0.0.1 Append At

The Primitive *appendAt* uses the Primitive *atDepth* to navigated into a nested collection *coll?* (called *coll* bellow). The Value $v?$ passed to *appendAt* will be appended to *coll* at $idxs?_j$. This results in a *coll!* which is equivalent to *coll?* except for at the value at the path $idxs?_i \mathrel{..} idxs?_j \in coll?$.

$$
\begin{array}{|l}
\hline
AppendAt[Collection, Collection, V] \\
AtDepth \\
coll?, coll!, idxs? : Collection \\
v? : V \\
appendAt\_ : Collection \times Collection \times V \rightarrowtail Collection \\
\hline
appendAt = \langle atDepth\_, append\_, \langle atDepth\_, remove\_, append\_\rangle^{\# idxs? - 1}\rangle \\
coll! = appendAt(coll?, idxs?, v?) \bullet \\
\quad let \quad coll == append(atDepth(coll?, idxs? \lhd idxs?_j), v?, idxs?_j) \bullet \\
\qquad \forall n : i \mathrel{..} j - 1 \bullet j = first(last(idxs?)) \mid \exists c_n \bullet \\
\qquad\quad let \quad c_i == atDepth(coll?, (idxs? \upharpoonright i)) \Rightarrow atIndex(coll?, idxs?_i) \\
\qquad\qquad c_n == atDepth(c_{n-1}, (idxs? \upharpoonright n)) \\
\qquad\qquad c_{j-1} == atDepth(c_n, (idxs? \upharpoonright j - 1)) \iff n = j - 2 \\
\qquad\qquad c_j == append(c_{j-1}, v?, (idxs? \upharpoonright j)) \Rightarrow c_j = coll = coll!_j \\
\qquad\qquad coll!_{j-1} == append(remove(c_{j-2}, idxs?_{j-1}), c_j, idxs?_{j-1}) \\
\qquad\qquad coll!_n == append(remove(c_{n-1}, idxs?_n), coll!_n, idxs?_n) \\
\qquad\qquad coll!_i == append(remove(c_i, idxs?_n), coll!_n, idxs?_n) \iff n = i + 1 \\
\quad = append(remove(coll?, idxs?_i), coll!_i, idxs?_i) \\
\hline
\end{array}
$$

The relationship described above $coll? \lhd idxs_i = coll! \lhd idxs_i$ is described above as $\langle atDepth\_, remove\_, append\_\rangle^{\# idxs? - 1}$. The variables $coll!_{i \mathrel{..} j}$ were used to describe the sub Collections which have to have a single index updated given *idxs?*. Those subcollections are combined together to produce *coll!* such that the only difference between $coll? \land coll!$ is found at path *idxs?* The following examples demonstrate the properties of *appendAt* described above.

$$
\begin{array}{ll}
X = \langle x_0, x_1, x_2\rangle & \\
\quad x_0 = 0 & \\
\quad x_1 = foo & \\
\quad x_2 = \langle a, b, c\rangle & \\
appendAt(X, \langle 1, 3\rangle, z) = \langle x_0, fooz, x_2\rangle \Rightarrow foo = \langle f, o, o\rangle & \\
appendAt(X, \langle 1\rangle, 5) = \langle\langle 0, 5\rangle, x_1, x_2\rangle & \text{[existing item gets 0 index]} \\
appendAt(X, \langle 1, 0\rangle, 5) = \langle\langle 5, 0\rangle, x_1, x_2\rangle & \text{[overwritting default behavior]} \\
appendAt(X, \langle 2, 0\rangle, d) = \langle x_0, x_1, \langle d, a, b, c\rangle\rangle & \\
appendAt(X, \langle 2\rangle, d) = \langle x_0, x_1, \langle a, b, c, d\rangle\rangle &
\end{array}
$$