```
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 Op, TLC

   Constants for finite/bounded model checking.
12 CONSTANTS MaxPr,    max priority
13           MaxLen    max length of list

15 ASSUME ∧ MaxPr ∈ PosInt
16        ∧ MaxLen ∈ Nat

18 ListMaxLen ≜ SeqMaxLen(Char, MaxLen)
19 ==========

   OT (Operational Transformation) functions.

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

   The left "Ins" ins transformed against the right "Ins" rins.
29 XformII(lins, rins) ≜
30     IF lins.pos < rins.pos
31      THEN lins
32      ELSE IF lins.pos > rins.pos
33               THEN [lins EXCEPT !.pos = @ + 1]
34               ELSE IF lins.ch = rins.ch
35                        THEN Nop
36                        ELSE IF lins.pr > rins.pr
37                                 THEN [lins EXCEPT !.pos = @ + 1]
38                                 ELSE lins

   The left "Ins" ins transformed against the right "Del" del.
43 XformID(ins, del) ≜
44     IF ins.pos ≤ del.pos
45      THEN ins
46      ELSE [ins EXCEPT !.pos = @ − 1]

   The left "Del" del transformed against the right "Ins" ins.
51 XformDI(del, ins) ≜
52     IF del.pos < ins.pos
53      THEN del
54      ELSE [del EXCEPT !.pos = @ + 1]

   The left "Del" ldel transformed against the right "Del" rdel.
59 XformDD(ldel, rdel) ≜
60     IF ldel.pos < rdel.pos
61      THEN ldel
```

```
62        ELSE IF ldel.pos > rdel.pos
63               THEN [ldel EXCEPT !.pos = @ − 1]
64               ELSE Nop
65 ├──────────────────────────────────────────────────────────────────────┤
```

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

```
70  Xform(lop, rop) ≜
71      CASE lop = Nop ∨ rop = Nop → lop
72         □ lop.type = "Ins" ∧ rop.type = "Ins" → XformII(lop, rop)
73         □ lop.type = "Ins" ∧ rop.type = "Del" → XformID(lop, rop)
74         □ lop.type = "Del" ∧ rop.type = "Ins" → XformDI(lop, rop)
75         □ lop.type = "Del" ∧ rop.type = "Del" → XformDD(lop, rop)
76 ├──────────────────────────────────────────────────────────────────────┤
```

Generalized OT functions on operation sequences.

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

```
85  RECURSIVE XformOpOps(_, _)
86  XformOpOps(op, ops) ≜
87      IF ops = ⟨⟩
88          THEN op
89          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

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

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

```
107 XformOpsOp(ops, op) ≜
108     LET opX ≜ XformOpOpsX(op, ops)
109     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".

```
117 RECURSIVE XformOpsOps(_, _)
118 XformOpsOps(ops1, ops2) ≜
119     IF ops2 = ⟨⟩
120      THEN ops1
121      ELSE XformOpsOps(XformOpsOp(ops1, Head(ops2)), Tail(ops2))
122 ├──────────────────────────────────────────────────────────────────────┤
```

The $CP1$ (C for Convergence) property.

TODO: refactor the generation of $op1$ and $op2$.

Legal operations with respect to a list l.

$InsOp(l) \triangleq$ Position of an insertion cannot be too large.
  $[type : \{\text{“Ins”}\},\ pos : 1\ ..\ Len(l)+1,\ ch : Char,\ pr : 1\ ..\ MaxPr]$

$DelOp(l) \triangleq$
  IF $l = \langle\rangle$
  THEN $\{\}$ Not allowed to delete elements from an empty list.
  ELSE $[type : \{\text{“Del”}\},\ pos : 1\ ..\ Len(l)]$ Position of a deletion cannot be too large.
$OpOnList(l) \triangleq InsOp(l) \cup DelOp(l)$

$CP1 \triangleq$
  $\forall\, l \in ListMaxLen :$
    $\forall\, op1 \in OpOnList(l),\ op2 \in OpOnList(l) :$
      $\wedge\ PrintT(ToString(l) \circ \text{“, ”} \circ ToString(op1) \circ \text{“, ”} \circ ToString(op2))$
      $\wedge$ Priorities of these two insertions cannot be the same.
        $\vee\ (op1.type = \text{“Ins”} \wedge op2.type = \text{“Ins”} \wedge op1.pr = op2.pr)$
          The $CP1$ itself.
        $\vee\ ApplyOps(\langle op1,\ Xform(op2,\ op1)\rangle,\ l) = ApplyOps(\langle op2,\ Xform(op1,\ op2)\rangle,\ l)$

The generalized $CP1$ (C for Convergence) property.

See also Theorem 2.14 of the paper “Imine @ *TCS*06”.

FIXME: Generate legal operation sequences.

$GCP1 \triangleq$
  $\forall\, l \in ListMaxLen,\ ops1 \in SeqMaxLen(Op, 1),\ ops2 \in SeqMaxLen(Op, 1) :$
    $\vee\ (Head(ops1).type = \text{“Del”} \vee Head(ops2).type = \text{“Del”})$
    $\vee\ ApplyOps(ops1 \circ XformOpsOps(ops2,\ ops1),\ l) =$
      $ApplyOps(ops2 \circ XformOpsOps(ops1,\ ops2),\ l)$

* Modification History
* *Last* modified Sat *Jul* 07 12:24:04 *CST* 2018 by *hengxin*
* Created Sun *Jun* 24 15:57:48 *CST* 2018 by *hengxin*