0.0.1 Map

The *map* operation takes in an Operation O or Primitive P, Collection *coll*? and additional Arguments *args*? and returns a modified Collection *coll*! where each member $coll!_i$ is the result of passing $coll?_i$ and *args*? to $P \lor O$. The ordering of *coll*? is maintained within *coll*!

$$
\begin{array}{|l}
map[O \lor P, Collection, V] \\
\hline
operation? : O \\
primitive? : P \\
args? : V \\
coll? , coll! : Collection \\
map : (O \lor P \times Collection \times V \to Collection) \lor (O \lor P \times Collection \to Collection) \\
\hline
coll! = map(operation? , coll?) \bullet \\
\qquad \langle \forall i : 0 .. j \in coll? \mid \\
\qquad operation? (coll?_0) \frown operation? (coll?_i) \frown operation? (coll?_j)\rangle \\
coll! = map(operation? , coll? , args?) \bullet \\
\qquad \langle \forall i : 0 .. j \in coll? \mid \\
\qquad operation? (coll?_0 , args?) \frown operation? (coll?_i , args?) \frown operation? (coll?_j , args?)\rangle \\
coll! = map(primitive? , coll?) \bullet \\
\qquad \langle \forall i : 0 .. j \in coll? \mid \\
\qquad primitive? (coll?_0) \frown primitive? (coll?_i) \frown primitive? (coll?_j)\rangle \\
coll! = map(primitive? , coll? , args?) \bullet \\
\qquad \langle \forall i : 0 .. j \in coll? \mid \\
\qquad primitive? (coll?_0 , args?) \frown primitive? (coll?_i , args?) \frown primitive? (coll?_j , args?)\rangle \\
\hline
\end{array}
$$

$X = \langle 1, 2, 3\rangle$

$map\,(succ, X) = \langle 2, 3, 4\rangle$ [increment each member of X]

$map\,(+, X, 2) = \langle 3, 4, 5\rangle$ [add 2 to each member of X]