We will extend the truth tree technique from PL to QL

- recap truth tree ideas
- recap QL semantics
- introduce additional tree development rules for QL
- identify when a branch is complete and open
- show how to read an interpretation from such a branch

### QL Semantics Recap

Recall that a QL interpretation consists of

- a **domain**, which is the set of things our formulae are talking about
- an individual from the domain for each name (a, b, ...)
- a subset of the domain for each predicate letter (F, G, ...)
- a set of pairs for each binary (dyadic) relation letter (R, ...)
- a set of triples for each ternary (triadic) relation letter ...

### Truth Trees

The truth-tree technique takes a set of formulae, and step by step, analyses their truth-conditions until we either find an interpretation that makes all the formulae true, or know that no such interpretation exists.

At each step, we choose a formula and ask ourselves “what would make this true?”

For example

- to make $A \& B$ true, we need $A$ to be true, and $B$ to be true
- to make $A \rightarrow B$ true, we need either $A$ to be false, or $B$ to be true

### Validity and Satisfiability

If there is a way of making all the input formulae true, then the tree test will find one. How can we use this to determine whether a sequent is **valid**?

A sequent is **invalid** iff there is an interpretation which makes the premises true and the conclusion false. So, we input the premises and the negated conclusion. If there is no interpretation to satisfy these input formulae (the tree closes) then the sequent is valid.
Truth Trees for QL

- The truth-tree development rules for PL also apply for QL.
- We need new rules for the quantifiers.
- There is a bit more work to read the interpretation from an open completed QL branch.

Reading an Interpretation from a Branch

How can we make all these formulae true?

\[ Fa, Fb, Gb \]

That is, find a domain, individuals for each name, and extensions for each of the predicates and relations, such that these three formulae are true. We use numbers as individuals, and give them the properties the formulae assign them.

<table>
<thead>
<tr>
<th>domain: {1, 2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>a: 1</td>
</tr>
<tr>
<td>b: 2</td>
</tr>
<tr>
<td>F: {1, 2}</td>
</tr>
<tr>
<td>G: {2}</td>
</tr>
</tbody>
</table>

Truth Tree Rules for QL - Universal

From the previous example, we can see that it is enough to break the formulae down into "atomic" formulae. That is predicate and relation symbols applied to the correct number of names. The QL truth-tree rules allow us to do this.

What would make \( \forall x [\phi(x)] \) true? Since we only put individuals in our domain when we have a name to interpret, it is enough to have \( \phi(\rho) \) for every name \( \rho \) in the branch. In the previous example, the interpretation makes \( \forall x [Fx] \) true, because we have \( Fa \) and \( Fb \), and \( a \) and \( b \) are the only names on the branch.

Example: \( Fa, \forall x [Gx] \)

Truth Tree Rules for QL - Existential

What would make \( \exists x [\phi(x)] \) true? If we put \( \phi(\rho) \) on the branch, for some name \( \rho \), that will make it true. But be careful!

We must take care to keep all possibilities for making the input formula true as we develop the tree.

Examples

- \( Fa, \exists x [Gx] \)
- \( Fa, \exists x [Gx], \sim Ga \)

If we use a name that is already on the branch to satisfy an existential, we risk incorrectly closing that branch. So always use a name that does not yet appear on the branch when developing an existential. New names for existentials!
Negated Quantifiers

Consider these two statements and their QL formalisations:

- It is not the case that everybody loves Alice. \( \neg \forall x [L_x a] \)
- There is somebody who does not love Alice. \( \exists x [\neg L_x a] \)

They are logically equivalent, in any situation they will have the same truth value.

Similarly for \( \neg \exists x [L_x a] \) and \( \forall x [\neg L_x a] \).

Equality

What do we do when the branch contains \( a = b \)?

**Development:** if \( \phi(a) \) is on a branch, we may add \( \phi(b) \), and vice versa. We do this if it can help us close a branch.

**Reading an interpretation:** we make \( a \) and \( b \) different names for the same individual.

- **domain:** \{\ldots, 42, \ldots\}
- a: 42
- b: 42

An example

Let’s study an (invalid) argument

- \( \forall x [C_x] \) everybody likes chocolate
- \( \exists x [S_x \lor P_x] \) somebody likes strawberries or peaches
- \( \exists x [C_x \land P_x] \) therefore, somebody likes chocolate and strawberries

Exercises

<table>
<thead>
<tr>
<th>Homework Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Please do more than these, until you are confident you know what you are doing. But hand in those listed below.</td>
</tr>
<tr>
<td>- Page 167, 4.6 1</td>
</tr>
<tr>
<td>- Page 176, 4.7 4</td>
</tr>
<tr>
<td>- Page 346, 7.1 7</td>
</tr>
<tr>
<td>- Page 372, 7.3 16</td>
</tr>
</tbody>
</table>

**Please note:** I may not be able to mark the test before Tuesday’s lesson, in which case, I will show you the marked papers on Thursday.