Truth Trees for Predicate Logic

Computability and Logic
Running Examples

*Valid Argument* (13.24):
\[ \exists x \ (\text{Cube}(x) \land \text{Small}(x)) \]
\[ \therefore \exists x \ \text{Cube}(x) \land \exists x \ \text{Small}(x) \]

*Invalid Argument* (13.25):
\[ \exists x \ \text{Cube}(x) \land \exists x \ \text{Small}(x) \]
\[ \therefore \exists x \ (\text{Cube}(x) \land \text{Small}(x)) \]
Truth-Functional Expansions

• Suppose that our Universe of Discourse (UD) contains only the objects a and b.

• Given this UD, the claim $\forall x \text{ Cube}(x)$ is true iff $\text{Cube}(a) \land \text{Cube}(b)$ is true.

• Similarly, the claim $\exists x \text{ Cube}(x)$ is true iff $\text{Cube}(a) \lor \text{Cube}(b)$ is true.

• The truth-functional interpretation of the FO statements given a fixed UD is called the *truth-functional expansion* of the original FO statement with regard to that UD.
Truth-Functional Expansions and Proving FO Invalidity

- Truth-Functional expansions can be used to prove FO invalidity. Example (13.25):

\[ \exists x \text{ Cube}(x) \land \exists x \text{ Small}(x) \]

[UD = \{a,b\}]

\[ \therefore \exists x (\text{Cube}(x) \land \text{Small}(x)) \]

\[
\begin{array}{cccccccccc}
T & T & F & T & F & T & T & T & T & T \\
\end{array}
\]

\[
(Cube(a) \lor Cube(b)) \land (Small(a) \lor Small(b))
\]

\[ \therefore (Cube(a) \land Small(a)) \lor (Cube(b) \land Small(b)) \]

\[
\begin{array}{cccccccc}
T & F & F & F & F & F & F & F & F & T \\
\end{array}
\]

This shows that there is a world in which the premise is true and the conclusion false. Hence, the original argument is FO invalid.
Truth-Functional Expansions and Proving FO Validity

• If the truth-functional expansion of an FO argument in some UD is truth-functionally invalid, then the original argument is FO invalid, but if it is truth-functionally valid, then that does not mean that the original argument is FO valid.

• For example, with UD = \{a\}, the expansion of the argument would be truth-functionally valid. In general, it is always possible that adding one more object to the UD makes the expansion invalid.

• Thus, we can’t prove validity using the expansion method, as we would have to show the expansion to be valid in every possible UD, and there are infinitely many UD’s.

• The expansion method is therefore only good for proving invalidity. Indeed, it searches for countermodels.
The Expansion Method as a Systematic Procedure

• Still, the expansion method can be made into a systematic procedure to test for FO invalidity:
  – Step 1: Expand FO argument (which can be done systematically) in UD = \{a\}.
  – Step 2: Use some systematic procedure (e.g. truth-table method or truth-tree method) to test whether the expansion is TF invalid. If it is TF invalid, then stop: the FO argument is FO invalid. Otherwise, expand FO argument in UD = \{a,b\}, and repeat step 2.
Incompleteness of the Expansion Method

• We saw that the expansion method is not a complete test for FO validity.
• However, it is also an incomplete test for FO invalidity!
• Proof: Consider the following argument:

\[ \forall x \forall y (x \neq y \rightarrow ((x > y \lor y > x) \land \neg (x > y \land y > x))) \]
\[ \forall x \forall y \forall z ((x > y \land y > z) \rightarrow x > z) \]
\[ \therefore \exists x \forall y (x \neq y \rightarrow x > y) \]

For any UD with an arbitrarily large yet finite number of objects, the expansion of this argument will be truth-functionally valid. However, the argument is FO invalid (consider the natural numbers)!
A More Focused Search

• A further drawback of the expansion method is that the search for a counterexample is very inefficient.

• A focused search for a counterexample is more efficient:
  – (13.25) I want there to be at least one cube, and at least one small object, but no small cubes. So, if we have a cube, a, then a cannot be small, so I need a second object, b, which is small, but not a cube. Counterexample, so the argument is invalid.
Advantage of a Focused Search

• The focused search method is like the indirect truth-table method.

• Indeed, like the indirect truth-table method, the focused search method can prove validity:
  – (13.24) I want there to be at least one small cube. Let us call this small cube $a$. How, I don’t want it to be true that there is at least one cube and at least one small object. However, $a$ is both a cube and small. Contradiction, so I can’t generate a counterexample.
Truth-Trees for Predicate Logic

• Like the direct method, the focused search method needs to be systematized, especially since the search often involves making choices.

• Fortunately, the truth-tree method, which systematized the indirect truth-table method in truth-functional logic, can be extended for predicate logic.
Truth-Tree Rules for Quantifiers

\[ \neg \forall x \varphi(x) \quad \sqrt{\ } \quad \neg \exists x \varphi(x) \quad \sqrt{\ } \]
\[ \exists x \neg \varphi(x) \quad \forall x \neg \varphi(x) \]

\[ \exists x \varphi(x) \quad \sqrt{\ } \quad \forall x \varphi(x) \]
\[ \varphi(c) \quad \varphi(c) \]

with ‘c’ a new constant in that branch
with ‘c’ any constant
Truth-Tree Example I

\( \exists x \text{ Cube}(x) \land \exists x \text{ Small}(x) \quad \checkmark \)

\( \neg \exists x (\text{Cube}(x) \land \text{Small}(x)) \quad \checkmark \)

\( \exists x \text{ Cube}(x) \quad \checkmark \)

\( \exists x \text{ Small}(x) \quad \checkmark \)

\( \forall x \neg (\text{Cube}(x) \land \text{Small}(x)) \)

\( \text{Cube}(a) \)

\( \text{Small}(b) \)

\( \neg (\text{Cube}(a) \land \text{Small}(a)) \quad \checkmark \)

\( \neg (\text{Cube}(b) \land \text{Small}(b)) \quad \checkmark \)

\( \neg \text{Cube}(a) \quad \times \quad \neg \text{Small}(a) \)

Open branch, so it’s invalid

\( \neg \text{Cube}(b) \quad \neg \text{Small}(b) \quad \times \)
Truth-Tree Example II

\[ \exists x (\text{Cube}(x) \land \text{Small}(x)) \quad \checkmark \]
\[ \neg (\exists x \text{Cube}(x) \land \exists x \text{Small}(x)) \quad \checkmark \]
\[ \text{Cube}(a) \land \text{Small}(a) \quad \checkmark \]
\[ \text{Cube}(a) \]
\[ \text{Small}(a) \]
\[ \neg \exists x \text{Cube}(x) \quad \checkmark \]
\[ \neg \exists x \text{Small}(x) \quad \checkmark \]
\[ \forall x \neg \text{Cube}(x) \quad \checkmark \]
\[ \forall x \neg \text{Small}(x) \]
\[ \neg \text{Cube}(a) \]
\[ \neg \text{Small}(a) \]

All branches close, so it’s valid.
Finished Trees

• A branch is closed if it contains a statement and its negation.
• An open branch is finished if every statements in that branch that has not been decomposed is either a literal or a universal that has been instantiated for every constant in that branch.
• A tree is finished if all its branches are closed (in which case the statements at the root cannot be satisfied), or if it contains a finished open branch (in which case the statements can be satisfied).
Infinite Trees

\[ \forall x \ \exists y \ \text{Likes}(x,y) \]
\[ \exists y \ \text{Likes}(a,y) \quad \checkmark \]
\[ \text{Likes}(a,b) \]
\[ \exists y \ \text{Likes}(b,y) \quad \checkmark \]
\[ \text{Likes}(b,c) \]
\[ \exists y \ \text{Likes}(c,y) \quad \checkmark \]
\[ \text{Likes}(c,d) \]
\[ \exists y \ \text{Likes}(d,y) \quad \checkmark \]
\[ \text{Likes}(d,e) \]
\[ \vdots \]

This tree will never be finished, so the tree method will not give us any answer!