```
──────────────────── 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.

```
EXTENDS Op, TLC
───────────────────────────────────────────────────
```

OT (Operational Transformation) functions.

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

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

```
XformII(lins, rins) ≜
    IF lins.pos < rins.pos
     THEN lins
     ELSE IF lins.pos > rins.pos
            THEN [lins EXCEPT !.pos = @ + 1]
            ELSE IF lins.ch = rins.ch
                   THEN Nop
                   ELSE IF lins.pr > rins.pr
                          THEN [lins EXCEPT !.pos = @ + 1]
                          ELSE lins
```

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

```
XformID(ins, del) ≜
    IF ins.pos ≤ del.pos
     THEN ins
     ELSE [ins EXCEPT !.pos = @ − 1]
```

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

```
XformDI(del, ins) ≜
    IF del.pos < ins.pos
     THEN del
     ELSE [del EXCEPT !.pos = @ + 1]
```

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

```
XformDD(ldel, rdel) ≜
    IF ldel.pos < rdel.pos
     THEN ldel
     ELSE IF ldel.pos > rdel.pos
            THEN [ldel EXCEPT !.pos = @ − 1]
            ELSE Nop
───────────────────────────────────────────────────
```

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

```
Xform(lop, rop) ≜
    CASE lop = Nop ∨ rop = Nop → lop
       □ lop.type = "Ins" ∧ rop.type = "Ins" → XformII(lop, rop)
```

```
        $\Box\ lop.type = \text{“Ins”} \land rop.type = \text{“Del”} \rightarrow XformID(lop, rop)$
        $\Box\ lop.type = \text{“Del”} \land rop.type = \text{“Ins”} \rightarrow XformDI(lop, rop)$
        $\Box\ lop.type = \text{“Del”} \land rop.type = \text{“Del”} \rightarrow XformDD(lop, rop)$
```

Generalized OT functions on operation sequences.

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

```
RECURSIVE $XformOpOps(\_, \_)$
$XformOpOps(op, ops) \triangleq$
    IF $ops = \langle\rangle$
        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 $XformOpOpsX(\_, \_)$
$XformOpOpsX(op, ops) \triangleq$
    IF $ops = \langle\rangle$
        THEN $\langle op \rangle$
        ELSE $\langle op \rangle \circ XformOpOpsX(Xform(op, Head(ops)), Tail(ops))$
```

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

```
$XformOpsOp(ops, op) \triangleq$
    LET $opX \triangleq XformOpOpsX(op, ops)$
    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 @ $TCS06$”.

```
RECURSIVE $XformOpsOps(\_, \_)$
$XformOpsOps(ops1, ops2) \triangleq$
    IF $ops2 = \langle\rangle$
     THEN $ops1$
     ELSE $XformOpsOps(XformOpsOp(ops1, Head(ops2)), Tail(ops2))$
```

The $CP1$ (C for Convergence) property.

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

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

The generalized *CP*1 (C for Convergence) property.

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

FIXME: Generate legal operation sequences.

```
$GCP1 \triangleq$
    $\forall l \in List, ops1 \in SeqMaxLen(Op, 1), ops2 \in SeqMaxLen(Op, 1) :$
        $\lor (Head(ops1).type = \text{"Del"} \lor Head(ops2).type = \text{"Del"})$
        $\lor ApplyOps(ops1 \circ XformOpsOps(ops2, ops1), l) =$
          $ApplyOps(ops2 \circ XformOpsOps(ops1, ops2), l)$
```

* Modification History
* *Last* modified *Fri Jul* 06 16:18:49 *CST* 2018 by *hengxin*
* Created Sun *Jun* 24 15:57:48 *CST* 2018 by *hengxin*