Serebin Path Tree

by Sven Nilsen, 2018

In this paper I represent a new data structure called "Serebin Path Tree" and provide a notation for indices. The advantage of this notation is the ability to describe "for every path" in expressions that does not depend on the sibling but on some parent, while allowing index inference.
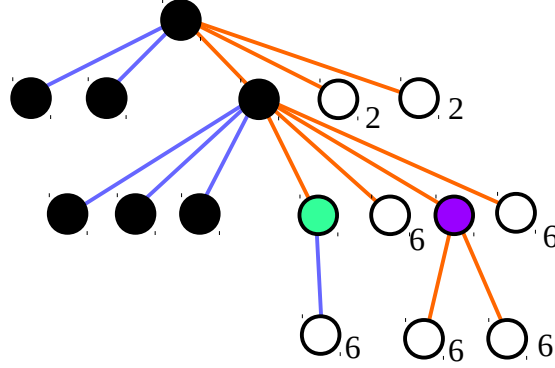

Illustration of a Serebin Path Tree. A blue-linked sub-node has same orange-linked children as the parent.
Data is stored in every possible path that can be taken through blue links or purple nodes to a white node.
Numbers show possible paths for every white node. The sum is the size of the data structure.

Node colors	Has blue-linked children	Has no blue-linked children
Has orange-linked children	black node ●	purple node 🟣
Has no orange-linked children	turqoise node 🟢	white node ○

A Path Tree stores data for every possible path to the leaves. Serebin Tree is a Path Tree where blue-linked nodes share orange-linked children with their parent node.

The indices of a Serebin can be written using superscript and subscript, or in plain text format:

$$A^\wedge x = A^x \qquad \forall\, x\, \{\, orange\_children(A^x) = orange\_children(A)\, \}$$
$$A\_x = A_x \qquad \neg\exists\, \{\, orange\_children(A_x) = orange\_children(A)\, \}$$

Inference rules (satisfies Boolean algebra with index inference and custom path semantical notation):

$$A^x_y \cap A_y{}^z = A^x_y{}^z \qquad A^x \cup A_y = A \qquad A[{}^{g0}{}_{g1} \to g_n] <=> A[G_{i \to n}] <=> B \;:\text{serebin}$$

$$\cap[\neg] <=> \cup \qquad \cup[\neg] <=> \cap \qquad \cap, \cup : A \times A \to A \qquad \neg : A \to A$$
$$A : \forall\, {}^x_y{}^z\ldots \{\, (() \to T)\, \} \wedge ({}^{nat}_{nat}{}^{nat}\ldots \to (\subseteq A)) \qquad \forall A : {}^{nat}_{nat}{}^{nat}\ldots \to bool \qquad \exists A : T \to bool$$
$$A^\forall : nat \to bool \quad A^{-}_\forall : nat \to bool \quad A^\forall_\forall : nat \times nat \to bool \quad \forall[A] : [nat \to bool] \quad A[[x, y, z]] = A^x_y{}^z$$

The type is `serebin[T]`. The primary purpose of a **Se**lf-**Re**ferential **Bi**nary **N**-Tree (**Serebin** Tree) is to write expressions where indices are inferred. It is also a superset of common data structures such as multi-dimensional arrays or trees. This allows more efficient reasoning about some complex problems. A Serebin Tree occurs in Symbolent Calculus, an extension of Sequent Calculus, using `Γ`.

When a blue link is omitted, it means that either the node contains no blue-linked sub-nodes, or the expression is meant to refer to "for all paths" through all blue-linked sub-nodes.

For example:

$$X^{0\ 5}_{3\ 6\ 2} \qquad X\wedge 0\_3\wedge 5\_6\_2$$

When two blue links are in a row, it means that an orange link must have been omitted. This is because a blue-linked sub-node shares orange-linked children with its parent, but no blue-linked sub-nodes. This means "for all paths" through the orange-linked sub-nodes.

$$X^{2\ 2}$$

All indices are natural numbers, so negative numbers are not indices.
When a `-` sign is used in front of a number, it means omitting the index explicitly:

$$X^{-} = X$$
$$X_{-} = X$$

$$X^{-\ -}_{\ \ 2} = X^{-}_{-2}$$
$$X^{-\ 3}_{-} = X^{\ 3}_{-} = X^{-3}$$
$$X^{-}_{\ 2} = X_{2}$$

Two `-` in a row can not be erased if is an index later, no matter how they are arranged.

When a Serebin Tree contains a unique path, one can use it as a function:

$$X : serebin[T] \wedge [\|\|]\ 1 \implies \qquad X^{x}_{y}() : T$$

When the Serebin Tree called is not unique, it is inferred that the expression depends on the indices:

$$X : serebin[nat] \qquad \implies \qquad (X - X)() \iff (X^{a\ c}_{b\ d}... - X^{a\ c}_{b\ d}...)() \iff (0 : \{^{nat}_{nat}{}^{nat}_{nat}...\} \to nat)$$

This makes it possible to manipulate expressions symbolically that describes many functions.

A common short version used in Symbolent Calculus is the following:

$$(\Gamma : serebin[bool]) \wedge (a : [\Gamma()]\ true) \qquad \iff \qquad a : \Gamma$$

Serebin Trees supports some function currying, which cuts off branches and returns a new tree:

$$A^{x\ z}_{y} = (A^{x}_{y})^{z} = (A^{x})^{z}_{y} = (A)^{x\ z}_{y} = (A^{\ z}_{y})^{x} = (A^{x\ z})_{y}$$

However, at deeper levels than 3 indices, you need to omit or contract explicitly:

$$A^{x\ z}_{y\ w} = (A^{x\ z}_{y})^{-\ w} = contract(A^{x\ z}_{y})^{w} \qquad contract\_map : serebin[T] \to [([nat], [nat])]$$
$$contract : (A : serebin[T]) \to$$
$$\qquad (X : serebin[T] \wedge [contract]\ (= X) \wedge \forall\ (i, j) : contract\_map(A)\ \{\ A[i] \iff X[j]\ \})$$