantic notions are fully covered by our syntactic rules)
  - (Means: we wrote down *enough* rules!)

Basic Proof Strategy

- Notice that both soundness and completeness are if–then statements (i.e. of the form $P \supset Q$):

Basic Proof Strategy

- Notice that both soundness and completeness are if–then statements (i.e. of the form $P \supset Q$):
  - **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$

Basic Proof Strategy

- Notice that both soundness and completeness are if–then statements (i.e. of the form $P \supset Q$):
  - **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
  - **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD} \Theta$

Basic Proof Strategy

- Notice that both soundness and completeness are if–then statements (i.e. of the form $P \supset Q$):
  - **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
  - **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD} \Theta$

- Hence, they are logically equivalent to their contrapositives:

Basic Proof Strategy

- Notice that both soundness and completeness are if-then statements (i.e. of the form $P \supset Q$):
  - **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
  - **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD} \Theta$
- Hence, they are logically equivalent to their contrapositives:
  - Contrapositive of $(P \supset Q)$ is $(\sim Q \supset \sim P)$

Basic Proof Strategy

- Notice that both soundness and completeness are if–then statements (i.e. of the form $P \supset Q$):
  - **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
  - **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD} \Theta$

- Hence, they are logically equivalent to their contrapositives:
  - Contrapositive of $(P \supset Q)$ is $({\sim}Q \supset {\sim}P)$
  - **Soundness**: If $\Gamma \nvDash \Theta$, then $\Gamma \nvdash_{STD} \Theta$

Basic Proof Strategy

- Notice that both soundness and completeness are if–then statements (i.e. of the form $P \supset Q$):
  - **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
  - **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD} \Theta$

- Hence, they are logically equivalent to their contrapositives:
  - Contrapositive of $(P \supset Q)$ is $(\sim Q \supset \sim P)$
  - **Soundness**: If $\Gamma \nvDash \Theta$, then $\Gamma \nvdash_{STD} \Theta$
  - **Completeness**: If $\Gamma \nvdash_{STD} \Theta$, then $\Gamma \nvDash \Theta$

Basic Proof Strategy

- Notice that both soundness and completeness are if–then statements (i.e. of the form $P \supset Q$):
  - **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \models \Theta$
  - **Complete**: If $\Gamma \models \Theta$, then $\Gamma \vdash_{STD} \Theta$
- Hence, they are logically equivalent to their contrapositives:
  - Contrapositive of $(P \supset Q)$ is $(\sim Q \supset \sim P)$
  - **Soundness**: If $\Gamma \not\models \Theta$, then $\Gamma \nvdash_{STD} \Theta$
  - **Completeness**: If $\Gamma \nvdash_{STD} \Theta$, then $\Gamma \not\models \Theta$
- We will prove the contrapositives, using induction! Wooooo!

5. Metalogic for STD

b. Soundness of System STD

Sounding out Soundness

- **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$

Sounding out Soundness

- **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
- *$Sound_{contra}$*: If $\Gamma \nvDash \Theta$, then $\Gamma \nvdash_{STD} \Theta$

Sounding out Soundness

- **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
- *$Sound_{contra}$*: If $\Gamma \nvDash \Theta$, then $\Gamma \nvdash_{STD} \Theta$
- As always: ask what this means, based on the definitions:

Sounding out Soundness

- **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
- *$Sound_{contra}$*: If $\Gamma \nvDash \Theta$, then $\Gamma \nvdash_{STD} \Theta$
- As always: ask what this means, based on the definitions:
- "$\Gamma \nvDash \Theta$" means that *it is not the case that* the set Γ entails Θ, i.e. there is a TVA where Γ is true but Θ is false.

Sounding out Soundness

- **Sound**: If $\Gamma \vdash_{STD} \Theta$, then $\Gamma \vDash \Theta$
- *Sound$_{contra}$*: If $\Gamma \nvDash \Theta$, then $\Gamma \nvdash_{STD} \Theta$
- As always: ask what this means, based on the definitions:
- "$\Gamma \nvDash \Theta$" means that *it is not the case that* the set Γ entails Θ, i.e. there is a TVA where Γ is true but Θ is false.
  - So on this TVA, $\sim\Theta$ is true $\Rightarrow \Gamma \cup \{\sim\Theta\}$ is CONSISTENT

Analyzing Key Definitions Continued

- *Sound$_{contra}$*: If $\Gamma \nvDash \Theta$, then $\Gamma \nvdash_{STD} \Theta$

Analyzing Key Definitions Continued

- *Sound$_{contra}$*: If $\Gamma \not\models \Theta$, then $\Gamma \nvdash_{STD} \Theta$
- "$\Gamma \nvdash_{STD} \Theta$" means that *it is not the case that* the argument from Γ to Θ is **tree-valid**

Analyzing Key Definitions Continued

- *Sound$_{contra}$*: If $\Gamma \nvDash \Theta$, then $\Gamma \nvdash_{STD} \Theta$
- "$\Gamma \nvdash_{STD} \Theta$" means that *it is not the case that* the argument from Γ to Θ is **tree-valid**
  - i.e., *it is not the case that* there exists a tree with root $\Gamma \cup \{\sim\Theta\}$ that possesses **all closed branches**

Analyzing Key Definitions Continued

- *Sound*$_{contra}$: If $\Gamma \not\models \Theta$, then $\Gamma \nvdash_{STD} \Theta$
- "$\Gamma \nvdash_{STD} \Theta$" means that *it is not the case that* the argument from Γ to Θ is **tree-valid**
  - i.e., *it is not the case that* there exists a tree with root $\Gamma \cup \{\sim\Theta\}$ that possesses **all closed branches**
  - Equivalently: ANY tree with root $\Gamma \cup \{\sim\Theta\}$ possesses **at least one complete open branch**

Analyzing Key Definitions Continued

- *Sound$_{contra}$*: If $\Gamma \not\vDash \Theta$, then $\Gamma \nvdash_{STD} \Theta$
- "$\Gamma \nvdash_{STD} \Theta$" means that *it is not the case that* the argument from Γ to Θ is **tree-valid**
    - i.e., *it is not the case that* there exists a tree with root $\Gamma \cup \{\sim\Theta\}$ that possesses **all closed branches**
    - Equivalently: ANY tree with root $\Gamma \cup \{\sim\Theta\}$ possesses **at least one complete open branch**
    - (Aside: this is NOT the same as saying that the argument is **tree-invalid**, since that only requires the existence of a single tree with a complete open branch)

Putting these Two Pieces Together

- Assume $\Gamma \nvDash \Theta$, i.e. assume that $\Gamma \cup \{\sim\Theta\}$ is CONSISTENT

Putting these Two Pieces Together

- Assume $\Gamma \nvDash \Theta$, i.e. assume that $\Gamma \cup \{\sim\Theta\}$ is CONSISTENT
  - Then, there is a TVA that makes the premises true and the conclusion false. Call this TVA '$\mathcal{I}$', which we'll use throughout

Putting these Two Pieces Together

- Assume $\Gamma \nvDash \Theta$, i.e. assume that $\Gamma \cup \{\sim\Theta\}$ is CONSISTENT
  - Then, there is a TVA that makes the premises true and the conclusion false. Call this TVA '$\mathcal{I}$', which we'll use throughout
- Need to Show: ANY tree with root $\Gamma \cup \{\sim\Theta\}$ possesses a complete open branch (i.e. the argument is NOT tree-valid)

Putting these Two Pieces Together

- Assume $\Gamma \nvDash \Theta$, i.e. assume that $\Gamma \cup \{\sim\Theta\}$ is CONSISTENT
  - Then, there is a TVA that makes the premises true and the conclusion false. Call this TVA '$\mathcal{I}$', which we'll use throughout
- Need to Show: ANY tree with root $\Gamma \cup \{\sim\Theta\}$ possesses a complete open branch (i.e. the argument is NOT tree-valid)
- Equivalently, we are showing that if the root is consistent, then any tree with that root possesses a complete open branch

Putting these Two Pieces Together

- Assume $\Gamma \nvDash \Theta$, i.e. assume that $\Gamma \cup \{\sim\Theta\}$ is CONSISTENT
  - Then, there is a TVA that makes the premises true and the conclusion false. Call this TVA '$\mathcal{I}$', which we'll use throughout
- Need to Show: ANY tree with root $\Gamma \cup \{\sim\Theta\}$ possesses a complete open branch (i.e. the argument is NOT tree-valid)
- Equivalently, we are showing that if the root is consistent, then any tree with that root possesses a complete open branch
- (i.e. whenever the root is satisfiable, the tree will not close)

Key Facts about our TVA $\mathcal{I}$

- By definition, $\mathcal{I}$ makes Θ false but everything in Γ true

Key Facts about our TVA $\mathcal{I}$

- By definition, $\mathcal{I}$ makes Θ false but everything in Γ true
- As a TVA, $\mathcal{I}$ assigns either 'true' or 'false' (exclusive-or) to every atomic sentence letter appearing in the sentences of Γ and Θ

Key Facts about our TVA $\mathcal{I}$

- By definition, $\mathcal{I}$ makes Θ false but everything in Γ true
- As a TVA, $\mathcal{I}$ assigns either 'true' or 'false' (exclusive-or) to every atomic sentence letter appearing in the sentences of Γ and Θ
- Hence, $\mathcal{I}$ determines a truth value for any wff built from these atomic sentences

Key Facts about our TVA $\mathcal{I}$

- By definition, $\mathcal{I}$ makes Θ false but everything in Γ true
- As a TVA, $\mathcal{I}$ assigns either 'true' or 'false' (exclusive-or) to every atomic sentence letter appearing in the sentences of Γ and Θ
- Hence, $\mathcal{I}$ determines a truth value for any wff built from these atomic sentences
- for any such wff Ψ, $\mathcal{I}$ makes Ψ true if and only if it makes $\sim\Psi$ false

Some Convenient Definitions to Streamline our Inductive Proof

- Call a tree **good** if its root contains only $\sim\Theta$ and the wff in Γ

Some Convenient Definitions to Streamline our Inductive Proof

- Call a tree **good** if its root contains only $\sim\Theta$ and the wff in Γ
- Say that a branch in a good tree is an **$\mathcal{I}$–branch** if every wff on it is $\mathcal{I}$–true, i.e. true according to $\mathcal{I}$

Some Convenient Definitions to Streamline our Inductive Proof

- Call a tree **good** if its root contains only $\sim\Theta$ and the wff in Γ
- Say that a branch in a good tree is an **$\mathcal{I}$–branch** if every wff on it is $\mathcal{I}$–true, i.e. true according to $\mathcal{I}$
- Note that each $\mathcal{I}$–branch is open, since $\mathcal{I}$ cannot make true both a wff and its negation (so an $\mathcal{I}$–branch can't be closed!)

Some Convenient Definitions to Streamline our Inductive Proof

- Call a tree **good** if its root contains only $\sim\Theta$ and the wff in Γ
- Say that a branch in a good tree is an **$\mathcal{I}$–branch** if every wff on it is $\mathcal{I}$–true, i.e. true according to $\mathcal{I}$
- Note that each $\mathcal{I}$–branch is open, since $\mathcal{I}$ cannot make true both a wff and its negation (so an $\mathcal{I}$–branch can't be closed!)
- **Proof plan**: show that every **good** tree contains an **$\mathcal{I}$–branch**

Some Convenient Definitions to Streamline our Inductive Proof

- Call a tree **good** if its root contains only $\sim\Theta$ and the wff in Γ
- Say that a branch in a good tree is an **$\mathcal{I}$–branch** if every wff on it is $\mathcal{I}$–true, i.e. true according to $\mathcal{I}$
- Note that each $\mathcal{I}$–branch is open, since $\mathcal{I}$ cannot make true both a wff and its negation (so an $\mathcal{I}$–branch can't be closed!)
- **Proof plan**: show that every **good** tree contains an **$\mathcal{I}$–branch**
- This will include all the completed good trees, which will have a *complete* open branch, which is what we want to show

Set-Up for Induction

- Trees have a recursive structure (they ‘grow’ by nine rules),
  $\Rightarrow$ we can perform proof by induction (woooooooooo)!

Set-Up for Induction

- Trees have a recursive structure (they ‘grow’ by nine rules),
  $\Rightarrow$ we can perform proof by induction (wooooooooo)!
- We’ll do complete induction over the *number of nodes*!
  (could also do ordinary induction on # of times we apply a rule)

Set-Up for Induction

- Trees have a recursive structure (they ‘grow’ by nine rules),
  $\Rightarrow$ we can perform proof by induction (woooooooooo)!
- We’ll do complete induction over the *number of nodes*!
  (could also do ordinary induction on # of times we apply a rule)
- **Base case**: the smallest good tree (1 node)

Set-Up for Induction

- Trees have a recursive structure (they ‘grow’ by nine rules),
  $\Rightarrow$ we can perform proof by induction (wooooooooo)!
- We’ll do complete induction over the *number of nodes*!
  (could also do ordinary induction on # of times we apply a rule)
- **Base case**: the smallest good tree (1 node)
- **Induction hypothesis**: Assume that every good tree with n–many nodes, where $1 \leq n < k$, contains an $\mathcal{I}$–branch

Set–Up for Induction

- Trees have a recursive structure (they 'grow' by nine rules), $\Rightarrow$ we can perform proof by induction (wooooooooo)!
- We'll do complete induction over the *number of nodes*! (could also do ordinary induction on # of times we apply a rule)
- **Base case**: the smallest good tree (1 node)
- **Induction hypothesis**: Assume that every good tree with n–many nodes, where $1 \leq n < k$, contains an $\mathcal{I}$–branch
- Induction Step: show that an arbitrary good tree with k nodes also contains an $\mathcal{I}$–branch (where $k > 1$)

Getting this Party Started: the Base Case

- **Base case**: consider a good tree with one node

Getting this Party Started: the Base Case

- **Base case**: consider a good tree with one node
- This is just the root $\Gamma \cup \{\sim\Theta\}$

Getting this Party Started: the Base Case

- **Base case**: consider a good tree with one node
- This is just the root $\Gamma \cup \{\sim\Theta\}$
- By definition, the TVA $\mathcal{I}$ satisfies the root

Getting this Party Started: the Base Case

- **Base case**: consider a good tree with one node
- This is just the root $\Gamma \cup \{\sim\Theta\}$
- By definition, the TVA $\mathcal{I}$ satisfies the root
- $\Rightarrow$ root is an $\mathcal{I}$–branch, i.e. every wff on it is true according to $\mathcal{I}$

Keeping the Party Going

- Consider an arbitrary good tree 'Theodore' with k nodes ($k > 1$)

Keeping the Party Going

- Consider an arbitrary good tree 'Theodore' with k nodes ($k > 1$)
- Then there must be a good tree 'Theo' with $1 \leq n < k$ nodes such that 'Theodore' results from applying one of our nine STD rules to 'Theo'. Call this rule 'Ruby' $\in \{\sim, \&, \vee, \supset, \equiv, \sim\&, \sim\vee, \sim\supset, \sim\equiv\}$

Keeping the Party Going

- Consider an arbitrary good tree 'Theodore' with k nodes ($k > 1$)
- Then there must be a good tree 'Theo' with $1 \leq n < k$ nodes such that 'Theodore' results from applying one of our nine STD rules to 'Theo'. Call this rule 'Ruby' $\in \{\sim, \&, \vee, \supset, \equiv, \sim\&, \sim\vee, \sim\supset, \sim\equiv\}$
- Since 'Theo' falls within the scope of our induction hypothesis, 'Theo' has at least one $\mathcal{I}$–branch, called 'Ida'

Keeping the Party Going

- Consider an arbitrary good tree 'Theodore' with k nodes ($k > 1$)
- Then there must be a good tree 'Theo' with $1 \leq n < k$ nodes such that 'Theodore' results from applying one of our nine STD rules to 'Theo'. Call this rule 'Ruby' $\in \{\sim, \&, \vee, \supset, \equiv, \sim\&, \sim\vee, \sim\supset, \sim\equiv\}$
- Since 'Theo' falls within the scope of our induction hypothesis, 'Theo' has at least one $\mathcal{I}$–branch, called 'Ida'
- Case a) Ruby extends a different open branch than Ida. Then Theodore still has $\mathcal{I}$–branch Ida, and so has an $\mathcal{I}$–branch

Keeping the Party Going

- Consider an arbitrary good tree 'Theodore' with k nodes ($k > 1$)
- Then there must be a good tree 'Theo' with $1 \leq n < k$ nodes such that 'Theodore' results from applying one of our nine STD rules to 'Theo'. Call this rule 'Ruby' $\in \{\sim, \&, \vee, \supset, \equiv, \sim\&, \sim\vee, \sim\supset, \sim\equiv\}$
- Since 'Theo' falls within the scope of our induction hypothesis, 'Theo' has at least one $\mathcal{I}$–branch, called 'Ida'
- Case a) Ruby extends a different open branch than Ida. Then Theodore still has $\mathcal{I}$–branch Ida, and so has an $\mathcal{I}$–branch
- Case b) Ruby extends Ida into branchlet(s). We need to show that for any rule, at least one of the branchlets is an $\mathcal{I}$–branch

The Simple way of Putting Case b)

- We have to show the following: if a good tree Theo contains an $\mathcal{I}$–branch, then so do all the trees that can be built from it by applying a single tree rule
(all the possible Theodore's, from all the possible Ruby's)

The Simple way of Putting Case b)

- We have to show the following: if a good tree Theo contains an $\mathcal{I}$–branch, then so do all the trees that can be built from it by applying a single tree rule
  (all the possible Theodore's, from all the possible Ruby's)
- i.e., each of our nine rules preserves the property of having at least one $\mathcal{I}$–branch (i.e. a branch that satisfies the root)

The Simple way of Putting Case b)

- We have to show the following: if a good tree Theo contains an $\mathcal{I}$–branch, then so do all the trees that can be built from it by applying a single tree rule
(all the possible Theodore's, from all the possible Ruby's)
- i.e., each of our nine rules preserves the property of having at least one $\mathcal{I}$–branch (i.e. a branch that satisfies the root)
- The book's induction step basically begins here at 'Case b'.
We've just justified why it's okay to jump right to Case b.

The Simple way of Putting Case b)

- We have to show the following: if a good tree Theo contains an $\mathcal{I}$-branch, then so do all the trees that can be built from it by applying a single tree rule
  (all the possible Theodore's, from all the possible Ruby's)
- i.e., each of our nine rules preserves the property of having at least one $\mathcal{I}$-branch (i.e. a branch that satisfies the root)
- The book's induction step basically begins here at 'Case b'. We've just justified why it's okay to jump right to Case b.
- This mirrors our lazy induction schema for SL: where we start with two arbitrary wffs Φ and Ψ that have the property, and show that any application of the SL recursion clause preserves the property

Subcase i) Ruby is Double Negation ($\sim$)

- Suppose Theo's $\mathcal{I}$–branch Ida contains $\sim\sim\Psi$

Subcase i) Ruby is Double Negation ($\sim$)

- Suppose Theo's $\mathcal{I}$–branch Ida contains $\sim\sim\Psi$
- Ruby extends Ida to 'Idaho' by adding Ψ

Subcase i) Ruby is Double Negation ($\sim$)

- Suppose Theo's $\mathcal{I}$–branch Ida contains $\sim\sim\Psi$
- Ruby extends Ida to 'Idaho' by adding Ψ
- By assumption, $\sim\sim\Psi$ is true according to $\mathcal{I}$

Subcase i) Ruby is Double Negation ($\sim$)

- Suppose Theo's $\mathcal{I}$–branch Ida contains $\sim\sim\Psi$
- Ruby extends Ida to 'Idaho' by adding Ψ
- By assumption, $\sim\sim\Psi$ is true according to $\mathcal{I}$

  $\Rightarrow$ Ψ is also true according to $\mathcal{I}$

Subcase i) Ruby is Double Negation ($\sim$)

- Suppose Theo's $\mathcal{I}$–branch Ida contains $\sim\sim\Psi$
- Ruby extends Ida to 'Idaho' by adding Ψ
- By assumption, $\sim\sim\Psi$ is true according to $\mathcal{I}$

  $\Rightarrow$ Ψ is also true according to $\mathcal{I}$

  - (Since otherwise, $\sim\Psi$ would be $\mathcal{I}$–true, but then $\sim\sim\Psi$ would be $\mathcal{I}$–false, which would contradict our assumption)

Subcase i) Ruby is Double Negation ($\sim$)

- Suppose Theo's $\mathcal{I}$–branch Ida contains $\sim\sim\Psi$
- Ruby extends Ida to 'Idaho' by adding Ψ
- By assumption, $\sim\sim\Psi$ is true according to $\mathcal{I}$

  $\Rightarrow$ Ψ is also true according to $\mathcal{I}$

  - (Since otherwise, $\sim\Psi$ would be $\mathcal{I}$–true, but then $\sim\sim\Psi$ would be $\mathcal{I}$–false, which would contradict our assumption)
  - So Idaho is also an $\mathcal{I}$–branch, this time of Theodore

Subcase i) Ruby is Double Negation ($\sim$)

- Suppose Theo's $\mathcal{I}$–branch Ida contains $\sim\sim\Psi$
- Ruby extends Ida to 'Idaho' by adding Ψ
- By assumption, $\sim\sim\Psi$ is true according to $\mathcal{I}$

  $\Rightarrow$ Ψ is also true according to $\mathcal{I}$

  - (Since otherwise, $\sim\Psi$ would be $\mathcal{I}$–true, but then $\sim\sim\Psi$ would be $\mathcal{I}$–false, which would contradict our assumption)
  - So Idaho is also an $\mathcal{I}$–branch, this time of Theodore

- Hence, our beloved Theodore contains an $\mathcal{I}$–branch

Subcase iii) Ruby is Negated Conjunction ($\sim\&$)

- Suppose Theo's $\mathcal{I}$-branch Ida contains $\sim(\Phi \,\&\, \Psi)$

Subcase iii) Ruby is Negated Conjunction ($\sim\&$)

- Suppose Theo's $\mathcal{I}$–branch Ida contains $\sim(\Phi \& \Psi)$
- Ruby extends Ida by splitting her into a branchlet with $\sim\Phi$ and a branchlet with $\sim\Psi$

Subcase iii) Ruby is Negated Conjunction ($\sim\&$)

- Suppose Theo's $\mathcal{I}$–branch Ida contains $\sim(\Phi \,\&\, \Psi)$
- Ruby extends Ida by splitting her into a branchlet with $\sim\Phi$ and a branchlet with $\sim\Psi$
- By assumption, $\sim(\Phi \,\&\, \Psi)$ is true according to $\mathcal{I}$

Subcase iii) Ruby is Negated Conjunction ($\sim\&$)

- Suppose Theo's $\mathcal{I}$-branch Ida contains $\sim(\Phi \& \Psi)$
- Ruby extends Ida by splitting her into a branchlet with $\sim\Phi$ and a branchlet with $\sim\Psi$
- By assumption, $\sim(\Phi \& \Psi)$ is true according to $\mathcal{I}$
    - So $\mathcal{I}$ cannot assign 0 to both $\sim\Phi$ and $\sim\Psi$, for if it did, then it would assign 1 to both Φ and Ψ, and hence 0 to $\sim(\Phi \& \Psi)$ (which would contradict our assumption)

Subcase iii) Ruby is Negated Conjunction ($\sim\&$)

- Suppose Theo's $\mathcal{I}$–branch Ida contains $\sim(\Phi \& \Psi)$
- Ruby extends Ida by splitting her into a branchlet with $\sim\Phi$ and a branchlet with $\sim\Psi$
- By assumption, $\sim(\Phi \& \Psi)$ is true according to $\mathcal{I}$
    - So $\mathcal{I}$ cannot assign 0 to both $\sim\Phi$ and $\sim\Psi$, for if it did, then it would assign 1 to both Φ and Ψ, and hence 0 to $\sim(\Phi \& \Psi)$ (which would contradict our assumption)
    - So $\mathcal{I}$ must make at least one of $\sim\Phi$ or $\sim\Psi$ true

Subcase iii) Ruby is Negated Conjunction ($\sim\&$)

- Suppose Theo's $\mathcal{I}$–branch Ida contains $\sim(\Phi \,\&\, \Psi)$
- Ruby extends Ida by splitting her into a branchlet with $\sim\Phi$ and a branchlet with $\sim\Psi$
- By assumption, $\sim(\Phi \,\&\, \Psi)$ is true according to $\mathcal{I}$
    - So $\mathcal{I}$ cannot assign 0 to both $\sim\Phi$ and $\sim\Psi$, for if it did, then it would assign 1 to both Φ and Ψ, and hence 0 to $\sim(\Phi \,\&\, \Psi)$ (which would contradict our assumption)
    - So $\mathcal{I}$ must make at least one of $\sim\Phi$ or $\sim\Psi$ true
    - So **at least one** of our two branchlets must be an $\mathcal{I}$–branch

Subcase iii) Ruby is Negated Conjunction ($\sim\&$)

- Suppose Theo's $\mathcal{I}$–branch Ida contains $\sim(\Phi \& \Psi)$
- Ruby extends Ida by splitting her into a branchlet with $\sim\Phi$ and a branchlet with $\sim\Psi$
- By assumption, $\sim(\Phi \& \Psi)$ is true according to $\mathcal{I}$
  - So $\mathcal{I}$ cannot assign 0 to both $\sim\Phi$ and $\sim\Psi$, for if it did, then it would assign 1 to both Φ and Ψ, and hence 0 to $\sim(\Phi \& \Psi)$ (which would contradict our assumption)
  - So $\mathcal{I}$ must make at least one of $\sim\Phi$ or $\sim\Psi$ true
  - So **at least one** of our two branchlets must be an $\mathcal{I}$–branch
- Hence, Theodore contains at least one $\mathcal{I}$–branch

5. Metalogic for STD

c. Testing Alternative Rules: Soundness

Modifying system STD

- What if we had chosen an alternative rule(s)?

Modifying system STD

- What if we had chosen an alternative rule(s)?
- Simplest case: we swap out one of our nine rules for a different rule, i.e. for a given connective or negated connective.

Modifying system STD

- What if we had chosen an alternative rule(s)?
- Simplest case: we swap out one of our nine rules for a different rule, i.e. for a given connective or negated connective.
- Call our modified system 'STD^*'

Modifying system STD

- What if we had chosen an alternative rule(s)?
- Simplest case: we swap out one of our nine rules for a different rule, i.e. for a given connective or negated connective.
- Call our modified system ‘STD^*’
- Would our modified system STD^* remain Sound?

Modifying system STD

- What if we had chosen an alternative rule(s)?
- Simplest case: we swap out one of our nine rules for a different rule, i.e. for a given connective or negated connective.
- Call our modified system ‘STD^*’
- Would our modified system STD^* remain Sound?
- Would our modified system STD^* remain Complete?

Checking Soundness: “top-down” reasoning

- **Heuristic for Soundness checking**: consider an **arbitrary** truth value assignment (TVA) that satisfies the sentence being resolved (i.e. *makes true* the sentence ‘at the top’ of the rule)

Checking Soundness: “top-down” reasoning

- **Heuristic for Soundness checking**: consider an **arbitrary** truth value assignment (TVA) that satisfies the sentence being resolved (i.e. *makes true* the sentence ‘at the top’ of the rule)
- Ask whether this TVA guarantees that **at least one** node below is satisfied, i.e. the sentences on that node are all made true

Checking Soundness: "top-down" reasoning

- **Heuristic for Soundness checking**: consider an **arbitrary** truth value assignment (TVA) that satisfies the sentence being resolved (i.e. *makes true* the sentence 'at the top' of the rule)
- Ask whether this TVA guarantees that **at least one** node below is satisfied, i.e. the sentences on that node are all made true
- If '**yes**', then the rule preserves soundness: proceed to extend our soundness proof for this case (reasoning in terms of a TVA '$\mathcal{I}$').

Checking Soundness: "top-down" reasoning

- **Heuristic for Soundness checking**: consider an **arbitrary** truth value assignment (TVA) that satisfies the sentence being resolved (i.e. *makes true* the sentence 'at the top' of the rule)
- Ask whether this TVA guarantees that **at least one** node below is satisfied, i.e. the sentences on that node are all made true
- If '**yes**', then the rule preserves soundness: proceed to extend our soundness proof for this case (reasoning in terms of a TVA '$\mathcal{I}$').
- If '**no**', then the rule BREAKS soundness:
  proceed to **construct a counter-example** using the rule.

Counterexamples to Soundness

- To show soundness fails, it is NECESSARY to provide a CONCRETE counterexample, using SL sentences

Counterexamples to Soundness

- To show soundness fails, it is NECESSARY to provide a CONCRETE counterexample, using SL sentences
  - Further heuristic reasoning about TVAs is not enough!

Counterexamples to Soundness

- To show soundness fails, it is NECESSARY to provide a CONCRETE counterexample, using SL sentences
  - Further heuristic reasoning about TVAs is not enough!
- What you need for a counterexample to soundness:

Counterexamples to Soundness

- To show soundness fails, it is NECESSARY to provide a CONCRETE counterexample, using SL sentences
  - Further heuristic reasoning about TVAs is not enough!

- What you need for a counterexample to soundness:
  1.) Choose a satisfiable root, i.e. a **consistent** set of sentences

Counterexamples to Soundness

- To show soundness fails, it is NECESSARY to provide a CONCRETE counterexample, using SL sentences
  - Further heuristic reasoning about TVAs is not enough!

- What you need for a counterexample to soundness:
  1.) Choose a satisfiable root, i.e. a **consistent** set of sentences
  2.) Apply the modified rule in the tree; you may also use other rules that we've already shown preserve soundness

Counterexamples to Soundness

- To show soundness fails, it is NECESSARY to provide a CONCRETE counterexample, using SL sentences
  - Further heuristic reasoning about TVAs is not enough!

- What you need for a counterexample to soundness:
  1.) Choose a satisfiable root, i.e. a **consistent** set of sentences
  2.) Apply the modified rule in the tree; you may also use other rules that we've already shown preserve soundness
  3.) Show that the tree **closes** (i.e. NO complete open branches)

Why such Counterexamples work

- **Sound**: If $\Gamma \vdash_{STD^*} \Theta$, then $\Gamma \vDash \Theta$

Why such Counterexamples work

- **Sound**: If $\Gamma \vdash_{STD^*} \Theta$, then $\Gamma \vDash \Theta$
- Counterexample: $\Gamma \vdash_{STD^*} \Theta$ but $\Gamma \nvDash \Theta$:

Why such Counterexamples work

- **Sound**: If $\Gamma \vdash_{STD^*} \Theta$, then $\Gamma \models \Theta$
- Counterexample: $\Gamma \vdash_{STD^*} \Theta$ but $\Gamma \not\models \Theta$:
  - '$\Gamma \vdash_{STD^*} \Theta$' means tree with root $\Gamma \cup \{\sim\Theta\}$ **CLOSES** (i.e. is tree–valid in system STD^*)

Why such Counterexamples work

- **Sound**: If $\Gamma \vdash_{STD*} \Theta$, then $\Gamma \vDash \Theta$
- Counterexample: $\Gamma \vdash_{STD*} \Theta$ but $\Gamma \nvDash \Theta$:
  - '$\Gamma \vdash_{STD*} \Theta$' means tree with root $\Gamma \cup \{\sim\Theta\}$ **CLOSES** (i.e. is tree–valid in system $STD*$)
  - '$\Gamma \nvDash \Theta$' means that $\Gamma \cup \{\sim\Theta\}$ is **CONSISTENT** (i.e. satisfiable) (i.e. there exists a TVA that makes the premises Γ true but the conclusion Θ false)

Why such Counterexamples work

- **Sound**: If $\Gamma \vdash_{STD^*} \Theta$, then $\Gamma \vDash \Theta$
- Counterexample: $\Gamma \vdash_{STD^*} \Theta$ but $\Gamma \nvDash \Theta$:
  - '$\Gamma \vdash_{STD^*} \Theta$' means tree with root $\Gamma \cup \{\sim\Theta\}$ **CLOSES** (i.e. is tree–valid in system STD^*)
  - '$\Gamma \nvDash \Theta$' means that $\Gamma \cup \{\sim\Theta\}$ is **CONSISTENT** (i.e. satisfiable) (i.e. there exists a TVA that makes the premises Γ true but the conclusion Θ false)
- If our system *were* sound, then whenever a tree closes, it *would* correspond to a semantically valid argument.

Liberal Conditional and Conservative Biconditional

*STD**: replace rule (⊃) with:

Liberal Conditional (L⊃)

m.	$\Phi \supset \Psi$		
⋮	⋮		
j.	$\sim\Phi$	$\Psi \vee \Phi$	m L⊃

Liberal Conditional and Conservative Biconditional

STD*: replace rule ($\supset$) with:

Liberal Conditional (L$\supset$)

m.	$\Phi \supset \Psi$		
$\vdots$	$\vdots$		
j.	$\sim\Phi$	$\Psi \vee \Phi$	m L$\supset$

STD†: replace rule ($\equiv$) with:

Conservative Biconditional (C$\equiv$)

m.	$\Phi \equiv \Psi$	
$\vdots$	$\vdots$	
j.	Φ	m C$\equiv$
j+1.	Ψ	

Liberal Conditional: Heuristic Reasoning

- The following is ‘heuristic reasoning’ using our rule; it is NOT a formal answer to the question:

Liberal Conditional: Heuristic Reasoning

- The following is ‘heuristic reasoning’ using our rule; it is NOT a formal answer to the question:
- Reason from the top–down: $\Phi \supset \Psi$ is logically equivalent to $\sim\Phi \vee \Psi$

Liberal Conditional: Heuristic Reasoning

- The following is ‘heuristic reasoning’ using our rule; it is NOT a formal answer to the question:
- Reason from the top–down: $\Phi \supset \Psi$ is logically equivalent to $\sim\Phi \vee \Psi$
- The sentences across the two nodes are logically equivalent to $(\sim\Phi) \vee (\Psi \vee \Phi)$

Liberal Conditional: Heuristic Reasoning

- The following is 'heuristic reasoning' using our rule; it is NOT a formal answer to the question:
- Reason from the top–down: $\Phi \supset \Psi$ is logically equivalent to $\sim\Phi \vee \Psi$
- The sentences across the two nodes are logically equivalent to $(\sim\Phi) \vee (\Psi \vee \Phi)$
- So note that the top entails the (disjunction of the) bottom! So given an interpretation that satisfies $\Phi \supset \Psi$, it must satisfy the sentence in at least one branch below.

Liberal Conditional: Heuristic Reasoning

- The following is 'heuristic reasoning' using our rule; it is NOT a formal answer to the question:
- Reason from the top-down: $\Phi \supset \Psi$ is logically equivalent to $\sim\Phi \vee \Psi$
- The sentences across the two nodes are logically equivalent to $(\sim\Phi) \vee (\Psi \vee \Phi)$
- So note that the top entails the (disjunction of the) bottom! So given an interpretation that satisfies $\Phi \supset \Psi$, it must satisfy the sentence in at least one branch below.
- Next, proceed to formally extend our soundness proof!

Liberal Conditional: Formal Answer

- **Formally**: assume $\Phi \supset \Psi$ is true according to $\mathcal{I}$.

Liberal Conditional: Formal Answer

- **Formally**: assume $\Phi \supset \Psi$ is true according to $\mathcal{I}$.
- Then $\mathcal{I}$ assigns 0 to Φ or 1 to Ψ (or both).

Liberal Conditional: Formal Answer

- **Formally**: assume $\Phi \supset \Psi$ is true according to $\mathcal{I}$.
- Then $\mathcal{I}$ assigns 0 to Φ or 1 to Ψ (or both).
- In the first case, $\mathcal{I}$ assigns 1 to $\sim\Phi$, satisfying the left branch.

Liberal Conditional: Formal Answer

- **Formally**: assume $\Phi \supset \Psi$ is true according to $\mathcal{I}$.
- Then $\mathcal{I}$ assigns 0 to Φ or 1 to Ψ (or both).
- In the first case, $\mathcal{I}$ assigns 1 to $\sim\Phi$, satisfying the left branch.
- In the second case, $\mathcal{I}$ makes $(\Psi \vee \Phi)$ true, so it satisfies the right branch.

Liberal Conditional: Formal Answer

- **Formally**: assume $\Phi \supset \Psi$ is true according to $\mathcal{I}$.
- Then $\mathcal{I}$ assigns 0 to Φ or 1 to Ψ (or both).
- In the first case, $\mathcal{I}$ assigns 1 to $\sim\Phi$, satisfying the left branch.
- In the second case, $\mathcal{I}$ makes $(\Psi \vee \Phi)$ true, so it satisfies the right branch.
- Either way, $\mathcal{I}$ satisfies the new sentences on **at least one** branch.

Liberal Conditional: Formal Answer

- **Formally**: assume $\Phi \supset \Psi$ is true according to $\mathcal{I}$.
- Then $\mathcal{I}$ assigns 0 to Φ or 1 to Ψ (or both).
- In the first case, $\mathcal{I}$ assigns 1 to $\sim\Phi$, satisfying the left branch.
- In the second case, $\mathcal{I}$ makes $(\Psi \vee \Phi)$ true, so it satisfies the right branch.
- Either way, $\mathcal{I}$ satisfies the new sentences on **at least one** branch.
- Hence, Liberal Conditional does not break soundness

Conservative Biconditional

- Reason from the top-down:
  $\Phi \equiv \Psi$ is logically equivalent to $(\Phi \,\&\, \Psi) \vee ({\sim}\Phi \,\&\, {\sim}\Psi)$

Conservative Biconditional

- Reason from the top-down:
  $\Phi \equiv \Psi$ is logically equivalent to $(\Phi \,\&\, \Psi) \vee ({\sim}\Phi \,\&\, {\sim}\Psi)$
- The node below is logically equivalent to $(\Phi \,\&\, \Psi)$

Conservative Biconditional

- Reason from the top-down:
  $\Phi \equiv \Psi$ is logically equivalent to $(\Phi \,\&\, \Psi) \vee (\sim\Phi \,\&\, \sim\Psi)$

- The node below is logically equivalent to $(\Phi \,\&\, \Psi)$

- Note that the top does NOT entail the bottom! Given an interpretation that satisfies $\Phi \equiv \Psi$, it is NOT guaranteed to satisfy the sentence(s) in at least one branch below.

Conservative Biconditional

- Reason from the top-down:
  $\Phi \equiv \Psi$ is logically equivalent to $(\Phi \,\&\, \Psi) \vee (\sim\Phi \,\&\, \sim\Psi)$

- The node below is logically equivalent to $(\Phi \,\&\, \Psi)$

- Note that the top does NOT entail the bottom! Given an interpretation that satisfies $\Phi \equiv \Psi$, it is NOT guaranteed to satisfy the sentence(s) in at least one branch below.

- Hence, **to the counterexample!**
  (note that the problem REQUIRES this! can't stop won't stop!)

Counterexample to Soundness of $STD^\dagger$

- For a counterexample, need: $\Gamma \vdash_{STD^\dagger} \Theta$ but $\Gamma \not\models \Theta$

Counterexample to Soundness of $STD^\dagger$

- For a counterexample, need: $\Gamma \vdash_{STD^\dagger} \Theta$ but $\Gamma \nvDash \Theta$

  $STD^\dagger$: replace rule ($\equiv$) with *Conservative Biconditional* (C$\equiv$):

1.	$P \equiv P$	Premise (PR)
2.	$\sim P$	$\sim$Conclusion
3.	P	1 C$\equiv$
4.	P	
	$\times$	
	2, 3, so tree-valid in $STD^\dagger$	

Counterexample to Soundness of $STD^{\dagger}$

- For a counterexample, need: $\Gamma \vdash_{STD^{\dagger}} \Theta$ but $\Gamma \nvDash \Theta$

  $STD^{\dagger}$: replace rule ($\equiv$) with *Conservative Biconditional* (C$\equiv$):

1.	$P \equiv P$	Premise (PR)
2.	$\sim P$	$\sim$Conclusion
3.	P	1 C$\equiv$
4.	P	
	$\times$	
	2, 3, so tree-valid in $STD^{\dagger}$	

- But $(P \equiv P) \nvDash P$ because $\{(P \equiv P), \sim P\}$ is consistent! Assign 'P' false! N.B.: your counterexample MUST use actual sentences of SL; not meta-variables!

5. Metalogic for STD

d. Completeness of System STD

Basic Proof Strategy for Completeness

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

Basic Proof Strategy for Completeness

- By proving that our tree system is *complete*, we will show that we never need truth tables: trees suffice
  - **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD} \Theta$

Basic Proof Strategy for Completeness

- By proving that our tree system is *complete*, we will show that we never need truth tables: trees suffice
  - **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD} \Theta$
  - (semantic notions are fully covered by our syntactic rules)

Basic Proof Strategy for Completeness

- By proving that our tree system is *complete*, we will show that we never need truth tables: trees suffice
  - **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD} \Theta$
  - (semantic notions are fully covered by our syntactic rules)
  - (Means: we wrote down *enough* rules!)

Basic Proof Strategy for Completeness

- By proving that our tree system is *complete*, we will show that we never need truth tables: trees suffice
  - **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD} \Theta$
  - (semantic notions are fully covered by our syntactic rules)
  - (Means: we wrote down *enough* rules!)

- We will prove the contrapositive, using induction:

Basic Proof Strategy for Completeness

- By proving that our tree system is *complete*, we will show that we never need truth tables: trees suffice
  - **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD} \Theta$
  - (semantic notions are fully covered by our syntactic rules)
  - (Means: we wrote down *enough* rules!)

- We will prove the contrapositive, using induction:
- **Complete**: If $\Gamma \nvdash_{STD} \Theta$, then $\Gamma \nvDash \Theta$

Basic Proof Strategy for Completeness

- By proving that our tree system is *complete*, we will show that we never need truth tables: trees suffice
  - **Complete**: If $\Gamma \models \Theta$, then $\Gamma \vdash_{STD} \Theta$
  - (semantic notions are fully covered by our syntactic rules)
  - (Means: we wrote down *enough* rules!)

- We will prove the contrapositive, using induction:
- **Complete**: If $\Gamma \nvdash_{STD} \Theta$, then $\Gamma \not\models \Theta$
- "if an argument is not tree-valid, then it's not semantically valid"

Fleshing out the Proof Idea

- “Not tree-valid” means that EVERY tree with root $\Gamma \cup \{\sim\Theta\}$ remains **open** (i.e. never closes, even after resolving every sentence)

Fleshing out the Proof Idea

- "Not tree-valid" means that EVERY tree with root $\Gamma \cup \{\sim\Theta\}$ remains **open** (i.e. never closes, even after resolving every sentence)
- Call this open branch '**Oprah**' (or 'O' for short)

Fleshing out the Proof Idea

- "Not tree-valid" means that EVERY tree with root $\Gamma \cup \{\sim\Theta\}$ remains **open** (i.e. never closes, even after resolving every sentence)
- Call this open branch '**Oprah**' (or 'O' for short)
- Use Oprah to define a TVA '$\mathcal{I}$' that makes true every wff on O

Fleshing out the Proof Idea

- "Not tree-valid" means that EVERY tree with root $\Gamma \cup \{\sim\Theta\}$ remains **open** (i.e. never closes, even after resolving every sentence)
- Call this open branch '**Oprah**' (or 'O' for short)
- Use Oprah to define a TVA '$\mathcal{I}$' that makes true every wff on O
- So in particular, $\mathcal{I}$ will make true the sentences in the root, showing they are **consistent** $\Rightarrow$ argument is semantically invalid: i.e. $\mathcal{I}$ makes Γ true and $\sim\Theta$ true, i.e. Θ FALSE

Fleshing out the Proof Idea

- "Not tree–valid" means that EVERY tree with root $\Gamma \cup \{\sim\Theta\}$ remains **open** (i.e. never closes, even after resolving every sentence)
- Call this open branch '**Oprah**' (or 'O' for short)
- Use Oprah to define a TVA '$\mathcal{I}$' that makes true every wff on O
- So in particular, $\mathcal{I}$ will make true the sentences in the root, showing they are **consistent** $\Rightarrow$ argument is semantically invalid: i.e. $\mathcal{I}$ makes Γ true and $\sim\Theta$ true, i.e. Θ FALSE
- This will prove: **Completeness**: If $\Gamma \nvdash_{STD} \Theta$, then $\Gamma \nvDash \Theta$

Starting the Completeness Proof!

- Consider an arbitrary argument with premise set Γ and putative conclusion Θ

Starting the Completeness Proof!

- Consider an arbitrary argument with premise set Γ and putative conclusion Θ
- Construct a tree with root $\Gamma \cup \{\sim\Theta\}$

Starting the Completeness Proof!

- Consider an arbitrary argument with premise set Γ and putative conclusion Θ
- Construct a tree with root $\Gamma \cup \{{\sim}\Theta\}$
- Assume that this argument is NOT tree valid: so every such tree is open and extends to a complete open tree

Starting the Completeness Proof!

- Consider an arbitrary argument with premise set Γ and putative conclusion Θ
- Construct a tree with root $\Gamma \cup \{\sim\Theta\}$
- Assume that this argument is NOT tree valid: so every such tree is open and extends to a complete open tree
- Hence, there must be at least one complete open branch: O

Starting the Completeness Proof!

- Consider an arbitrary argument with premise set Γ and putative conclusion Θ
- Construct a tree with root $\Gamma \cup \{\sim\Theta\}$
- Assume that this argument is NOT tree valid: so every such tree is open and extends to a complete open tree
- Hence, there must be at least one complete open branch: O
- Notice that O must end with atomic sentences or negations of these in its last node

Starting the Completeness Proof!

- Consider an arbitrary argument with premise set Γ and putative conclusion Θ
- Construct a tree with root $\Gamma \cup \{\sim\Theta\}$
- Assume that this argument is NOT tree valid: so every such tree is open and extends to a complete open tree
- Hence, there must be at least one complete open branch: O
- Notice that O must end with atomic sentences or negations of these in its last node
- (So we see that we could do induction, starting 'from the bottom' of our tree)

Setting up the Induction (HW question!)

- At this point, we would need to (i) decide what we want to do induction over

Setting up the Induction (HW question!)

- At this point, we would need to (i) decide what we want to do induction over
- (ii) Handle the base case(s)

Setting up the Induction (HW question!)

- At this point, we would need to (i) decide what we want to do induction over
- (ii) Handle the base case(s)
- (iii) State the induction hypothesis

Setting up the Induction (HW question!)

- At this point, we would need to (i) decide what we want to do induction over
- (ii) Handle the base case(s)
- (iii) State the induction hypothesis
- (iv) Proceed to handle all cases in the induction step (coming from a recursive definition(s))

Setting up the Induction (HW question!)

- At this point, we would need to (i) decide what we want to do induction over
- (ii) Handle the base case(s)
- (iii) State the induction hypothesis
- (iv) Proceed to handle all cases in the induction step (coming from a recursive definition(s))
- We'll leave this to an optional HW question and proceed casually

Using Oprah to define $\mathcal{I}$

- Since Oprah is open, no wff and its negation ever appear on it

Using Oprah to define $\mathcal{I}$

- Since Oprah is open, no wff and its negation ever appear on it
- Hence, we can define a TVA '$\mathcal{I}$' as follows:

Using Oprah to define $\mathcal{I}$

- Since Oprah is open, no wff and its negation ever appear on it
- Hence, we can define a TVA ‘$\mathcal{I}$’ as follows:
    - $\mathcal{I}$ assigns ‘False’ to an atomic sentence if and only if its negation appears by itself on a node of O

Using Oprah to define $\mathcal{I}$

- Since Oprah is open, no wff and its negation ever appear on it
- Hence, we can define a TVA '$\mathcal{I}$' as follows:
  - $\mathcal{I}$ assigns 'False' to an atomic sentence if and only if its negation appears by itself on a node of O
  - Hence, $\mathcal{I}$ assigns 'True' to every atomic sentence appearing by itself on one of O's nodes

Idea: show $\mathcal{I}$ makes true every wff on O

- Consider an arbitrary wff Δ on Oprah

Idea: show $\mathcal{I}$ makes true every wff on O

- Consider an arbitrary wff Δ on Oprah
- If Δ is an atomic sentence or negated atomic sentence, then $\mathcal{I}$ makes it true by definition

Idea: show $\mathcal{I}$ makes true every wff on O

- Consider an arbitrary wff Δ on Oprah
- If Δ is an atomic sentence or negated atomic sentence, then $\mathcal{I}$ makes it true by definition
- Otherwise, there must exist wff α and β that compose Δ according to our recursive definition of SL sentences

Idea: show $\mathcal{I}$ makes true every wff on O

- Consider an arbitrary wff Δ on Oprah
- If Δ is an atomic sentence or negated atomic sentence, then $\mathcal{I}$ makes it true by definition
- Otherwise, there must exist wff α and β that compose Δ according to our recursive definition of SL sentences
- Since Oprah is complete, Δ must be resolved according to one of our nine tree rules, leading to sentences on the 'child nodes', and one of these child nodes must lie on Oprah

Idea: show $\mathcal{I}$ makes true every wff on O

- Consider an arbitrary wff Δ on Oprah
- If Δ is an atomic sentence or negated atomic sentence, then $\mathcal{I}$ makes it true by definition
- Otherwise, there must exist wff α and β that compose Δ according to our recursive definition of SL sentences
- Since Oprah is complete, Δ must be resolved according to one of our nine tree rules, leading to sentences on the 'child nodes', and one of these child nodes must lie on Oprah
- By induction, we assume that the child node on O is satisfied by $\mathcal{I}$

Idea: show $\mathcal{I}$ makes true every wff on O

- Consider an arbitrary wff Δ on Oprah
- If Δ is an atomic sentence or negated atomic sentence, then $\mathcal{I}$ makes it true by definition
- Otherwise, there must exist wff α and β that compose Δ according to our recursive definition of SL sentences
- Since Oprah is complete, Δ must be resolved according to one of our nine tree rules, leading to sentences on the 'child nodes', and one of these child nodes must lie on Oprah
- By induction, we assume that the child node on O is satisfied by $\mathcal{I}$
- Given this, we aim to show that $\mathcal{I}$ makes Δ true, **no matter which** child node lies on Oprah and is satisfied by $\mathcal{I}$

What we can conclude AFTER the subcases:

- Assuming we complete our heroic quest, we get to conclude that $\mathcal{I}$ makes Δ true

What we can conclude AFTER the subcases:

- Assuming we complete our heroic quest, we get to conclude that $\mathcal{I}$ makes Δ true
- But Δ was an arbitrary wff on Oprah

What we can conclude AFTER the subcases:

- Assuming we complete our heroic quest, we get to conclude that $\mathcal{I}$ makes Δ true
- But Δ was an arbitrary wff on Oprah
- So we'll have shown that $\mathcal{I}$ makes each sentence in the root true

What we can conclude AFTER the subcases:

- Assuming we complete our heroic quest, we get to conclude that $\mathcal{I}$ makes Δ true
- But Δ was an arbitrary wff on Oprah
- So we'll have shown that $\mathcal{I}$ makes each sentence in the root true
- By showing that the root is **satisfiable**, $\mathcal{I}$ shows that the argument is **semantically invalid**, which is what we wanted to show!

What we can conclude AFTER the subcases:

- Assuming we complete our heroic quest, we get to conclude that $\mathcal{I}$ makes Δ true
- But Δ was an arbitrary wff on Oprah
- So we'll have shown that $\mathcal{I}$ makes each sentence in the root true
- By showing that the root is **satisfiable**, $\mathcal{I}$ shows that the argument is **semantically invalid**, which is what we wanted to show!
- We'll have used Oprah to define a TVA that makes the premises true but the conclusion false

Reasoning from the ‘Bottom Up’

- Notice that we are reasoning from the sentences *below* the resolved sentence, back up the tree

Reasoning from the 'Bottom Up'

- Notice that we are reasoning from the sentences *below* the resolved sentence, back up the tree
- We need to show that *for **EACH** child node*, if $\mathcal{I}$ satisfies it, then $\mathcal{I}$ makes the resolved sentence Δ true

Reasoning from the ‘Bottom Up’

- Notice that we are reasoning from the sentences *below* the resolved sentence, back up the tree
- We need to show that *for **EACH** child node*, if $\mathcal{I}$ satisfies it, then $\mathcal{I}$ makes the resolved sentence Δ true
- Effectively, we are showing that each child node separately entails the resolved sentence

Reasoning from the 'Bottom Up'

- Notice that we are reasoning from the sentences *below* the resolved sentence, back up the tree
- We need to show that *for* ***EACH*** *child node*, if $\mathcal{I}$ satisfies it, then $\mathcal{I}$ makes the resolved sentence Δ true
- Effectively, we are showing that each child node separately entails the resolved sentence
- There are nine cases to consider, coming from our nine tree rules

Subcase i) Δ is resolved by Double Negation ($\sim$)

Double Negation ($\sim$)

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

- Only one child node $\Rightarrow \Phi$ lies on O

Subcase i) Δ is resolved by Double Negation ($\sim$)

Double Negation ($\sim$)

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

- Only one child node $\Rightarrow \Phi$ lies on O
- By induction hypothesis, $\mathcal{I}$ makes Φ true

Subcase i) Δ is resolved by Double Negation ($\sim$)

Double Negation ($\sim$)

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

- Only one child node $\Rightarrow \Phi$ lies on O
- By induction hypothesis, $\mathcal{I}$ makes Φ true
- Hence, $\mathcal{I}$ must assign true to $\sim\sim\Phi$, which in this case is Δ

Subcase i) Δ is resolved by Double Negation ($\sim$)

Double Negation ($\sim$)

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

- Only one child node $\Rightarrow \Phi$ lies on O
- By induction hypothesis, $\mathcal{I}$ makes Φ true
- Hence, $\mathcal{I}$ must assign true to $\sim\sim\Phi$, which in this case is Δ
- So $\mathcal{I}$ satisfies Δ

Subcase i) Δ is resolved by Double Negation ($\sim$)

Double Negation ($\sim$)

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

- Only one child node $\Rightarrow \Phi$ lies on O
- By induction hypothesis, $\mathcal{I}$ makes Φ true
- Hence, $\mathcal{I}$ must assign true to $\sim\sim\Phi$, which in this case is Δ
- So $\mathcal{I}$ satisfies Δ
- On to the next one!

Subcase iii) Δ is resolved by Negated Conjunction ($\sim\&$)

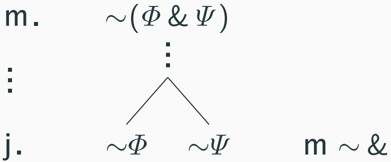
Negated Conjunction ($\sim\&$)

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

- Exactly one child node lies on O, but we don't know which one (so we must show $\mathcal{I}$ satisfies Δ either way!)

Subcase iii) Δ is resolved by Negated Conjunction ($\sim\&$)

Negated Conjunction ($\sim\&$)



- Exactly one child node lies on O, but we don't know which one (so we must show $\mathcal{I}$ satisfies Δ either way!)
- a) If $\sim\Phi$ lies on O, then $\mathcal{I}$ makes $\sim\Phi$ true (by induction hypothesis)

Subcase iii) Δ is resolved by Negated Conjunction ($\sim\&$)

Negated Conjunction ($\sim\&$)

$$
\begin{array}{llll}
\text{m.} & & \sim(\Phi \,\&\, \Psi) & \\
\vdots & & \vdots & \\
\text{j.} & \sim\Phi & \sim\Psi & \text{m} \sim\&
\end{array}
$$

- Exactly one child node lies on O, but we don't know which one (so we must show $\mathcal{I}$ satisfies Δ either way!)
- a) If $\sim\Phi$ lies on O, then $\mathcal{I}$ makes $\sim\Phi$ true (by induction hypothesis)
  $\Rightarrow \mathcal{I}$ makes Φ false

Subcase iii) Δ is resolved by Negated Conjunction ($\sim\&$)

Negated Conjunction ($\sim\&$)

m. $\sim(\Phi \,\&\, \Psi)$

$\vdots$

j. $\sim\Phi \qquad \sim\Psi \qquad$ m $\sim\&$

- Exactly one child node lies on O, but we don't know which one (so we must show $\mathcal{I}$ satisfies Δ either way!)
- a) If $\sim\Phi$ lies on O, then $\mathcal{I}$ makes $\sim\Phi$ true (by induction hypothesis)
  $\Rightarrow \mathcal{I}$ makes Φ false
  $\Rightarrow \mathcal{I}$ makes $(\Phi \,\&\, \Psi)$ false, and hence makes Δ true

Subcase iii) Δ is resolved by Negated Conjunction ($\sim\&$)

Negated Conjunction ($\sim\&$)

m. $\sim(\Phi \,\&\, \Psi)$

$\vdots$

j. $\sim\Phi$ $\quad\sim\Psi$ $\quad$ m $\sim\&$

- Exactly one child node lies on O, but we don't know which one (so we must show $\mathcal{I}$ satisfies Δ either way!)
- a) If $\sim\Phi$ lies on O, then $\mathcal{I}$ makes $\sim\Phi$ true (by induction hypothesis)
  $\Rightarrow \mathcal{I}$ makes Φ false
  $\Rightarrow \mathcal{I}$ makes $(\Phi \,\&\, \Psi)$ false, and hence makes Δ true
- b) Likewise if $\sim\Psi$ lies on O...

Subcase iii) Δ is resolved by Negated Conjunction ($\sim\&$)

Negated Conjunction ($\sim\&$)

$$
\begin{array}{lccc}
\text{m.} & \multicolumn{2}{c}{\sim(\Phi \,\&\, \Psi)} & \\
\vdots & \multicolumn{2}{c}{\vdots} & \\
\text{j.} & \sim\Phi & \sim\Psi & \text{m} \sim\&
\end{array}
$$

- Exactly one child node lies on O, but we don't know which one (so we must show $\mathcal{I}$ satisfies Δ either way!)
- a) If $\sim\Phi$ lies on O, then $\mathcal{I}$ makes $\sim\Phi$ true (by induction hypothesis)
  $\Rightarrow \mathcal{I}$ makes Φ false
  $\Rightarrow \mathcal{I}$ makes $(\Phi \,\&\, \Psi)$ false, and hence makes Δ true
- b) Likewise if $\sim\Psi$ lies on O...
- Either way, $\mathcal{I}$ satisfies Δ

Subcase v) Δ is resolved by Negated Disjunction ($\sim\vee$)

Negated Disjunction ($\sim\vee$)

$$
\begin{array}{lll}
\text{m.} & \sim(\Phi \vee \Psi) & \\
\vdots & \vdots & \\
 & | & \\
\text{j.} & \sim\Phi & \text{m} \sim\vee \\
\text{j+1.} & \sim\Psi &
\end{array}
$$

- Only one child node $\Rightarrow$ both $\sim\Phi$ and $\sim\Psi$ lie on O

Subcase v) Δ is resolved by Negated Disjunction ($\sim\vee$)

Negated Disjunction ($\sim\vee$)

$$
\begin{array}{lcl}
\text{m.} & \sim(\Phi \vee \Psi) & \\
\vdots & \vdots & \\
 & | & \\
\text{j.} & \sim\Phi & \text{m } \sim\vee \\
\text{j+1.} & \sim\Psi &
\end{array}
$$

- Only one child node $\Rightarrow$ both $\sim\Phi$ and $\sim\Psi$ lie on O
- By induction hypothesis, $\mathcal{I}$ makes both children wff true

Subcase v) Δ is resolved by Negated Disjunction ($\sim\vee$)

Negated Disjunction ($\sim\vee$)

$$
\begin{array}{lcl}
\text{m.} & \sim(\Phi \vee \Psi) & \\
\vdots & \vdots & \\
 & | & \\
\text{j.} & \sim\Phi & \text{m } \sim\vee \\
\text{j+1.} & \sim\Psi &
\end{array}
$$

- Only one child node $\Rightarrow$ both $\sim\Phi$ and $\sim\Psi$ lie on O
- By induction hypothesis, $\mathcal{I}$ makes both children wff true
- Hence, $\mathcal{I}$ must make false both Φ and Ψ

Subcase v) Δ is resolved by Negated Disjunction ($\sim\vee$)

Negated Disjunction ($\sim\vee$)

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

- Only one child node $\Rightarrow$ both $\sim\Phi$ and $\sim\Psi$ lie on O
- By induction hypothesis, $\mathcal{I}$ makes both children wff true
- Hence, $\mathcal{I}$ must make false both Φ and Ψ
  $\Rightarrow \mathcal{I}$ makes false $(\Phi \vee \Psi)$

Subcase v) Δ is resolved by Negated Disjunction ($\sim\vee$)

Negated Disjunction ($\sim\vee$)

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

- Only one child node $\Rightarrow$ both $\sim\Phi$ and $\sim\Psi$ lie on O
- By induction hypothesis, $\mathcal{I}$ makes both children wff true
- Hence, $\mathcal{I}$ must make false both Φ and Ψ
  $\Rightarrow \mathcal{I}$ makes false $(\Phi \vee \Psi)$
  $\Rightarrow \mathcal{I}$ makes true $\sim(\Phi \vee \Psi)$

Subcase v) Δ is resolved by Negated Disjunction ($\sim\vee$)

Negated Disjunction ($\sim\vee$)

$$
\begin{array}{lcl}
\text{m.} & \sim(\Phi \vee \Psi) & \\
 & \vdots & \\
\vdots & | & \\
\text{j.} & \sim\Phi & \text{m } \sim\vee \\
\text{j+1.} & \sim\Psi &
\end{array}
$$

- Only one child node $\Rightarrow$ both $\sim\Phi$ and $\sim\Psi$ lie on O
- By induction hypothesis, $\mathcal{I}$ makes both children wff true
- Hence, $\mathcal{I}$ must make false both Φ and Ψ
  $\Rightarrow \mathcal{I}$ makes false $(\Phi \vee \Psi)$
  $\Rightarrow \mathcal{I}$ makes true $\sim(\Phi \vee \Psi)$
- So $\mathcal{I}$ satisfies Δ

A Key fact about Complete Open Trees

- A complete open tree has at least one complete open branch

A Key fact about Complete Open Trees

- A complete open tree has at least one complete open branch
- A complete open branch never contains both a sentence Φ and its negation $\sim\Phi$

A Key fact about Complete Open Trees

- A complete open tree has at least one complete open branch
- A complete open branch never contains both a sentence Φ and its negation $\sim\Phi$
- In a 'partially complete system', **our construction above lets us define a TVA '$\mathcal{I}$' that makes true each sentence on a complete open branch**, including the root

A Key fact about Complete Open Trees

- A complete open tree has at least one complete open branch
- A complete open branch never contains both a sentence Φ and its negation $\sim\Phi$
- In a 'partially complete system', **our construction above lets us define a TVA '$\mathcal{I}$' that makes true each sentence on a complete open branch**, including the root
- Upshot: if an argument is tree-invalid (i.e. has at least one complete open tree) in a 'partially complete' system , then there is a truth-value assignment that satisfies the root (so $\Gamma \nvDash \Theta$)

A Key fact about Complete Open Trees

- A complete open tree has at least one complete open branch
- A complete open branch never contains both a sentence Φ and its negation $\sim\Phi$
- In a 'partially complete system', **our construction above lets us define a TVA '$\mathcal{I}$' that makes true each sentence on a complete open branch**, including the root
- Upshot: if an argument is tree-invalid (i.e. has at least one complete open tree) in a 'partially complete' system , then there is a truth-value assignment that satisfies the root (so $\Gamma \nvDash \Theta$)
- If the system is also SOUND, one could then conclude that the argument is *not* tree-valid.

What are the ‘partially complete’ systems?

- Our construction of a TVA that satisfies each wff on the complete open branch just requires that *for the rules we use*, the sentences on each child node entail the sentence being resolved.

What are the ‘partially complete’ systems?

- Our construction of a TVA that satisfies each wff on the complete open branch just requires that *for the rules we use*, the sentences on each child node entail the sentence being resolved.
- Call systems “partially complete” if they have this property

What are the ‘partially complete’ systems?

- Our construction of a TVA that satisfies each wff on the complete open branch just requires that *for the rules we use*, the sentences on each child node entail the sentence being resolved.
- Call systems “partially complete” if they have this property
- Note that the system STD^{conj} which has only the rules ($\sim$), ($\&$), and ($\sim\&$) is partially complete in this sense:

What are the ‘partially complete’ systems?

- Our construction of a TVA that satisfies each wff on the complete open branch just requires that *for the rules we use*, the sentences on each child node entail the sentence being resolved.
- Call systems “partially complete” if they have this property
- Note that the system STD^{conj} which has only the rules ($\sim$), (&), and ($\sim$ &) is partially complete in this sense:
- for the sentences that *can be* completely resolved with these limited rules, we can turn any complete open branch into a TVA that satisfies all wff on the branch.

5. Metalogic for STD

e. Testing Alternative Rules: Completeness

Checking Completeness: “bottom-up” reasoning

- **Heuristic for Completeness checking**: ask whether **EACH** child node (below) individually entails the sentence being resolved (i.e. the sentence ‘up top’)

Checking Completeness: "bottom-up" reasoning

- **Heuristic for Completeness checking**: ask whether **EACH** child node (below) individually entails the sentence being resolved (i.e. the sentence 'up top')

- If '**yes**', then the rule preserves completeness: proceed to extend our completeness proof for this case (reasoning in terms of arbitrary TVA's, one for each child node).

Checking Completeness: "bottom-up" reasoning

- **Heuristic for Completeness checking**: ask whether **EACH** child node (below) individually entails the sentence being resolved (i.e. the sentence 'up top')
- If '**yes**', then the rule preserves completeness: proceed to extend our completeness proof for this case (reasoning in terms of arbitrary TVA's, one for each child node).
- If '**no**', then the rule BREAKS completeness: proceed to **construct a counter-example** using the rule.

Counterexamples to Completeness

- To show completeness fails, it is NECESSARY to provide a CONCRETE counterexample, using SL sentences

Counterexamples to Completeness

- To show completeness fails, it is NECESSARY to provide a CONCRETE counterexample, using SL sentences
  - Further heuristic reasoning about TVAs is not enough!

Counterexamples to Completeness

- To show completeness fails, it is NECESSARY to provide a CONCRETE counterexample, using SL sentences
    - Further heuristic reasoning about TVAs is not enough!
- What you need for a counterexample to completeness:

Counterexamples to Completeness

- To show completeness fails, it is NECESSARY to provide a CONCRETE counterexample, using SL sentences
  - Further heuristic reasoning about TVAs is not enough!

- What you need for a counterexample to completeness:

1.) Choose an **UNsatisfiable root**, i.e. an INconsistent set of sentences

Counterexamples to Completeness

- To show completeness fails, it is NECESSARY to provide a CONCRETE counterexample, using SL sentences
  - Further heuristic reasoning about TVAs is not enough!

- What you need for a counterexample to completeness:
  1.) Choose an **UNsatisfiable root**, i.e. an INconsistent set of sentences
  2.) Apply the modified rule in the tree; you may also use other rules that we've already shown preserve completeness

Counterexamples to Completeness

- To show completeness fails, it is NECESSARY to provide a CONCRETE counterexample, using SL sentences
  - Further heuristic reasoning about TVAs is not enough!

- What you need for a counterexample to completeness:
  1.) Choose an **UNsatisfiable root**, i.e. an INconsistent set of sentences
  2.) Apply the modified rule in the tree; you may also use other rules that we've already shown preserve completeness
  3.) Show that the tree has a **complete open branch**, i.e. does NOT close

Why such Counterexamples work

- **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD^*} \Theta$

Why such Counterexamples work

- **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD*} \Theta$
- Counterexample: $\Gamma \vDash \Theta$ but $\Gamma \nvdash_{STD*} \Theta$

Why such Counterexamples work

- **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD^*} \Theta$
- Counterexample: $\Gamma \vDash \Theta$ but $\Gamma \nvdash_{STD^*} \Theta$
  - '$\Gamma \vDash \Theta$' means that $\Gamma \cup \{\sim\Theta\}$ is INconsistent (i.e. UNsatisfiable) (i.e. any TVA that makes Γ true makes the conclusion Θ true)

Why such Counterexamples work

- **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD^*} \Theta$
- Counterexample: $\Gamma \vDash \Theta$ but $\Gamma \nvdash_{STD^*} \Theta$
  - '$\Gamma \vDash \Theta$' means that $\Gamma \cup \{\sim\Theta\}$ is INconsistent (i.e. UNsatisfiable) (i.e. any TVA that makes Γ true makes the conclusion Θ true)
  - '$\Gamma \nvdash_{STD^*} \Theta$' means that it is NOT the case that the argument is tree-valid in STD^* (a claim about ALL trees)

Why such Counterexamples work

- **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD^*} \Theta$
- Counterexample: $\Gamma \vDash \Theta$ but $\Gamma \nvdash_{STD^*} \Theta$
  - '$\Gamma \vDash \Theta$' means that $\Gamma \cup \{\sim\Theta\}$ is INconsistent (i.e. UNsatisfiable) (i.e. any TVA that makes Γ true makes the conclusion Θ true)
  - '$\Gamma \nvdash_{STD^*} \Theta$' means that it is NOT the case that the argument is tree-valid in STD^* (a claim about ALL trees)
  - From completeness proof, we know that if the system *were* complete, then we could use any **complete open branch** to construct a TVA that **satisfies** the root

Why such Counterexamples work

- **Complete**: If $\Gamma \vDash \Theta$, then $\Gamma \vdash_{STD^*} \Theta$
- Counterexample: $\Gamma \vDash \Theta$ but $\Gamma \nvdash_{STD^*} \Theta$
  - '$\Gamma \vDash \Theta$' means that $\Gamma \cup \{\sim\Theta\}$ is INconsistent (i.e. UNsatisfiable) (i.e. any TVA that makes Γ true makes the conclusion Θ true)
  - '$\Gamma \nvdash_{STD^*} \Theta$' means that it is NOT the case that the argument is tree–valid in STD^* (a claim about ALL trees)
  - From completeness proof, we know that if the system *were* complete, then we could use any **complete open branch** to construct a TVA that **satisfies** the root
  - Hence, if we construct a **complete open tree** with an **unsatisfiable root**, this is a reductio of completeness

Liberal Conditional and Conservative Biconditional

STD*: replace rule ($\supset$) with:

Liberal Conditional (L$\supset$)

m.	$\Phi \supset \Psi$		
$\vdots$	$\vdots$		
j.	$\sim\Phi$	$\Psi \vee \Phi$	m L$\supset$

STD†: replace rule ($\equiv$) with:

Conservative Biconditional (C$\equiv$)

m.	$\Phi \equiv \Psi$	
$\vdots$	$\vdots$	
j.	Φ	m C$\equiv$
j+1.	Ψ	

Liberal Conditional (now from the bottom up!)

- Reason from the bottom-up, handling each child node separately

Liberal Conditional (now from the bottom up!)

- Reason from the bottom-up, handling each child node separately
- $\sim\Phi$ entails $\Phi \supset \Psi$

Liberal Conditional (now from the bottom up!)

- Reason from the bottom–up, handling each child node separately
- $\sim\Phi$ entails $\Phi \supset \Psi$
- $(\Psi \vee \Phi)$ does NOT entail $\Phi \supset \Psi$, since Φ doesn't entail it

Liberal Conditional (now from the bottom up!)

- Reason from the bottom–up, handling each child node separately
- $\sim\Phi$ entails $\Phi \supset \Psi$
- $(\Psi \vee \Phi)$ does NOT entail $\Phi \supset \Psi$, since Φ doesn't entail it
- (Remember: $\Phi \supset \Psi$ is equivalent to $\sim\Phi \vee \Psi$)

Liberal Conditional (now from the bottom up!)

- Reason from the bottom–up, handling each child node separately
- $\sim\Phi$ entails $\Phi \supset \Psi$
- $(\Psi \vee \Phi)$ does NOT entail $\Phi \supset \Psi$, since Φ doesn't entail it
- (Remember: $\Phi \supset \Psi$ is equivalent to $\sim\Phi \vee \Psi$)
- Hence, we **proceed to counterexample!**

Counterexample to Completeness of STD^*

- Counterexample: find complete open tree with unsatisfiable root:

Counterexample to Completeness of STD^*

- Counterexample: find complete open tree with unsatisfiable root:

1.	$P \supset Q$ ✓		
2.	P		
3.	$\sim Q$		$\sim$Conclusion
4.	$\sim P$	$Q \vee P$ ✓	1 L$\supset$
	$\times$ 2, 4		
5.		Q P	4 $\vee$
		$\times$ 3, 5 $\uparrow$	

Counterexample to Completeness of STD^*

- Counterexample: find complete open tree with unsatisfiable root:

1.		$P \supset Q$ ✓		
2.		P		
3.		$\sim Q$		$\sim$Conclusion
4.	$\sim P$	$Q \vee P$ ✓		1 $\mathrm{L}\supset$
	$\times$ 2, 4			
5.		Q	P	4 $\vee$
		$\times$ 3, 5	$\uparrow$	

- Note that $\{P \supset Q,\ P\} \vDash Q$ but **tree-invalid** in STD^*

Counterexample to Completeness of STD^*

- Counterexample: find complete open tree with unsatisfiable root:

1.	$P \supset Q$ ✓			
2.	P			
3.	$\sim Q$			$\sim$Conclusion
4.	$\sim P$	$Q \vee P$ ✓		1 L$\supset$
	$\times$ 2, 4			
5.		Q	P	4 $\vee$
		$\times$ 3, 5	$\uparrow$	

- Note that $\{P \supset Q,\ P\} \vDash Q$ but **tree-invalid** in STD^*
- If STD^* *were* complete, the complete open branch would lead to a TVA that satisfies the root. Contradiction $\Rightarrow$ not complete

A Common Mistake to Avoid!!!!

- You are very liable to forget to COMPLETE YOUR open branch!!!

A Common Mistake to Avoid!!!!

- You are very liable to forget to COMPLETE YOUR open branch!!!
- You only have a counterexample to completeness if you have a COMPLETE open branch with an unsatisfiable root

A Common Mistake to Avoid!!!!

- You are very liable to forget to COMPLETE YOUR open branch!!!
- You only have a counterexample to completeness if you have a COMPLETE open branch with an unsatisfiable root
- So make sure you FULLY RESOLVE every sentence on the open branch, until you can write that coveted up arrow '$\uparrow$'!

A Common Mistake to Avoid!!!!

- You are very liable to forget to COMPLETE YOUR open branch!!!
- You only have a counterexample to completeness if you have a COMPLETE open branch with an unsatisfiable root
- So make sure you FULLY RESOLVE every sentence on the open branch, until you can write that coveted up arrow '$\uparrow$'!
- When in doubt, just complete the whole tree

Conservative Biconditional (bottoms up!)

- Reason from the bottom up: there is only one child node. So we only need its sentences to entail $\Phi \equiv \Psi$

Conservative Biconditional (bottoms up!)

- Reason from the bottom up: there is only one child node. So we only need its sentences to entail $\Phi \equiv \Psi$
- Sentences on bottom are equivalent to $(\Phi \,\&\, \Psi)$, which DOES entail $\Phi \equiv \Psi$. So we proceed to formally extend our completeness proof:

Conservative Biconditional (bottoms up!)

- Reason from the bottom up: there is only one child node. So we only need its sentences to entail $\Phi \equiv \Psi$
- Sentences on bottom are equivalent to $(\Phi \& \Psi)$, which DOES entail $\Phi \equiv \Psi$. So we proceed to formally extend our completeness proof:
  - **Formally**: since only one child node, both Φ and Ψ lie on the complete open branch O.

Conservative Biconditional (bottoms up!)

- Reason from the bottom up: there is only one child node. So we only need its sentences to entail $\Phi \equiv \Psi$
- Sentences on bottom are equivalent to $(\Phi \& \Psi)$, which DOES entail $\Phi \equiv \Psi$. So we proceed to formally extend our completeness proof:
  - **Formally**: since only one child node, both Φ and Ψ lie on the complete open branch O.
  - By induction hypothesis, both are true according to $\mathcal{I}$.

Conservative Biconditional (bottoms up!)

- Reason from the bottom up: there is only one child node. So we only need its sentences to entail $\Phi \equiv \Psi$
- Sentences on bottom are equivalent to $(\Phi \& \Psi)$, which DOES entail $\Phi \equiv \Psi$. So we proceed to formally extend our completeness proof:
  - **Formally**: since only one child node, both Φ and Ψ lie on the complete open branch O.
  - By induction hypothesis, both are true according to $\mathcal{I}$.
  - Hence, $\Phi \equiv \Psi$ is true on $\mathcal{I}$, which is what we needed to show.

Some Fun Questions to Ponder

- What is the least number of tree rules required for soundness? For completeness?

Some Fun Questions to Ponder

- What is the least number of tree rules required for soundness? For completeness?
  - We’ll return to this question if we ever get around to discussing the “expressive adequacy” of a given set of connectives

Some Fun Questions to Ponder

- What is the least number of tree rules required for soundness? For completeness?
  - We'll return to this question if we ever get around to discussing the "expressive adequacy" of a given set of connectives

- When assessing how a modification affects soundness, do we need to know anything about the other rules?

Some Fun Questions to Ponder

- What is the least number of tree rules required for soundness? For completeness?
  - We'll return to this question if we ever get around to discussing the "expressive adequacy" of a given set of connectives
- When assessing how a modification affects soundness, do we need to know anything about the other rules?
- When assessing how a modification affects completeness, do we need to know anything about the other rules?