```
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, AdditionalSetOperators

9  Nop ≜ PickNone(Nat)
10 ------------------------------
   OT (Operational Transformation) functions.

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

   The left "Ins" lins transformed against the right "Ins" rins.
20 XformII(lins, rins) ≜
21     IF lins.pos < rins.pos
22      THEN lins
23      ELSE IF lins.pos > rins.pos
24               THEN [lins EXCEPT !.pos = @ + 1]
25               ELSE IF lins.ch = rins.ch
26                        THEN Nop
27                        ELSE IF lins.pr > rins.pr
28                                 THEN [lins EXCEPT !.pos = @ + 1]
29                                 ELSE lins

   The left "Ins" ins transformed against the right "Del" del.
34 XformID(ins, del) ≜
35     IF ins.pos ≤ del.pos
36      THEN ins
37      ELSE [ins EXCEPT !.pos = @ − 1]

   The left "Del" del transformed against the right "Ins" ins.
42 XformDI(del, ins) ≜
43     IF del.pos < ins.pos
44      THEN del
45      ELSE [del EXCEPT !.pos = @ + 1]

   The left "Del" ldel transformed against the right "Del" rdel.
50 XformDD(ldel, rdel) ≜
51     IF ldel.pos < rdel.pos
52      THEN ldel
53      ELSE IF ldel.pos > rdel.pos
54               THEN [ldel EXCEPT !.pos = @ − 1]
55               ELSE Nop
56 ------------------------------

   Transform the left operation lop against the right operation rop with appropriate OT function.
61 Xform(lop, rop) ≜
62     CASE lop = Nop ∨ rop = Nop → lop
```

```
        □ lop.type = "Ins" ∧ rop.type = "Ins" → XformII(lop, rop)
        □ lop.type = "Ins" ∧ rop.type = "Del" → XformID(lop, rop)
        □ lop.type = "Del" ∧ rop.type = "Ins" → XformDI(lop, rop)
        □ lop.type = "Del" ∧ rop.type = "Del" → XformDD(lop, rop)
```

Generalized *OT* functions on operation sequences.

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

```
RECURSIVE XformOpOps(_, _)
XformOpOps(op, ops) ≜
    IF ops = ⟨⟩
        THEN op
        ELSE XformOpOps(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

```
RECURSIVE XformOpOpsX(_, _)
XformOpOpsX(op, ops) ≜
    IF ops = ⟨⟩
        THEN ⟨op⟩
        ELSE ⟨op⟩ ∘ XformOpOpsX(Xform(op, Head(ops)), Tail(ops))
```

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

```
XformOpsOp(ops, op) ≜
    LET opX ≜ XformOpOpsX(op, ops)
    IN  [i ∈ 1 .. Len(ops) ↦ Xform(ops[i], opX[i])]
```

Iteratively/recursively transforms an operation sequence *ops*1 against another operation sequence *ops*2.

See also Definition 2.13 of the paper "Imine @ *TCS*06".

```
RECURSIVE XformOpsOps(_, _)
XformOpsOps(ops1, ops2) ≜
    IF ops2 = ⟨⟩
     THEN ops1
     ELSE XformOpsOps(XformOpsOp(ops1, Head(ops2)), Tail(ops2))
```

* Modification History
* *Last* modified *Tue Aug* 28 15:21:56 *CST* 2018 by *hengxin*
* Created Sun *Jun* 24 15:57:48 *CST* 2018 by *hengxin*