1 ───────────── MODULE OT ─────────────

Specification of OT (Operational Transformation) functions. It consists of the basic OT functions for two operations and more general ones involving operation sequences.

7 EXTENDS $OpOperators$, $SetUtils$

8 ├──

OT (Operational Transformation) functions.

Naming convention: I for "Ins" and D for "Del".

The left "Ins" $lins$ transformed against the right "Ins" $rins$.

18 $XformII(lins,\ rins) \triangleq$
19 IF $lins.pos < rins.pos$
20 THEN $lins$
21 ELSE IF $lins.pos > rins.pos$
22 THEN $[lins$ EXCEPT $!.pos = @ + 1]$
23 ELSE IF $lins.ch = rins.ch$
24 THEN Nop
25 ELSE IF $lins.pr > rins.pr$
26 THEN $[lins$ EXCEPT $!.pos = @ + 1]$
27 ELSE $lins$

The left "Ins" ins transformed against the right "Del" del.

31 $XformID(ins,\ del) \triangleq$
32 IF $ins.pos \leq del.pos$
33 THEN ins
34 ELSE $[ins$ EXCEPT $!.pos = @ - 1]$

The left "Del" del transformed against the right "Ins" ins.

38 $XformDI(del,\ ins) \triangleq$
39 IF $del.pos < ins.pos$
40 THEN del
41 ELSE $[del$ EXCEPT $!.pos = @ + 1]$

The left "Del" $ldel$ transformed against the right "Del" $rdel$.

45 $XformDD(ldel,\ rdel) \triangleq$
46 IF $ldel.pos < rdel.pos$
47 THEN $ldel$
48 ELSE IF $ldel.pos > rdel.pos$
49 THEN $[ldel$ EXCEPT $!.pos = @ - 1]$
50 ELSE Nop

51 ├──

Transform the left operation lop against the right operation rop with appropriate OT function.

56 $Xform(lop,\ rop) \triangleq$
57 CASE $lop = Nop \lor rop = Nop \rightarrow lop$
58 $\square\ lop.type =$ "Ins" $\land rop.type =$ "Ins" $\rightarrow XformII(lop,\ rop)$
59 $\square\ lop.type =$ "Ins" $\land rop.type =$ "Del" $\rightarrow XformID(lop,\ rop)$
60 $\square\ lop.type =$ "Del" $\land rop.type =$ "Ins" $\rightarrow XformDI(lop,\ rop)$

1

61       $\square$ $lop.type =$ "Del" $\land$ $rop.type =$ "Del" $\rightarrow XformDD(lop, rop)$

62 $\vdash$───

Generalized OT functions on operation sequences.

Iteratively/recursively transforms the operation op against an operation sequence ops.

71 RECURSIVE $XformOpOps(\_, \_, \_)$
72 $XformOpOps(xform(\_, \_),\ op,\ ops)\ \triangleq$
73   IF $ops = \langle\rangle$
74   THEN op
75   ELSE $XformOpOps(xform,\ xform(op,\ Head(ops)),\ Tail(ops))$

Iteratively/recursively transforms the operation op against an operation sequence ops. Being different from $XformOpOps$, $XformOpOpsX$ maintains the intermediate transformed operation

82 RECURSIVE $XformOpOpsX(\_, \_, \_)$
83 $XformOpOpsX(xform(\_, \_),\ op,\ ops)\ \triangleq$
84   IF $ops = \langle\rangle$
85   THEN $\langle op\rangle$
86   ELSE $\langle op\rangle \circ XformOpOpsX(xform,\ xform(op,\ Head(ops)),\ Tail(ops))$

Iteratively/recursively transforms the operation sequence ops against an operation op.

91 $XformOpsOp(xform(\_, \_),\ ops,\ op)\ \triangleq$
92   LET $opX\ \triangleq\ XformOpOpsX(xform,\ op,\ ops)$
93   IN $[i \in 1\ ..\ Len(ops) \mapsto xform(ops[i],\ opX[i])]$

Iteratively/recursively transforms an operation sequence $ops1$ against another operation sequence $ops2$.

See also Definition 2.13 of the paper "Imine @ TCS06".

100 RECURSIVE $XformOpsOps(\_, \_, \_)$
101 $XformOpsOps(xform(\_, \_),\ ops1,\ ops2)\ \triangleq$
102   IF $ops2 = \langle\rangle$
103   THEN $ops1$
104   ELSE $XformOpsOps(xform,\ XformOpsOp(xform,\ ops1,\ Head(ops2)),\ Tail(ops2))$

105 $\vdash$───

\ * Modification History
\ * Last modified *Fri Dec* 28 14:58:58 *CST* 2018 by *hengxin*
\ * Created Sun *Jun* 24 15:57:48 *CST* 2018 by *hengxin*