Infinite Complete Binary Trees

by Sven Nilsen, 2019

In this paper I formalize infinite complete binary trees in path semantics.

Building on previous definition of rooted binary trees, an infinite complete binary tree is the following:

∃left <=> branch / root
∃right <=> branch / root

branch : binary_tree → bool
root : binary_tree → bool
left : branch → binary_tree
right : branch → binary_tree

Here, `branch / root` means `branch` except `root`.

To explain what this means, one can assume some function `infinite` exists such that:

∃left <=> infinite / root

infinite : binary_tree → bool

Since `left` takes a `branch`, there must exist one branch for every infinite binary tree.
However, this does not say how many infinite branches there are.
There could be more than one branch mapping to the same infinite binary tree.
A such binary tree is not complete, because it does not cover the set of countable infinity.
If each node were represented as a natural number, then all natural numbers would not be covered.

By adding the following statement:

infinite <=> branch

This means that if a binary tree is infinite, then it is also a branch and vice versa.

From previous results, is known that `root` is a `branch`, so here a root must be `infinite` too.
It is also known that `root` can not be returned from `left` or `right`.
One can use this to say the following about `left` and `right`:

∃left <=> branch / root
∃right <=> branch / root

If a such binary tree is rooted, then there exists some input that is not returned by `left` and `right`.
If a such binary tree is not rooted, then all branches have a parent and are also infinite.
In the unrooted case, `left` and `right` maps one-by-one between branches and therefore covers up to set of countable infinity or more. In the rooted case, it covers the set up to countable infinity, no more.