MODULE *TLAToPCal*

EXTENDS *Integers*, *Sequences*, *TLC*

We define the minimum and maximum of a nonempty set of numbers, and the absolute value of a number.

$Min(S) \triangleq$ CHOOSE $i \in S : \forall\, j \in S : i \leq j$
$Max(S) \triangleq$ CHOOSE $i \in S : \forall\, j \in S : i \geq j$
$Abs(x) \triangleq Max(\{x, -x\})$

TP Mapping Specifiers .

$Location \triangleq [line : Nat,\ column : Nat]$

This is a location in the file, which might be in the TLA+ spec or in the *PCal* code.

$loc1 <: loc2 \triangleq$

This is the "equals or to the left of" relation on locations.

$\lor\ loc1.line < loc2.line$
$\lor\ \land\ loc1.line = loc2.line$
$\quad \land\ loc1.column \leq loc2.column$

We define $Dist(loc1, loc2)$ to be a natural number representing a distance between locations $loc1$ and $loc2$. This distance is used only to determine which of two other locations a location is closer to. Thus, its magnitude doesn't matter. I should make *Dist* a parameter of the spec, but it's less effort to give it some reasonable definition.

$Dist(loc1, loc2) \triangleq$
$\quad 10000 * Abs(loc2.line - loc1.line) + Abs(loc2.column - loc1.column)$

$Region \triangleq \{r \in [begin : Location,\ end : Location] : r.begin <: r.end\}$

This describes a region within the file. We say that region $r1$ is to the left of region $r2$ iff $r1.end <: r2.begin$

TLA to *PCal* translation objects.

$TLAToken \triangleq [type : \{\text{"token"}\},\ region : Region,\ inExpr : \text{BOOLEAN}\ ]$

This represents a region of tokens in the TLA+ spec, with *inExpr* being true iff that region lies within an expression.

$Paren \triangleq [type : \{\text{"begin"}, \text{"end"}\},\ loc : Location]$

This represents the beginning or end of a region in the *PlusCal* spec.

$Break \triangleq [type : \{\text{"break"}\},\ depth : Nat]$

A *Break* comes between a right and left *Paren* at the same parenthesis level (possibly with *TLATokens* also between them). It indicates that there is some *PlusCal* code between the locations indicated by those parentheses that should not be displayed when displaying the *PlusCal* code for parenthesis levels between the current level lv and $lv - depth$.

$TPObject \triangleq TLAToken \cup Paren \cup Break$

RECURSIVE $ParenDepth(\_, \_)$
$ParenDepth(objSeq, pos) \triangleq$

Equals the parenthesis depth of the point in the *TPObject* sequence *objSeq* just after element number *pos*, or at the beginning if $pos = 0$.

```
IF pos = 0 THEN 0
   ELSE LET obj ≜ objSeq[pos]
        IN  ParenDepth(objSeq, pos − 1) +
              (CASE obj.type = "begin" → 1    □
                    obj.type = "end"   → −1   □
                    OTHER              → 0    )
```

WellFormed(*seq*) is true for a *TPObject* sequence iff it begins and ends with a parenthesis and all parentheses are properly matching.

$$\begin{aligned} IsWellFormed(seq) \triangleq\ & \land\ seq \neq \langle\rangle \\ & \land\ seq[1].type = \text{"begin"} \\ & \land\ \forall\, i \in 1\,..\,(Len(seq)-1) : ParenDepth(seq,\, i) \geq 0 \\ & \land\ ParenDepth(seq,\, Len(seq)) = 0 \end{aligned}$$

TokensInOrder(*seq*) is true for a *TPObject* sequence iff its *TLAToken* objects represent regions that are ordered properly–that is, if *TLAToken* $T1$ precedes *TLAToken* $T2$ in *seq*, then $T1.region$ is to the left of $T2.region$.

$$TokensOf(seq) \triangleq \{i \in 1\,..\,Len(seq) : seq[i].type = \text{"token"}\}$$

$$\begin{aligned} & TokensInOrder(seq) \triangleq \\ & \quad \forall\, i \in TokensOf(seq) : \\ & \qquad \forall\, j \in \{jj \in TokensOf(seq) : jj > i\} : \\ & \qquad\quad (seq[i].region.end <: seq[j].region.begin) \end{aligned}$$

$$MatchingParen(seq,\, pos) \triangleq$$

If element number *pos* in *TPObject* sequence *seq* is a left paren, then this equals the number n such that element number n is the matching right paren.

$$\begin{aligned} & \text{CHOOSE } i \in pos+1\,..\,Len(seq) : \\ & \quad \land\ ParenDepth(seq,\, i) = ParenDepth(seq,\, pos-1) \\ & \quad \land\ \forall\, j \in (pos)\,..\,(i-1) : ParenDepth(seq,\, j) > ParenDepth(seq,\, pos-1) \end{aligned}$$

A *TPMap* is a sequence of *TPObject* elements that has the following interpretation. The regions of the TLA+ spec contained within its *TLAToken* elements contain the "important" text of the spec. Text not in those regions can be treated as if it were white space when determining the *PCal* region that maps to a part of the TLA+ spec.

Each pair of matching parentheses defines the smallest syntactic unit (*e.g.*, expression or statement) whose translation contains the text in the *TLATokens* between them. All the top level (lowest depth) parentheses between those matching parentheses describe successive regions in the same part of the *PCal* text. (Code in a macro and code in a procedure is an example of two regions in completely different parts of the *PCal* code, and hence are not successive regions.) Two successive regions are adjacent if, to higlight both of them, one highlights those regions and the text between them. If two successive regions represented by two pairs of matching parentheses are not adjacent, then the *TPMap* contains a *Break* between them. The depth of a break indicates the number of parenthesis levels containing the break that represent *PCal* code in which the region between the parenthesized regions on either side of the break should not be highlighted.

The following predicate asserts that *seq* is a proper *TPMap*.

$$IsTPMap(seq) \triangleq$$

There is at least one *TLAToken* between every matching pair of parentheses.

$\land \forall\, i \in 1 \,..\, Len(seq) :$
$\quad (seq[i].type = \text{“begin”}) \Rightarrow$
$\quad\quad \exists\, j \in (i+1) \,..\, (MatchingParen(seq,\, i) - 1) : seq[j].type = \text{“token”}$

A token in an expression is surrounded by parentheses.

$\land \forall\, i \in TokensOf(seq) : seq[i].inExpr \Rightarrow \quad \land\, seq[i-1].type = \text{“begin”}$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \land\, seq[i+1].type = \text{“end”}$
$\land\, IsWellFormed(seq)$
$\land\, TokensInOrder(seq)$

The following conjunct asserts that a *Break* comes between a right and a left parenthesis at its level, perhaps with intervening tokens.

$\land \forall\, i \in 1 \,..\, Len(seq) :$
$\quad (seq[i].type = \text{“break”}) \Rightarrow$
$\quad\quad \land \exists\, j \in 1 \,..\, (i-1) :$
$\quad\quad\quad \land\, seq[j].type = \text{“end”}$
$\quad\quad\quad \land \forall\, k \in (j+1) \,..\, (i-1) : seq[j].type \neq \text{“begin”}$
$\quad\quad \land\ \exists\, j \in (i+1) \,..\, Len(seq) :$
$\quad\quad\quad \land\, seq[j].type = \text{“begin”}$
$\quad\quad\quad \land \forall\, k \in (i+1) \,..\, (j-1) : seq[j].type \neq \text{“end”}$

The following conjunct asserts that matching parentheses have non-decreasing locations, and that within a pair of matched parentheses, the regions represented by the top-level matching parentheses are properly ordered.

$\land \forall\, i \in 1 \,..\, Len(seq) :$
$\quad (seq[i].type = \text{“begin”}) \Rightarrow$
$\quad\quad \text{LET } j \triangleq MatchingParen(seq,\, i)$
$\quad\quad\quad\ \ dp \triangleq ParenDepth(seq,\, i-1) + 1$
$\quad\quad \text{IN} \quad \land\, seq[i].loc <: seq[j].loc$
$\quad\quad\quad\quad\ \land \forall\, k \in (i+1) \,..\, (j-1) :$
$\quad\quad\quad\quad\quad\ \land\, seq[k].type = \text{“end”}$
$\quad\quad\quad\quad\quad\ \land\, ParenDepth(seq,\, k) = dp$
$\quad\quad\quad\quad\quad\ \Rightarrow \forall\, m \in (k+1) \,..\, (j-1) :$
$\quad\quad\quad\quad\quad\quad\ \land\, seq[m].type = \text{“begin”}$
$\quad\quad\quad\quad\quad\quad\ \land\, ParenDepth(seq,\, m-1) = dp$
$\quad\quad\quad\quad\quad\quad\ \Rightarrow seq[k].loc <: seq[m].loc$

$TPMap \triangleq \{s \in Seq(TPObject) : IsTPMap(s)\}$

The *Region* in the *PCal* code specified by a *Region* in the TLA+ spec.

$RegionToTokPair(spec,\, reg) \triangleq$

A pair of integers that are the positions of the pair of *TLATokens* in spec such that they and the *TLATokens* between them are the ones that the user has chosen if she has highlighted the region specified by *reg*. (Both tokens could be the same.)

If the region *reg* does not intersect with the region of any *TLAToken* (so it lies entirely inside “white space”), then the value is $\langle t,\, t \rangle$ for the token t that lies either to the left or the right of *reg*.

```
LET TokensContaining(loc) ≜
      The set of positions of tokens in spec containing loc. (It contains 0 or 1 element.)
      {i ∈ TokensOf(spec) : ∧ spec[i].region.begin <: loc
                            ∧ spec[i].region.begin ≠ loc
                            ∧ loc <: spec[i].region.end
                            ∧ loc ≠ spec[i].region.end }

    TokensToLeft(loc) ≜
      The set of positions of tokens in spec at or to the left of loc.
      {i ∈ TokensOf(spec) : spec[i].region.end <: loc}

    TokensToRight(loc) ≜
      The set of positions of tokens in spec at or to the right of loc.
      {i ∈ TokensOf(spec) : loc <: spec[i].region.begin}

    TokensInRegion ≜
      The set of tokens whose regions lie within reg.
      TokensToRight(reg.begin) ∩ TokensToLeft(reg.end)

    S ≜ TokensInRegion ∪ TokensContaining(reg.begin)
          ∪ TokensContaining(reg.end)

IN  IF S ≠ {}
    THEN ⟨Min(S), Max(S)⟩
    ELSE LET LeftOfReg ≜ TokensToLeft(reg.begin)
             LeftTok ≜ Max(LeftOfReg)
             RightOfReg ≜ TokensToRight(reg.end)
             RightTok ≜ Min(RightOfReg)
         IN  CASE LeftOfReg = {}  → ⟨RightTok, RightTok⟩ □
                  RightOfReg = {} → ⟨LeftTok, LeftTok⟩   □
                  OTHER →
                    LET dl ≜ Dist(spec[LeftTok].region.end, reg.begin)
                        dr ≜ Dist(spec[RightTok].region.begin, reg.end)
                    IN  CASE dl < dr → ⟨LeftTok, LeftTok⟩   □
                             dl > dr → ⟨RightTok, RightTok⟩ □
                             dl = dr → ⟨LeftTok, RightTok⟩

TokPairToParens(spec, ltok, rtok) ≜
  Assumes ltok and rtok are the positions of TLAToken elements of the TPMap spec with ltok
  equal to or to the left of rtok. It equals the pair ⟨lparen, rparen⟩ where lparen is the
  position of the right-most left paren to the left of ltok that enters level dp and rparen is
  the position of the left-most right paren to the right of rtok that leaves level dp, where
  dp is defined as follows:

  Let d be the minimum paren depth any token from ltok and rtok. If ltok ≠ rtok and every
  TLAToken element from positions ltok through rtok is an expression token, then dp = d + 1.
  Otherwise, dp = d.
  LET d ≜ Min({ParenDepth(spec, i) : i ∈ ltok .. rtok})
```

$dp \triangleq$ IF $\land\, ltok \neq rtok$
$\land\, \forall\, i \in ltok \,..\, rtok : (spec[i].type = \text{“token”}) \Rightarrow spec[i].inExpr$
THEN $d + 1$
ELSE d
$lp \triangleq Max(\{i \in 1 \,..\, ltok : \land\, spec[i].type = \text{“begin”} \land\, ParenDepth(spec,\, i) = dp\})$
$rp \triangleq Min(\{i \in rtok \,..\, Len(spec) : \land\, spec[i].type = \text{“end”} \land\, ParenDepth(spec,\, i-1) = dp\})$
IN $\langle lp,\, rp \rangle$

For Debugging

To simplify debugging, we assume that locations are all on the same line.

$Loc(pos) \triangleq [line \mapsto 0,\, column \mapsto pos]$
$Reg(beg,\, end) \triangleq [begin \mapsto Loc(beg),\, end \mapsto Loc(end)]$
$T(beg,\, end) \triangleq [type \mapsto \text{“token”},\, region \mapsto Reg(beg,\, end),\, inExpr \mapsto \text{FALSE}]$
$TE(beg,\, end) \triangleq [type \mapsto \text{“token”},\, region \mapsto Reg(beg,\, end),\, inExpr \mapsto \text{TRUE}]$
$L(pos) \triangleq [type \mapsto \text{“begin”},\, loc \mapsto Loc(pos)]$
$R(pos) \triangleq [type \mapsto \text{“end”},\, loc \mapsto Loc(pos)]$
$B(dep) \triangleq [type \mapsto \text{“break”},\, depth \mapsto dep]$

$tpMap\_1 \triangleq \langle L(-5),\, T(2,\, 3),\, L(11),\, T(3,\, 4),\, L(12),\, T(4,\, 5),\, R(13),$
$T(6,\, 7),\, R(14),\, T(8,\, 9),\, R(42)\rangle$
$tpRegion\_1 \triangleq Reg(5,\, 20)$

$tpMap\_2 \triangleq \langle L(10),\, T(1,\, 2),\, L(11),\, T(3,\, 4),\, L(12),\, T(5,\, 6),\, L(13),\, T(7,\, 8),\, R(14),$
$T(9,\, 10),\ R(15),\, B(1),\, L(16),\, T(11,\, 12),\, L(17),\, T(13,\, 14),\, R(18),$
$R(19),\, T(15,\, 16),\, R(20),\, R(21)\rangle$

$tpRegion1 \triangleq Reg(0,\, 16)$

$tpMap1 \triangleq \langle L(1),\, L(2),\, TE(1,\, 2),\, R(3),\, L(4),\, TE(2,\, 3),$
$R(5),\, T(4,\, 5),\, L(6),\, TE(5,\, 6),\, R(7),\, R(8)\rangle$

$SpecToRegions(spec) \triangleq$

The set of all regions whose endpoints are in the smallest region containing all the tokens of spec.

LET $TT \triangleq TokensOf(spec)$
$left \triangleq Min(\{spec[i].region.begin.column : i \in TT\})$
$right \triangleq Max(\{spec[i].region.end.column : i \in TT\})$
IN $\{Reg(r[1],\, r[2]) :$
$r \in \{rr \in (left \,..\, right) \times (left \,..\, right) : rr[1] \leq rr[2]\}\}$

Declare *tpMap* to be the *TPMap* and *tpLoc* the *Location* that are the inputs to the algorithm.

CONSTANT $tpMap,\, tpRegion$

```
***************************************************************************
                    The Mapping Algorithm

This algorithm sets the variable result to the sequence of regions in the PCal code that, according
to the mapping specification tpMap, should be highlighted when the user selects the region that is
the value of variable tpregion. (Variable tpregion is initialized to tpRegion to test a single region,
and to SpecToRegions(tpMap) to test all subregions.) In the initial Java implementation, I expect
that result will be set to a sequence containing only a single region.

–fair algorithm Map {
  variables
    tpregion \ * = tpRegion,
          ∈ SpecToRegions(tpMap),
    ltok,      \ * ⟨ltok, rtok⟩is set to
    rtok,      \ * RegionToTokPair(tpMap, tpregion)
    rtokDepth, \ * The paren depth of rtok relative to ltok
    minDepth, \ * The depth of the minimum paren depth TLAToken
    allExpr, \ * Set to true iff all tokens from ltok to rtok are
           \ * expression tokens.
    bParen, \ * ⟨bParen, eParen⟩is set to
    eParen, \ * TokPairToParens(tpMap, ltok, rtok)
    result, \ * Set to the sequence of Regions that is the translation.
    curBegin, \ * Used to construct the result
    lastRparen, \ * "
    i,         \ * For loop variable
    curDepth ; \ * Temporary variable for holding the paren depth



  macro ModifyDepth(var, pos, movingForward){
   with(amt = CASE  tpMap[pos].type = "begin" → 1 □
                tpMap[pos].type = "end" → − 1 □
                OTHER                → 0    ){
     var := var + IF  movingForward  THEN amt  ELSE  − amt
   }
  }

  {with(tp = RegionToTokPair(tpMap, tpregion)){
     ltok := tp[1];
     rtok := tp[2]
   };



   rtokDepth := 0;
   minDepth := 0 ;
   allExpr := tpMap[ltok].inExpr ;
   i := ltok + 1;
   while(i ≤ rtok){
     ModifyDepth(rtokDepth, i, TRUE);
     if(rtokDepth < minDepth){minDepth := rtokDepth};
```

```
  if(tpMap[i].type = "token"){
    allExpr := allExpr ∧ tpMap[i].inExpr
   };
  i := i + 1
 };
assert ∧ ParenDepth(tpMap, rtok) = ParenDepth(tpMap, ltok) + rtokDepth
      ∧ minDepth + ParenDepth(tpMap, ltok) =
          Min({ParenDepth(tpMap, k) : k ∈ ltok .. rtok})
      ∧ allExpr = ∀ k ∈ ltok .. rtok :
                  (tpMap[k].type = "token") ⇒ tpMap[k].inExpr ;

if(ltok ≠ rtok ∧ allExpr){minDepth := minDepth + 1};

curDepth := 0;
i := ltok − 1;
while(¬ ∧ tpMap[i].type = "begin"
       ∧ curDepth = minDepth){
  ModifyDepth(curDepth, i, FALSE);
  i := i − 1
 };
bParen := i ;

curDepth := rtokDepth;
i := rtok + 1;
while(¬ ∧ tpMap[i].type = "end"
       ∧ curDepth = minDepth){
  ModifyDepth(curDepth, i, TRUE);
  i := i + 1
 };
eParen := i ;
assert⟨bParen, eParen⟩ = TokPairToParens(tpMap, ltok, rtok);

result := ⟨⟩;
curBegin := tpMap[bParen].loc ;
curDepth := 0 ;
lastRparen := − 1 ;
i := bParen + 1 ;
while(i < eParen){
  if(tpMap[i].type = "end"){
    lastRparen := i
```

```
      };
     if( ∧ tpMap[i].type = "break"
         ∧ tpMap[i].depth − curDepth ≥ 0){
         assert lastRparen ≠ −1 ;



         result := Append(result,
                     [begin ↦ curBegin, end ↦ tpMap[lastRparen].loc]);

         lastRparen := −1 ;
         while(tpMap[i].type ≠ "begin"){
           ModifyDepth(curDepth, i, TRUE);
           i := i + 1;
          };
         curBegin := tpMap[i].loc ;
        };
      ModifyDepth(curDepth, i, TRUE);
      i := i + 1;
     };
    result := Append(result,
                [begin ↦ curBegin, end ↦ tpMap[eParen].loc]);

    \ * debugging output
    print⟨tpregion.begin.column, tpregion.end.column⟩;
    print[j ∈ 1 .. Len(result) ↦ ⟨result[j].begin.column,
                                  result[j].end.column⟩];
    print⟨"lrtok", ltok, rtok⟩
   }
}
*****************************************************************************
```

BEGIN TRANSLATION

CONSTANT *defaultInitValue*

VARIABLES *tpregion*, *ltok*, *rtok*, *rtokDepth*, *minDepth*, *allExpr*, *bParen*, *eParen*, *result*, *curBegin*, *lastRparen*, *i*, *curDepth*, *pc*

$vars \triangleq \langle tpregion, ltok, rtok, rtokDepth, minDepth, allExpr, bParen, eParen, result, curBegin, lastRparen, i, curDepth, pc\rangle$

$Init \triangleq$ Global variables

$\wedge\ tpregion \in SpecToRegions(tpMap)$
$\wedge\ ltok = defaultInitValue$
$\wedge\ rtok = defaultInitValue$
$\wedge\ rtokDepth = defaultInitValue$
$\wedge\ minDepth = defaultInitValue$
$\wedge\ allExpr = defaultInitValue$
$\wedge\ bParen = defaultInitValue$
$\wedge\ eParen = defaultInitValue$
$\wedge\ result = defaultInitValue$
$\wedge\ curBegin = defaultInitValue$

$\land\ lastRparen = defaultInitValue$
$\land\ i = defaultInitValue$
$\land\ curDepth = defaultInitValue$
$\land\ pc =$ "Lbl_1"

$Lbl\_1 \triangleq$ $\land\ pc =$ "Lbl_1"
$\land$ LET $tp \triangleq RegionToTokPair(tpMap,\ tpregion)$ IN
$\land\ ltok' = tp[1]$
$\land\ rtok' = tp[2]$
$\land\ rtokDepth' = 0$
$\land\ minDepth' = 0$
$\land\ allExpr' = tpMap[ltok'].inExpr$
$\land\ i' = ltok' + 1$
$\land\ pc' =$ "Lbl_2"
$\land$ UNCHANGED $\langle tpregion,\ bParen,\ eParen,\ result,\ curBegin,$
$lastRparen,\ curDepth\rangle$

$Lbl\_2 \triangleq$ $\land\ pc =$ "Lbl_2"
$\land$ IF $i \leq rtok$
THEN $\land$ LET $amt \triangleq$ CASE $tpMap[i].type =$ "begin" $\rightarrow\ 1\quad \square$
$tpMap[i].type =$ "end" $\rightarrow\ -1 \square$
OTHER $\rightarrow 0$ IN
$rtokDepth' = rtokDepth +$ IF TRUE THEN amt ELSE $-\ amt$
$\land$ IF $rtokDepth' < minDepth$
THEN $\land\ minDepth' = rtokDepth'$
ELSE $\land$ TRUE
$\land$ UNCHANGED $minDepth$
$\land$ IF $tpMap[i].type =$ "token"
THEN $\land\ allExpr' = (allExpr \land tpMap[i].inExpr)$
ELSE $\land$ TRUE
$\land$ UNCHANGED $allExpr$
$\land\ i' = i + 1$
$\land\ pc' =$ "Lbl_2"
$\land$ UNCHANGED $curDepth$
ELSE $\land\ Assert($ $\land\ ParenDepth(tpMap,\ rtok) = ParenDepth(tpMap,\ ltok) + rtokDepth$
$\land\ minDepth + ParenDepth(tpMap,\ ltok) =$
$Min(\{ParenDepth(tpMap,\ k) : k \in ltok\ ..\ rtok\})$
$\land\ allExpr = \forall\, k \in ltok\ ..\ rtok :$
$(tpMap[k].type =$ "token") $\Rightarrow tpMap[k].inExpr,$
"Failure of assertion at line 397, column 7.")
$\land$ IF $ltok \neq rtok \land allExpr$
THEN $\land\ minDepth' = minDepth + 1$
ELSE $\land$ TRUE
$\land$ UNCHANGED $minDepth$
$\land\ curDepth' = 0$

$\land\ i' = ltok - 1$
$\land\ pc' = \text{“Lbl\_3”}$
$\land$ UNCHANGED $\langle rtokDepth,\ allExpr\rangle$
$\land$ UNCHANGED $\langle tpregion,\ ltok,\ rtok,\ bParen,\ eParen,\ result,$
$curBegin,\ lastRparen\rangle$

$Lbl\_3 \triangleq\ \land\ pc = \text{“Lbl\_3”}$
$\land$ IF $\neg \land\ tpMap[i].type = \text{“begin”}$
$\land\ curDepth = minDepth$
THEN $\land$ LET $amt \triangleq$ CASE $tpMap[i].type = \text{“begin”} \rightarrow\ 1\quad \square$
$tpMap[i].type = \text{“end”}\quad \rightarrow\ -1 \square$
OTHER $\rightarrow 0$ IN
$curDepth' = curDepth +$ IF FALSE THEN amt ELSE $-\ amt$
$\land\ i' = i - 1$
$\land\ pc' = \text{“Lbl\_3”}$
$\land$ UNCHANGED $bParen$
ELSE $\land\ bParen' = i$
$\land\ curDepth' = rtokDepth$
$\land\ i' = rtok + 1$
$\land\ pc' = \text{“Lbl\_4”}$
$\land$ UNCHANGED $\langle tpregion,\ ltok,\ rtok,\ rtokDepth,\ minDepth,\ allExpr,$
$eParen,\ result,\ curBegin,\ lastRparen\rangle$

$Lbl\_4 \triangleq\ \land\ pc = \text{“Lbl\_4”}$
$\land$ IF $\neg \land\ tpMap[i].type = \text{“end”}$
$\land\ curDepth = minDepth$
THEN $\land$ LET $amt \triangleq$ CASE $tpMap[i].type = \text{“begin”} \rightarrow\ 1\quad \square$
$tpMap[i].type = \text{“end”}\quad \rightarrow\ -1 \square$
OTHER $\rightarrow 0$ IN
$curDepth' = curDepth +$ IF TRUE THEN amt ELSE $-\ amt$
$\land\ i' = i + 1$
$\land\ pc' = \text{“Lbl\_4”}$
$\land$ UNCHANGED $\langle eParen,\ result,\ curBegin,\ lastRparen\rangle$
ELSE $\land\ eParen' = i$
$\land\ Assert(\langle bParen,\ eParen'\rangle = TokPairToParens(tpMap,\ ltok,\ rtok),$
“Failure of assertion at line 434, column 7.”)
$\land\ result' = \langle\rangle$
$\land\ curBegin' = tpMap[bParen].loc$
$\land\ curDepth' = 0$
$\land\ lastRparen' = -1$
$\land\ i' = bParen + 1$
$\land\ pc' = \text{“Lbl\_5”}$
$\land$ UNCHANGED $\langle tpregion,\ ltok,\ rtok,\ rtokDepth,\ minDepth,\ allExpr,$
$bParen\rangle$

$Lbl\_5 \triangleq\ \land\ pc = \text{“Lbl\_5”}$

$\land$ IF $i < eParen$
THEN $\land$ IF $tpMap[i].type =$ "end"
THEN $\land\ lastRparen' = i$
ELSE $\land$ TRUE
$\land$ UNCHANGED $lastRparen$
$\land$ IF $\land\ tpMap[i].type =$ "break"
$\land\ tpMap[i].depth - curDepth \geq 0$
THEN $\land\ Assert(lastRparen' \neq -1,$
"Failure of assertion at line 450, column 14.")
$\land\ result' = Append(result,$
$[begin \mapsto curBegin,\ end \mapsto tpMap[lastRparen'].loc])$
$\land\ pc' =$ "Lbl_6"
ELSE $\land\ pc' =$ "Lbl_8"
$\land$ UNCHANGED $result$
ELSE $\land\ result' = Append(result,$
$[begin \mapsto curBegin,\ end \mapsto tpMap[eParen].loc])$
$\land\ PrintT(\langle tpregion.begin.column,\ tpregion.end.column\rangle)$
$\land\ PrintT([j \in 1\,..\,Len(result') \mapsto \langle result'[j].begin.column,$
$result'[j].end.column\rangle])$
$\land\ PrintT(\langle$"lrtok", $ltok,\ rtok\rangle)$
$\land\ pc' =$ "Done"
$\land$ UNCHANGED $lastRparen$
$\land$ UNCHANGED $\langle tpregion,\ ltok,\ rtok,\ rtokDepth,\ minDepth,\ allExpr,$
$bParen,\ eParen,\ curBegin,\ i,\ curDepth\rangle$

$Lbl\_8 \triangleq$ $\land\ pc =$ "Lbl_8"
$\land$ LET $amt \triangleq$ CASE $tpMap[i].type =$ "begin" $\to\ 1\quad \square$
$tpMap[i].type =$ "end" $\to -1 \square$
OTHER $\to 0$ IN
$curDepth' = curDepth +$ IF TRUE THEN amt ELSE $-\ amt$
$\land\ i' = i + 1$
$\land\ pc' =$ "Lbl_5"
$\land$ UNCHANGED $\langle tpregion,\ ltok,\ rtok,\ rtokDepth,\ minDepth,\ allExpr,$
$bParen,\ eParen,\ result,\ curBegin,\ lastRparen\rangle$

$Lbl\_6 \triangleq$ $\land\ pc =$ "Lbl_6"
$\land\ lastRparen' = -1$
$\land\ pc' =$ "Lbl_7"
$\land$ UNCHANGED $\langle tpregion,\ ltok,\ rtok,\ rtokDepth,\ minDepth,\ allExpr,$
$bParen,\ eParen,\ result,\ curBegin,\ i,\ curDepth\rangle$

$Lbl\_7 \triangleq$ $\land\ pc =$ "Lbl_7"
$\land$ IF $tpMap[i].type \neq$ "begin"
THEN $\land$ LET $amt \triangleq$ CASE $tpMap[i].type =$ "begin" $\to\ 1\quad \square$
$tpMap[i].type =$ "end" $\to -1 \square$
OTHER $\to 0$ IN

$$
\begin{array}{rl}
 & curDepth' = curDepth + \text{IF TRUE THEN } amt \text{ ELSE } -amt \\
 & \land\ i' = i + 1 \\
 & \land\ pc' = \text{"Lbl\_7"} \\
 & \land\ \text{UNCHANGED } curBegin \\
\text{ELSE} & \land\ curBegin' = tpMap[i].loc \\
 & \land\ pc' = \text{"Lbl\_8"} \\
 & \land\ \text{UNCHANGED } \langle i,\ curDepth \rangle \\
\land\ \text{UNCHANGED } & \langle tpregion,\ ltok,\ rtok,\ rtokDepth,\ minDepth,\ allExpr, \\
 & \quad bParen,\ eParen,\ result,\ lastRparen \rangle
\end{array}
$$

$$
\begin{array}{rcl}
Next & \triangleq & Lbl\_1 \lor Lbl\_2 \lor Lbl\_3 \lor Lbl\_4 \lor Lbl\_5 \lor Lbl\_8 \lor Lbl\_6 \\
 & & \lor\ Lbl\_7 \\
 & & \lor\ \text{Disjunct to prevent deadlock on termination} \\
 & & \quad (pc = \text{"Done"} \land \text{UNCHANGED } vars)
\end{array}
$$

$$
\begin{array}{rcl}
Spec & \triangleq & \land\ Init \land \Box[Next]_{vars} \\
 & & \land\ \mathrm{WF}_{vars}(Next)
\end{array}
$$

$$Termination \triangleq \Diamond(pc = \text{"Done"})$$

END TRANSLATION

\ * Modification History
\ * Last modified *Mon Jan* 09 15:51:06 *PST* 2012 by *lamport*
\ * Created *Thu Dec* 01 16:51:23 *PST* 2011 by *lamport*