MODULE OT

EXTENDS $Integers$, $Sequences$, $Naturals$, TLC

CONSTANTS $LIST$, CH, STR, PR

$POS \triangleq 1 \,..\, Len(LIST)$
$LEN \triangleq 1 \,..\, Len(LIST)$
$Maxnum(set) \triangleq$ CHOOSE $i \in set : \forall\, j \in set : i \geq j$ max number

$Intervals \triangleq \{\langle a,\, b\rangle \in POS \times POS : a + b \leq Maxnum(POS) + 1\}$
$NoncoIntervals \triangleq \{ints \in$ SUBSET $Intervals : \forall\, i,\, j \in ints : i[2] + i[1] \leq j[1] \vee j[2] + j[1] \leq i[1] \vee i = j\} \setminus \{\{\}\}$

RECURSIVE $SetTOSeq(\_)$
$SetTOSeq(T) \triangleq$ IF $T = \{\}$ THEN $\langle\rangle$
ELSE LET $t \triangleq$ CHOOSE $x \in T :$ TRUE
IN $\langle t\rangle \circ SetTOSeq(T \setminus \{t\})$

RECURSIVE $Seqset(\_)$
$Seqset(T) \triangleq$ IF $T = \{\}$ THEN $\{\}$
ELSE LET $t \triangleq$ CHOOSE $x \in T :$ TRUE
IN $Seqset(T \setminus \{t\}) \cup \{SetTOSeq(t)\}$
$NoncoSeq \triangleq Seqset(NoncoIntervals)$

RECURSIVE $SetSize(\_)$
$SetSize(T) \triangleq$ IF $T = \{\}$ THEN 0
ELSE LET $t \triangleq$ CHOOSE $x \in T :$ TRUE
IN $1 + SetSize(T \setminus \{t\})$
$Minint(ints) \triangleq$ CHOOSE $i \in ints : \forall\, j \in ints : i[1] \leq j[1] \setminus *$ min int

$compare(a,\, b) \triangleq$ IF $a[1] < b[1]$ THEN TRUE ELSE FALSE

$settoseq(set,\, list) \triangleq$
IF $set\,! =$ "" THEN $Append(list,\, Minint(set))$
ELSE $list$
$Durint(ints,\, pos) \triangleq$ CHOOSE $i \in ints : i[1] \leq pos \wedge pos \leq i[2]$

$IF\_Durint(ints,\, pos) \triangleq \exists\, i \in ints : i[1] < pos \wedge pos < i[2]$
RECURSIVE $SetSum(\_)$
$SetSum(T) \triangleq$ IF $T = \{\}$ THEN 0
ELSE LET $t \triangleq$ CHOOSE $x \in T :$ TRUE
IN $t[2] - t[1] + 1 + SetSum(T \setminus \{t\})$

$OP\_1\_set \triangleq [type : \{$"set"$\},\, pos : POS,\, ch : CH,\, pr : PR]$
$OP\_1\_ins \triangleq [type : \{$"ins"$\},\, pos : POS,\, ch : CH,\, pr : PR]$
$OP\_1\_del \triangleq [type : \{$"del"$\},\, pos : POS,\, pr : PR]$
$OP\_1 \triangleq OP\_1\_set \cup OP\_1\_ins \cup OP\_1\_del$

$OP\_2\_ins \triangleq [type : \{\text{“ins\_r”}\}, pos : POS, pr : PR, str : STR]$
$OP\_2\_del \triangleq [type : \{\text{“del\_r”}\}, pos : POS, pr : PR, len : LEN]$
$OP\_2 \triangleq OP\_2\_ins \cup OP\_2\_del$

$OP\_3 \triangleq [type : \{\text{“del\_m”}\}, ints : NoncoSeq]$

$NOP\_ALL \triangleq [type : \{\text{“null”}\}, pos : \{\text{“null”}\}]$
$NOP \triangleq \text{CHOOSE } v \in NOP\_ALL : v \in NOP\_ALL$

---

the first kind of opreations

$del\_op(list, pos) \triangleq SubSeq(list, 1, pos - 1) \circ SubSeq(list, pos + 1, Len(list))$
$ins\_op(list, pos, ch) \triangleq SubSeq(list, 1, pos - 1) \circ ch \circ SubSeq(list, pos, Len(list))$
$set\_op(list, pos, ch) \triangleq SubSeq(list, 1, pos - 1) \circ ch \circ SubSeq(list, pos + 1, Len(list))$

the second kind of operations

$ins\_ran\_op(list, pos, str) \triangleq SubSeq(list, 1, pos - 1) \circ str \circ SubSeq(list, pos, Len(list))$ insert interval
$del\_ran\_op(list, pos, len) \triangleq SubSeq(list, 1, pos - 1) \circ SubSeq(list, pos + len, Len(list))$ delete interval

the third kind of operations (how to express many intervals?)

RECURSIVE $del\_mulran\_op(\_, \_, \_)$
$del\_mulran\_op(list, ints, num) \triangleq$
IF $num = 0$ THEN $list$
ELSE $del\_mulran\_op(SubSeq(list, 1, ints[num][1] - 1) \circ SubSeq(list, ints[num][2] + ints[num][1], Len(lis$

the first kind of OT

ins

$Xform\_ins\_ins(lins, rins) \triangleq$
IF $lins.pos < rins.pos$
THEN $lins$
ELSE IF $lins.pos > rins.pos$
THEN $[lins \text{ EXCEPT } !.pos = @ + 1]$
ELSE IF $lins.pr < rins.pr$
THEN $[lins \text{ EXCEPT } !.pos = @ + 1]$
ELSE $lins$

$Xform\_ins\_del(ins, del) \triangleq$
IF $ins.pos \leq del.pos$
THEN ins
ELSE $[ins \text{ EXCEPT } !.pos = @ - 1]$

$Xform\_ins\_set(ins, set) \triangleq ins$

del

$Xform\_del\_ins(del, ins) \triangleq$
IF $del.pos < ins.pos$
THEN del
ELSE $[del \text{ EXCEPT } !.pos = @ + 1]$

$Xform\_del\_del(ldel,\ rdel) \triangleq$
IF $ldel.pos < rdel.pos$
THEN $ldel$
ELSE IF $ldel.pos > rdel.pos$
THEN $[ldel$ EXCEPT $!.pos = @ - 1]$
ELSE NOP

$Xform\_del\_set(del,\ set) \triangleq del$

set
$Xform\_set\_set(lset,\ rset) \triangleq$
IF $lset.pr > rset.pr \wedge lset.pos = rset.pos$
THEN NOP
ELSE $lset$

$Xform\_set\_ins(set,\ ins) \triangleq$
IF $set.pos \geq ins.pos$
THEN $[set$ EXCEPT $!.pos = @ + 1]$
ELSE set

$Xform\_set\_del(set,\ del) \triangleq$
IF $set.pos > del.pos$
THEN $[set$ EXCEPT $!.pos = @ - 1]$
ELSE IF $set.pos < del.pos$
THEN set
ELSE NOP

the second kind of OT
$Xform\_ins\_ins\_r(lins,\ rins) \triangleq$
IF $lins.pos < rins.pos$
THEN $lins$
ELSE IF $lins.pos > rins.pos$
THEN $[lins$ EXCEPT $!.pos = @ + Len(rins.str)]$
ELSE IF $lins.pr > rins.pr$
THEN $lins$
ELSE $[lins$ EXCEPT $!.pos = @ + Len(rins.str)]$

$Xform\_ins\_del\_r(ins,\ del) \triangleq$
CASE $ins.pos \leq del.pos \rightarrow ins$
$\square ins.pos > del.pos \wedge ins.pos < del.pos + del.len \rightarrow NOP$
$\square ins.pos \geq del.pos + del.len \rightarrow [ins$ EXCEPT $!.pos = @ - del.len]$

$Xform\_del\_ins\_r(del,\ ins) \triangleq$
CASE $ins.pos \leq del.pos \rightarrow [del$ EXCEPT $!.pos = @ + Len(ins.str)]$
$\square ins.pos > del.pos \wedge ins.pos < del.pos + del.len \rightarrow [del$ EXCEPT $!.len = @ + Len(ins.str)]$
$\square ins.pos \geq del.pos + del.len \rightarrow del$

$Xform\_del\_del\_r(ldel,\ rdel) \triangleq$
CASE $ldel.pos + ldel.len \leq rdel.pos \to ldel$
$\Box ldel.pos < rdel.pos \wedge\ \ ldel.pos + ldel.len > rdel.pos \wedge ldel.pos + ldel.len \leq rdel.pos + rdel.len \to [ldel$ EXC
$\Box ldel.pos < rdel.pos \wedge ldel.pos + ldel.len > rdel.pos + rdel.len \to [ldel$ EXCEPT $!.len = ldel.len - rdel.len]$
$\Box ldel.pos < rdel.pos \wedge ldel.pos + ldel.len > rdel.pos + rdel.len \to [ldel$ EXCEPT $!.len = ldel.len - rdel.len]$
$\Box ldel.pos \geq rdel.pos \wedge ldel.pos < rdel.pos + rdel.len \wedge ldel.pos + ldel.len \leq rdel.pos + rdel.len$
$\to NOP$
$\Box ldel.pos \geq rdel.pos \wedge ldel.pos < rdel.pos + rdel.len \wedge ldel.pos + ldel.len > rdel.pos + rdel.len$
$\to [ldel$ EXCEPT $!.pos = rdel.pos,\ !.len = ldel.pos + ldel.len - rdel.pos - rdel.len]$
$\Box ldel.pos \geq rdel.pos + rdel.len \to [ldel$ EXCEPT $!.pos = @ - rdel.len]$

RECURSIVE $dellen(\_, \_, \_, \_)$
$dellen(del,\ pos,\ len,\ i) \triangleq$ IF $i = 0$ THEN len
ELSE IF $del.ints[i][1] + del.ints[i][2] \leq pos$ THEN $dellen(del,\ pos,\ len + del.ints[i][$
ELSE $dellen(del,\ pos,\ len,\ i - 1)$

$Xform\_insr\_delm(ins,\ del) \triangleq$
CASE $ins.pos \leq del.ints[1][1] \to ins$
$\Box \exists\, i \in 1\,..\,Len(del.ints) : del.ints[i][1] < ins.pos \wedge ins.pos < del.ints[i][1] + del.ints[i][2] \to NOP$
$\Box \exists\, i \in 1\,..\,Len(del.ints) - 1 : ins.pos \geq del.ints[i][1] +\ \ del.ints[i][2] \wedge ins.pos \leq del.ints[i+1][1] \to [ins$ I
$\Box ins.pos \geq del.ints[Len(del.ints)][1] + del.ints[Len(del.ints)][2] \to [ins$ EXCEPT $!.pos = @ - dellen(del,\ @$

RECURSIVE $transdel1(\_, \_, \_)$
$transdel1(del,\ i,\ len) \triangleq$ IF $Len(del.ints) = i$ THEN $[del$ EXCEPT $!.ints[i][1] = @ + len]$
ELSE $transdel1([del$ EXCEPT $!.ints[i][1] = @ + len],\ i + 1,\ len)$

RECURSIVE $transdel2(\_, \_, \_, \_)$
$transdel2(del,\ i,\ pos,\ len) \triangleq$ IF $Len(del.ints) < i$ THEN del
ELSE IF $del.ints[i][1] < pos \wedge pos < del.ints[i][1] + del.ints[i][2]$ THEN $transdel$
ELSE IF $del.ints[i][1] > pos$ THEN $transdel2([del$ EXCEPT $!.ints[i][1] = @ + le$
ELSE IF $Len(del.ints) = i$ THEN $[del$ EXCEPT $!.ints[i][1] = @ + len]$
ELSE $transdel2(del,\ i + 1,\ pos,\ len)$

RECURSIVE $transdel3(\_, \_, \_, \_)$
$transdel3(del,\ i,\ pos,\ len) \triangleq$ IF $Len(del.ints) < i$ THEN del
ELSE IF $pos \leq del.ints[i][1]$ THEN $transdel3([del$ EXCEPT $!.ints[i][1] = @ + le$
ELSE IF $Len(del.ints) = i$ THEN $[del$ EXCEPT $!.ints[i][1] = @ + len]$
ELSE $transdel3(del,\ i + 1,\ pos,\ len)$

$Xform\_delm\_insr(del,\ ins) \triangleq$
CASE $ins.pos \leq del.ints[1][1] \to transdel1(del,\ 1,\ Len(ins.str))$
$\Box \exists\, i \in 1\,..\,Len(del.ints) : del.ints[i][1] < ins.pos \wedge ins.pos < del.ints[i][1] + del.ints[i][2] \to transdel2(del,$
$\Box \exists\, i \in 1\,..\,Len(del.ints) - 1 : ins.pos \geq del.ints[i][1] +\ \ del.ints[i][2] \wedge ins.pos \leq del.ints[i+1][1] \to trans$
$\Box ins.pos \geq del.ints[Len(del.ints)][1] + del.ints[Len(del.ints)][2] \to del$

```
RECURSIVE Dlen(_, _, _)
Dlen(ints, i, sum) ≜ IF i = 0 THEN sum
                     ELSE Dlen(ints, i − 1, sum + ints[i][2])

RECURSIVE newpos(_, _, _)
newpos(pos, ints, i) ≜
    IF pos < ints[1][1] THEN pos
     ELSE IF i = Len(ints) ∧ pos ≥ ints[i][1] + ints[i][2] THEN pos − Dlen(ints, i, 0)
     ELSE IF pos ≥ ints[i][1] ∧ pos < ints[i][1] + ints[i][2] THEN ints[i][1] − Dlen(ints, i − 1, 0)
     ELSE IF ints[i][1] + ints[i][2] ≤ pos ∧ pos < ints[i + 1][1] THEN pos − Dlen(ints, i, 0)
     ELSE newpos(pos, ints, i + 1)

RECURSIVE newlen(_, _, _, _, _)
newlen(pos, len, ints, i, sum) ≜
    IF i > Len(ints) THEN len − sum
     ELSE IF pos + len < ints[i][1] THEN newlen(pos, len, ints, i + 1, sum)
     ELSE IF pos < ints[i][1] ∧ ints[i][1] < pos + len ∧ pos + len ≤ ints[i][1] + ints[i][2] THEN newlen(pos, len,
     ELSE IF pos < ints[i][1] ∧ pos + len > ints[i][1] + ints[i][2] THEN newlen(pos, len, ints, i + 1, sum + ints
     ELSE IF ints[i][1] ≤ pos ∧ pos < ints[i][1] + ints[i][2] ∧ pos + len ≤ ints[i][1] + ints[i][2] THEN 0
     ELSE IF ints[i][1] ≤ pos ∧ pos < ints[i][1] + ints[i][2] ∧ pos + len > ints[i][1] + ints[i][2] THEN newlen(pos
     ELSE newlen(pos, len, ints, i + 1, sum)

transdel4(int, ints) ≜
  ⟨newpos(int[1], ints, 1), newlen(int[1], int[2], ints, 1, 0) ⟩

RECURSIVE Xform_del_del_m(_, _, _)
Xform_del_del_m(ldel, rdel, i) ≜ IF i > Len(ldel.ints) THEN ldel
                                 ELSE Xform_del_del_m([ldel EXCEPT !.ints[i] = transdel4(@, rdel.ints)],

Xform_insr_set(ins, set) ≜ ins
Xform_set_insr(set, ins) ≜
    IF set.pos ≥ ins.pos
     THEN [set EXCEPT !.pos = @ + Len(ins.str)]
     ELSE set
Xform_delm_set(del, set) ≜ del
Xform_set_delm(set, del) ≜
    CASE set.pos < del.ints[1][1] → set
    □∃ i ∈ 1 .. Len(del.ints) : del.ints[i][1] ≤ set.pos ∧ set.pos < del.ints[i][1] + del.ints[i][2] → NOP
    □∃ i ∈ 1 .. Len(del.ints) − 1 : set.pos ≥ del.ints[i][1] + del.ints[i][2] ∧ set.pos < del.ints[i + 1][1]
          → [set EXCEPT !.pos = @ − dellen(del, @, 0, Len(del.ints))]
    □set.pos ≥ del.ints[Len(del.ints)][1] + del.ints[Len(del.ints)][2]
            → [set EXCEPT !.pos = @ − dellen(del, @, 0, Len(del.ints))]
```

```
Xform(lop, rop) ≜  the left operation is transformed against the right operation
    CASE lop.type = "ins" ∧ rop.type = "ins" → Xform_ins_ins(lop, rop)
    □ lop.type = "ins" ∧ rop.type = "del" → Xform_ins_del(lop, rop)
```

$\square$ $lop.type =$ "ins" $\wedge$ $rop.type =$ "set" $\rightarrow$ $Xform\_ins\_set(lop,\ rop)$

$\square$ $lop.type =$ "del" $\wedge$ $rop.type =$ "ins" $\rightarrow$ $Xform\_del\_ins(lop,\ rop)$
$\square$ $lop.type =$ "del" $\wedge$ $rop.type =$ "del" $\rightarrow$ $Xform\_del\_del(lop,\ rop)$
$\square$ $lop.type =$ "del" $\wedge$ $rop.type =$ "set" $\rightarrow$ $Xform\_del\_set(lop,\ rop)$

$\square$ $lop.type =$ "set" $\wedge$ $rop.type =$ "ins" $\rightarrow$ $Xform\_set\_ins(lop,\ rop)$
$\square$ $lop.type =$ "set" $\wedge$ $rop.type =$ "del" $\rightarrow$ $Xform\_set\_del(lop,\ rop)$
$\square$ $lop.type =$ "set" $\wedge$ $rop.type =$ "set" $\rightarrow$ $Xform\_set\_set(lop,\ rop)$

$\square$ $lop.type =$ "ins_r" $\wedge$ $rop.type =$ "ins_r" $\rightarrow$ $Xform\_ins\_ins\_r(lop,\ rop)$
$\square$ $lop.type =$ "ins_r" $\wedge$ $rop.type =$ "del_r" $\rightarrow$ $Xform\_ins\_del\_r(lop,\ rop)$
$\square$ $lop.type =$ "del_r" $\wedge$ $rop.type =$ "del_r" $\rightarrow$ $Xform\_del\_del\_r(lop,\ rop)$
$\square$ $lop.type =$ "del_r" $\wedge$ $rop.type =$ "ins_r" $\rightarrow$ $Xform\_del\_ins\_r(lop,\ rop)$

$\square$ $lop.type =$ "ins_r" $\wedge$ $rop.type =$ "del_m" $\rightarrow$ $Xform\_insr\_delm(lop,\ rop)$
$\square$ $lop.type =$ "del_m" $\wedge$ $rop.type =$ "ins_r" $\rightarrow$ $Xform\_delm\_insr(lop,\ rop)$
$\square$ $lop.type =$ "del_m" $\wedge$ $rop.type =$ "del_m" $\rightarrow$ $Xform\_del\_del\_m(lop,\ rop,\ 1)$

$\square$ $lop.type =$ "ins_r" $\wedge$ $rop.type =$ "set" $\rightarrow$ $Xform\_insr\_set(lop,\ rop)$
$\square$ $lop.type =$ "set" $\wedge$ $rop.type =$ "ins_r" $\rightarrow$ $Xform\_set\_insr(lop,\ rop)$
$\square$ $lop.type =$ "del_m" $\wedge$ $rop.type =$ "set" $\rightarrow$ $Xform\_delm\_set(lop,\ rop)$
$\square$ $lop.type =$ "set" $\wedge$ $rop.type =$ "del_m" $\rightarrow$ $Xform\_set\_delm(lop,\ rop)$

$apply(list,\ op) \triangleq$ apply operation to a list
CASE $op.type =$ "del" $\rightarrow$ $del\_op(list,\ op.pos)$
$\square op.type =$ "ins" $\rightarrow$ $ins\_op(list,\ op.pos,\ op.ch)$
$\square op.type =$ "set" $\rightarrow$ $set\_op(list,\ op.pos,\ op.ch)$
$\square op.type =$ "ins_r" $\rightarrow$ $ins\_ran\_op(list,\ op.pos,\ op.str)$
$\square op.type =$ "del_r" $\rightarrow$ $del\_ran\_op(list,\ op.pos,\ op.len)$
$\square op.type =$ "del_m" $\rightarrow$ $del\_mulran\_op(list,\ op.ints,\ Len(op.ints))$
$\square$OTHER $\rightarrow$ $list$

$correctness\_1(list) \triangleq$ $CP1$ correctness of the first kind of functions
$\forall\ op1,\ op2 \in OP\_1$:
$\vee\ op1.pr = op2.pr$
$\vee\ apply(apply(list,\ op1),\ Xform(op2,\ op1)) = apply(apply(list,\ op2),\ Xform(op1,\ op2))$

$correctness\_2(list) \triangleq$ $CP1$ correctness of the second kind of functions
$\forall\ op1,\ op2 \in OP\_2$:
$\vee\ op1.pr = op2.pr$
$\vee\ apply(apply(list,\ op1),\ Xform(op2,\ op1)) = apply(apply(list,\ op2),\ Xform(op1,\ op2))$

$correctness\_3(list) \triangleq$ $CP1$ correctness of the third kind of functions
$\forall\ op1,\ op2 \in OP\_3,\ op3 \in OP\_2\_ins$:
$\wedge\ apply(apply(list,\ op1),\ Xform(op2,\ op1)) = apply(apply(list,\ op2),\ Xform(op1,\ op2))$
$\wedge\ apply(apply(list,\ op1),\ Xform(op3,\ op1)) = apply(apply(list,\ op3),\ Xform(op1,\ op3))$

$correctness\_4(list) \triangleq$ $CP1$ correctness of the fourth kind of functions
$\forall\, op1 \in OP\_1\_set,\ op2 \in OP\_2\_ins,\ op3 \in OP\_3 :$
$\wedge\ apply(apply(list,\ op1),\ Xform(op2,\ op1)) = apply(apply(list,\ op2),\ Xform(op1,\ op2))$
$\wedge\ apply(apply(list,\ op1),\ Xform(op3,\ op1)) = apply(apply(list,\ op3),\ Xform(op1,\ op3))$

$correctness(list) \triangleq correctness\_1(list) \wedge correctness\_2(list) \wedge correctness\_3(list) \wedge correctness\_4(list)$

* Modification History
* Last modified *Tue Dec* 18 19:46:06 *CST* 2018 by *xhdn*
* Created *Wed Apr* 18 14:07:40 *CST* 2018 by *xhdn*