Infibin

by Sven Nilsen, 2019

In this paper I explain the semantics of an encoding of infinite complete rooted binary trees using natural numbers. I suggest calling this an "infibin" to easier refer to it as a mathematical object.

Previously, I formalized full binary trees, rooted full binary trees and infinite complete binary trees. Here I will talk about an encoding of infinite complete rooted binary trees using natural numbers:

```
left(n) = 2 · n + 1
right(n) = 2 · n + 2
parent(n) = if n == 0 { 0 } else if even(n) { (n – 2) / 2 } else { (n – 1) / 2 }

left : nat → nat
right : nat → nat
parent : nat → nat
even : nat → bool
```

I call this an "infibin" to easier refer this particular mathematical object, instead of talking about it in terms of the more general semantics of full binary trees.

What makes infibin special is that the decidability of every rooted, infinite and complete binary tree can be mapped to an infibin. So, an infibin kind of represents this class of binary trees while at the same time being a very concrete mathematical object.

Formally, for every rooted, infinite and complete binary tree, there exists a symmetric path by `g`:

$$\text{left}_X[g] <=> \text{left}$$
$$\text{right}_X[g] <=> \text{right}$$
$$\text{parent}_X[g] <=> \text{parent}$$

$$g : X \to \text{nat}$$

One might choose to write this in the following notation:

```
infinite_complete_binary_tree[g] <=> infibin

infinite_complete_binary_tree := (left_X, right_X, parent_X)
infibin := (left, right, parent)
g : X → nat
```

The function `g` might be non-trivial, e.g. mapping segments of an infibin to an infibin. A such solution exists if grand-children have depth-dependence in Cartesian products, such that every grand-parent has a children product type of their own type as grand-children. The root has two children, then each children has four (two in each left and right branch), each grand-children has eight (four in each left and right branch), great-grand-children sixteen (eight in each left and right branch), and so on.