—— MODULE *Executor* ——

This specification models a subset of *X*10 programs that use finish, *async* at (place), simple expression statements, or error statements that raise exceptions. The following is a sample program that this specification can validate

finish {
 expr;
 async at(*p*1){
  expr;
  async at(*p*2){
   expr;
   error;
   expr;
  }
 }
 async at(*p*3){
  expr;
 }
}

Our goal is to ensure that the finish construct correctly detects termination and never causes a deadlock

EXTENDS *Integers*, *Sequences*

Constants and variables

CONSTANTS
    PLACE,   The set of places
    PROG,   The input program as a sequence of *async* statements
    MXFINISHES,   Maximum finish objects including root and remote
    ROOT_FINISH,   The selected finish implementation for root
    REMOTE_FINISH,   The selected finish implementation for remote
    PROG_HOME,
    NTHREADS,   Minimum number of threads, more threads can be added up to *MXTHREADS* when running threads blocks

    MXTHREADS,
    NBLOCKS,
    MXSTMTS,
    MUST_NOT_RUN   Validation constant: blocks that must not run, for example were not executed because of an execption

VARIABLES
    fstates,   Array of finish states
    msgs,   The set of inflight messages. We delete a message once received
    pstate,   Program state: $init \rightarrow running \rightarrow terminated$
    program,   Finish body as a sequence of statements
    aseq,   Sequence to generate activity ids
    fseq,   Sequence to generate finish ids
    mseq,   Sequence to generate *msg* ids

$readyQ$,	Queue of ready activities at each place
$thrds$,	Threads at all places
$incPar$,	Increase parallelism requests
$decPar$,	Decrease parallelism requests
$ppProgram$,	Preprocessing temporary variable: program
$ppcurStmt$	Preprocessing temporary variable: current statement

$Vars \triangleq \langle fstates, msgs, pstate, program, aseq, fseq, mseq,$
$\quad\quad\quad readyQ, thrds, ppProgram, ppcurStmt, incPar, decPar \rangle$

Predicate to hide the finish implementation

$Finish(fid) \triangleq \text{INSTANCE } AbstractFinish$

INSTANCE $Commons$

$GetRootFinishId(fid) \triangleq$
$\quad$ IF $fid = NoParent$ THEN $NotID$
$\quad$ ELSE IF $Finish(fid)!IsRoot$ THEN fid
$\quad$ ELSE $fstates[fid].root$

Invariants (formulas true in every reachable state.)

$TypeOK \triangleq$
$\quad \wedge \quad fstates \in [IDRange \rightarrow FinishState]$
$\quad \wedge \quad readyQ \in [PLACE \rightarrow Seq(Activity)]$
$\quad \wedge \quad thrds \in [PLACE \rightarrow [ThreadID \rightarrow Thread]]$
$\quad \wedge \quad msgs \subseteq Messages$
$\quad \wedge \quad pstate \in \{ \text{"init"}, \text{"running"}, \text{"terminated"}, \text{"exceptionThrown"} \}$
$\quad \wedge \quad program \in [BlockID \rightarrow Block]$
$\quad \wedge \quad PROG\_HOME \in PLACE$
$\quad \wedge \quad aseq \in Nat$
$\quad \wedge \quad mseq \in Nat$
$\quad \wedge \quad fseq \in IDRange$
$\quad \wedge \quad ppcurStmt \in Nat$
$\quad \wedge \quad incPar \in [PLACE \rightarrow Nat]$
$\quad \wedge \quad decPar \in [PLACE \rightarrow Nat]$
$\quad \wedge \quad MUST\_NOT\_RUN \subseteq BlockID$

$PartialCorrectness \triangleq$
$\quad \vee \quad \wedge pstate = \text{"init"}$
$\quad\quad\quad \wedge \forall p \in PLACE :$
$\quad\quad\quad\quad \wedge readyQ[p] = \langle \rangle$
$\quad\quad\quad\quad \wedge \forall t \in ThreadID : thrds[p][t].stack = \langle \rangle$
$\quad\quad\quad \wedge \forall fid \in IDRange : fstates[fid].status = \text{"unused"}$
$\quad \vee \quad \wedge pstate \in \{ \text{"terminated"}, \text{"exceptionThrown"} \}$

2

$$
\begin{aligned}
&\land ppProgram = \langle\rangle \\
&\land msgs = \{\} \\
&\land \forall p \quad \in PLACE : \\
&\quad \land readyQ[p] = \langle\rangle \\
&\quad \land \forall t \in ThreadID : thrds[p][t].stack = \langle\rangle \\
&\land \forall fid \in IDRange : \\
&\quad \land fstates[fid].status \in \{\text{``unused''}, \text{``forgotten''}\} \\
&\land \text{IF } pstate = \text{``terminated''} \\
&\quad \text{THEN } \land fstates[FIRST\_ID].excs = \langle\rangle \\
&\qquad\qquad \land \forall b \in BlockID : program[b].ran = 1 \\
&\quad \text{ELSE } \land fstates[FIRST\_ID].excs \neq \langle\rangle \\
&\qquad\qquad \land \forall b \in BlockID : \text{IF } b \in MUST\_NOT\_RUN \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{THEN } program[b].ran = 0 \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{ELSE } program[b].ran = 1 \\
\lor \quad &\land pstate = \text{``running''} \\
&\land ppProgram = \langle\rangle \\
&\land \lor \exists p \in PLACE : \\
&\qquad \lor readyQ[p] \neq \langle\rangle \\
&\qquad \lor \exists t \in ThreadID : thrds[p][t].stack \neq \langle\rangle \\
&\quad \lor fstates[FIRST\_ID].status \neq \text{``forgotten''}
\end{aligned}
$$

$CorrectTermination \triangleq$
$\quad \diamond(pstate \in \{\text{``terminated''}, \text{``exceptionThrown''}\})$

Initialization

$Init \triangleq$

$$
\begin{aligned}
&\land fstates \quad = [r \in IDRange \mapsto \\
&\qquad\qquad\qquad [id \mapsto NotID, status \mapsto \text{``unused''}, type \mapsto NotType, \\
&\qquad\qquad\qquad count \mapsto 0, excs \mapsto \langle\rangle, here \mapsto NotPlace, \\
&\qquad\qquad\qquad root \mapsto NotID, remActs \mapsto [p \in PLACE \mapsto 0]]] \\
&\land readyQ \quad = [p \in PLACE \mapsto \langle\rangle] \\
&\land msgs \qquad = \{\} \\
&\land pstate \quad\; = \text{``init''} \\
&\land program = [b \in BlockID \mapsto \\
&\qquad\qquad\qquad [b \mapsto NotBlockID, type \;\mapsto \text{``NA''}, dst \mapsto NotPlace, \\
&\qquad\qquad\qquad mxstmt \mapsto 0, stmts \mapsto [s \in StmtID \mapsto NotBlockID], \\
&\qquad\qquad\qquad ran \mapsto 0]] \\
&\land aseq \qquad = 1 \\
&\land fseq \qquad\; = FIRST\_ID \\
&\land mseq \quad\;\; = 0 \\
&\land ppProgram \;\; = PROG \\
&\land ppcurStmt \;\; = 0 \\
&\land incPar \qquad = [p \in PLACE \mapsto 0] \\
&\land decPar \qquad = [p \in PLACE \mapsto 0]
\end{aligned}
$$

$$\wedge\; thrds \qquad = [p \in PLACE \mapsto$$
$$[t \in ThreadID \mapsto$$
$$\text{IF } t < NTHREADS$$
$$\text{THEN } [tid \mapsto t,\; status \mapsto \text{"idle"},\; stack \mapsto \langle\rangle]$$
$$\text{ELSE } [tid \mapsto t,\; status \mapsto \text{"NA"},\; stack \mapsto \langle\rangle]]]$$

Parsing the input program into another format for easier processing

$ParseInputProgram \;\triangleq$
$\quad \wedge\; pstate = \text{"init"}$
$\quad \wedge\; Len(ppProgram) > 0$
$\quad \wedge\; \text{LET } curBlk \;\triangleq\; Head(ppProgram)$
$\qquad\qquad body \;\triangleq\; curBlk.body$
$\qquad\qquad t \quad\triangleq\; curBlk.type$
$\qquad\qquad d \quad\triangleq\; curBlk.dst$
$\qquad\qquad b \quad\triangleq\; curBlk.b$
$\qquad\qquad h \quad\triangleq\; \text{IF } body = \langle\rangle \text{ THEN } EMPTY\_BLOCK \text{ ELSE } Head(body)$
$\qquad \text{IN} \qquad \wedge\; program' = [program \text{ EXCEPT } ![b].b = b,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad ![b].type \quad = t,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad ![b].dst = d,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad ![b].mxstmt = ppcurStmt,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad ![b].ran = 0,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad ![b].stmts[ppcurStmt] = h]$
$\qquad\qquad\quad \wedge\; \text{IF } ((Len(body) = 0 \wedge ppcurStmt = 0) \vee Len(body) = 1)$
$\qquad\qquad\qquad \text{THEN } \wedge\; ppcurStmt' = 0$
$\qquad\qquad\qquad\qquad\quad \wedge\; ppProgram' = Tail(ppProgram)$
$\qquad\qquad\qquad \text{ELSE } \wedge\; ppcurStmt' = ppcurStmt + 1$
$\qquad\qquad\qquad\qquad\quad \wedge\; ppProgram' = \langle[type \mapsto t,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad dst \mapsto d,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad b \mapsto b,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad body \mapsto Tail(body),$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad err \mapsto \text{""}]$
$\qquad\qquad\qquad\qquad\qquad\quad \rangle \circ Tail(ppProgram)$
$\quad \wedge\; \text{UNCHANGED } \langle fstates,\; pstate,\; msgs,\; aseq,\; fseq,\; mseq,\; readyQ,$
$\qquad\qquad\qquad\qquad\qquad thrds,\; incPar,\; decPar\rangle$

Start program execution (*i.e.* simulate $X10$'s main method)

$Run \;\triangleq$
$\quad \wedge\; pstate = \text{"init"}$
$\quad \wedge\; Len(ppProgram) = 0$
$\quad \wedge\; pstate' = \text{"running"}$
$\quad \wedge\; \text{LET } curStmt \;\triangleq\; \text{IF } program[0].type = \text{"finish"} \text{ THEN } -2 \text{ ELSE } -1$
$\qquad \text{IN} \quad thrds' = [thrds \text{ EXCEPT } ![PROG\_HOME][0].stack =$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \langle[\; b \;\mapsto 0,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad i \;\mapsto curStmt,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad fid \mapsto NoParent]$

$$
\rangle,
$$
$$
\quad! \quad [PROG\_HOME][0].status = \text{``running''}\,]
$$
$$
\land\ program' = [program \text{ EXCEPT } ![0].ran = 1]
$$
$$
\land\ \text{UNCHANGED } \langle fstates,\ msgs,\ aseq,\ fseq,\ mseq,\ readyQ,\ ppProgram,
$$
$$
ppcurStmt,\ incPar,\ decPar \rangle
$$

Scheduler Actions

Helper action: push activity
$PushReadyFIFO(here,\ activity) \triangleq$
$\quad \land\ readyQ' = [readyQ \text{ EXCEPT } ![here] = Append(@,\ activity)]$

Helper action: poll activity
$PollReadyFIFO(here) \triangleq$
$\quad \land\ readyQ[here] \neq \langle \rangle$
$\quad \land\ readyQ' = [readyQ \text{ EXCEPT } ![here] = Tail(readyQ[here])]$

Push an activity received from another place
$SubmitRemoteActivity(here,\ src,\ activity) \triangleq$
$\quad \land\ \text{IF } activity.fid \neq NoParent$
$\quad\quad \text{THEN } Finish(activity.fid)!NotifyActivityCreation(src,\ activity)$
$\quad\quad \text{ELSE } fstates' = fstates$
$\quad \land\ PushReadyFIFO(here,\ activity)$

Push a local activity
$SubmitLocalActivity(here,\ activity) \triangleq$
$\quad \land\ \text{IF } activity.fid \neq NoParent$
$\quad\quad \text{THEN } Finish(activity.fid)!NotifyActivitySpawnAndCreation(here,\ activity)$
$\quad\quad \text{ELSE } fstates' = fstates$
$\quad \land\ PushReadyFIFO(here,\ activity)$

Increase the number of worker threads
$IncreaseParallelism(here) \triangleq$
$\quad \land\ pstate = \text{``running''}$
$\quad \land\ incPar[here] > 0$
$\quad \land\ \text{LET } tid \triangleq FindThread(here,\ \text{``NA''})$
$\quad\quad \text{IN} \quad \land\ tid \neq NotThreadID$
$\quad\quad\quad\quad \land\ incPar' = [incPar \text{ EXCEPT } ![here] = @ - 1]$
$\quad\quad\quad\quad \land\ thrds' = [thrds \text{ EXCEPT } ![here][tid].status = \text{``idle''}]$
$\quad \land\ \text{UNCHANGED } \langle fstates,\ msgs,\ pstate,\ program,\ aseq,\ fseq,\ mseq,\ readyQ,$
$\quad\quad\quad\quad ppProgram,\ ppcurStmt,\ decPar \rangle$

Decrease the number of worker threads
$DecreaseParallelism(here) \triangleq$
$\quad \land\ pstate = \text{``running''}$
$\quad \land\ decPar[here] > 0$

$\land$ LET $tid \triangleq FindThread(here,$ "idle"$)$
   IN $\land tid \neq NotThreadID$
        $\land decPar' = [decPar \text{ EXCEPT } ![here] = @ - 1]$
        $\land thrds' = [thrds \text{ EXCEPT } ![here][tid].status =$ "NA"$]$
$\land$ UNCHANGED $\langle fstates, msgs, pstate, program, aseq, fseq, mseq, readyQ,$
                  $ppProgram, ppcurStmt, incPar\rangle$

Idle thread fetching a ready activity

$IThreadFetchActivity(here, tid) \triangleq$
    $\land pstate =$ "running"
    $\land thrds[here][tid].status =$ "idle"
    $\land PollReadyFIFO(here)$
    $\land$ LET $act \triangleq Head(readyQ[here])$
            $stkEntry \triangleq [b \mapsto act.b, i \mapsto -1, fid \mapsto act.fid]$
       IN $\land thrds' = [thrds \text{ EXCEPT } ![here][tid].stack = \langle stkEntry \rangle,$
                             $![here][tid].status =$ "running"$]$
            $\land program' = [program \text{ EXCEPT } ![act.b].ran = 1]$
    $\land$ UNCHANGED $\langle fstates, msgs, pstate, aseq, fseq, mseq, ppProgram,$
                  $ppcurStmt, incPar, decPar\rangle$

Running thread processing an expression

$RThreadRunExpr(here, tid) \triangleq$
    $\land pstate =$ "running"
    $\land thrds[here][tid].status =$ "running"
    $\land$ LET $top \triangleq Head(thrds[here][tid].stack)$
          $blk \triangleq top.b$
       IN $\land program[blk].type =$ "expr"
           $\land$ IF $Len(thrds[here][tid].stack) = 1$
              THEN $thrds' = [thrds \text{ EXCEPT } ![here][tid].stack = \langle \rangle,$
                                   $![here][tid].status =$ "idle"$]$
              ELSE $thrds' = [thrds \text{ EXCEPT } ![here][tid].stack = Tail(@)]$
    $\land$ UNCHANGED $\langle fstates, msgs, pstate, program, aseq, fseq, mseq, readyQ,$
                  $ppProgram, ppcurStmt, incPar, decPar\rangle$

Running thread processing the end of an *async* block

$RThreadRunAsyncEnd(here, tid) \triangleq$
    $\land pstate =$ "running"
    $\land thrds[here][tid].status =$ "running"
    $\land$ LET $top \triangleq Head(thrds[here][tid].stack)$
          $blk \triangleq top.b$
          $fid \triangleq top.fid$
       IN $\land program[blk].type =$ "async"
           $\land program[blk].mxstmt = top.i$
           $\land Finish(fid)!NotifyActivityTermination$

6

$\wedge$ IF $\quad Len(thrds[here][tid].stack) = 1$
$\quad$ THEN $\quad thrds' = [thrds$ EXCEPT $![here][tid].stack = \langle\rangle,$
$\qquad\qquad\qquad\qquad\qquad\qquad ![here][tid].status = \text{"idle"}]$
$\quad$ ELSE $\quad thrds' = [thrds$ EXCEPT $![here][tid].stack = Tail(@)]$
$\wedge$ IF $\quad blk = 0$
$\quad$ THEN $pstate' = \text{"terminated"}$
$\quad$ ELSE $\quad pstate' = pstate$
$\wedge$ UNCHANGED $\langle msgs, pstate, program, aseq, fseq, mseq, readyQ, ppProgram,$
$\qquad\qquad\qquad ppcurStmt, incPar, decPar\rangle$

Running thread processing the end of a finish block and blocking itself

$RThreadRunFinishEnd(here, tid) \triangleq$
$\quad \wedge pstate = \text{"running"}$
$\quad \wedge thrds[here][tid].status = \text{"running"}$
$\quad \wedge$ LET $top \triangleq Head(thrds[here][tid].stack)$
$\qquad$ IN $\quad \wedge program[top.b].type = \text{"finish"}$
$\qquad\qquad \wedge program[top.b].mxstmt = top.i$
$\qquad\qquad \wedge Finish(top.fid)!NotifyActivityTermination$
$\qquad\qquad \wedge thrds' = [thrds$ EXCEPT $![here][tid].status = \text{"blocked"}]$
$\qquad\qquad \wedge incPar' = [incPar$ EXCEPT $![here] = @ + 1]$
$\quad \wedge$ UNCHANGED $\langle msgs, pstate, program, aseq, fseq, mseq, readyQ, ppProgram,$
$\qquad\qquad\qquad ppcurStmt, decPar\rangle$

Terminated finish unblocks its thread

$BThreadUnblock(here, tid) \triangleq$
$\quad \wedge pstate = \text{"running"}$
$\quad \wedge thrds[here][tid].status = \text{"blocked"}$
$\quad \wedge$ LET $top \triangleq Head(thrds[here][tid].stack)$
$\qquad\qquad blk \triangleq top.b$
$\qquad\qquad fid \triangleq top.fid$
$\qquad$ IN $\quad \wedge program[blk].type = \text{"finish"}$
$\qquad\qquad \wedge program[blk].mxstmt = top.i$
$\qquad\qquad \wedge Finish(fid)!Terminated$
$\qquad\qquad \wedge decPar' = [decPar$ EXCEPT $![here] = @ + 1]$
$\qquad\qquad \wedge$ IF $\quad Len(thrds[here][tid].stack) = 1$
$\qquad\qquad\quad$ THEN $\quad thrds' = [thrds$ EXCEPT $![here][tid].stack = \langle\rangle,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![here][tid].status = \text{"idle"}]$
$\qquad\qquad\quad$ ELSE $\quad thrds' = [thrds$ EXCEPT $![here][tid].stack = Tail(@),$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad ![here][tid].status = \text{"running"}]$
$\qquad\qquad \wedge$ IF $\quad blk = 0 \wedge Finish(fid)!HasExceptions$
$\qquad\qquad\quad$ THEN $pstate' = \text{"exceptionThrown"}$
$\qquad\qquad\quad$ ELSE IF $blk = 0 \wedge \neg Finish(fid)!HasExceptions$
$\qquad\qquad\quad$ THEN $pstate' = \text{"terminated"}$
$\qquad\qquad\quad$ ELSE $\quad pstate' = pstate$
$\quad \wedge$ UNCHANGED $\langle fstates, msgs, program, aseq, fseq, mseq, readyQ,$

7

$$ppProgram, \ ppcurStmt, \ incPar\rangle$$

$RThreadRunFinishFirstStmt(here, \ tid) \ \triangleq$

$\quad \wedge pstate = \text{"running"}$

$\quad \wedge thrds[here][tid].status = \text{"running"}$

$\quad \wedge \text{LET} \ top \ \triangleq \ Head(thrds[here][tid].stack)$

$\qquad\qquad tail \ \triangleq \ Tail(thrds[here][tid].stack)$

$\qquad\qquad lstStmt \ \triangleq \ top.i$

$\qquad\qquad curStmt \ \triangleq \ top.i + 1$

$\qquad\qquad blk \ \triangleq \ top.b$

$\qquad\qquad fid \ \triangleq \ top.fid$

$\quad\quad\text{IN} \qquad \wedge program[blk].type = \text{"finish"}$

$\qquad\qquad \wedge lstStmt = -2$

$\qquad\qquad \wedge Finish(fseq)!Alloc(ROOT\_FINISH, \ here, \ fid)$

$\qquad\qquad \wedge thrds' = [thrds \ \text{EXCEPT} \ ![here][tid].stack = \langle[b \ \mapsto \ top.b,$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad i \ \mapsto \ curStmt,$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad fid \mapsto fseq]$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \rangle \circ tail]$

$\qquad\qquad \wedge fseq' = fseq + 1$

$\quad \wedge \text{UNCHANGED} \ \langle msgs, \ pstate, \ program, \ aseq, \ mseq, \ readyQ, \ ppProgram,$

$\qquad\qquad\qquad\qquad\qquad ppcurStmt, \ incPar, \ decPar\rangle$

$RThreadRunNestedLocalAsync(here, \ tid) \ \triangleq$

$\quad \wedge pstate = \text{"running"}$

$\quad \wedge thrds[here][tid].status = \text{"running"}$

$\quad \wedge \text{LET} \ top \ \triangleq \ Head(thrds[here][tid].stack)$

$\qquad\qquad tail \ \triangleq \ Tail(thrds[here][tid].stack)$

$\qquad\qquad lstStmt \ \triangleq \ top.i$

$\qquad\qquad curStmt \ \triangleq \ top.i + 1$

$\qquad\qquad blk \ \triangleq \ top.b$

$\qquad\qquad fid \ \triangleq \ top.fid$

$\qquad\qquad nested \ \triangleq \ program[blk].stmts[curStmt]$

$\quad\quad\text{IN} \qquad \wedge program[blk].type \neq \text{"expr"}$

$\qquad\qquad \wedge curStmt \geq 0$

$\qquad\qquad \wedge curStmt \leq program[blk].mxstmt$

$\qquad\qquad \wedge program[nested].type = \text{"async"}$

$\qquad\qquad \wedge program[nested].dst = here$

$\qquad\qquad \wedge SubmitLocalActivity(here, \ [aid \mapsto aseq,$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad b \mapsto nested,$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad fid \ \mapsto fid])$

$\qquad\qquad \wedge aseq' = aseq + 1$

$\qquad\qquad \wedge thrds' = [thrds \ \text{EXCEPT} \ ![here][tid].stack =$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \langle[\ \ b \ \mapsto \ top.b,$

$$
\begin{aligned}
&\qquad\qquad\qquad\qquad\qquad\quad i \;\mapsto curStmt, \\
&\qquad\qquad\qquad\qquad\qquad\quad \mathit{fid} \mapsto \mathit{fid}] \\
&\qquad\qquad\qquad\qquad\quad \rangle \circ tail]
\end{aligned}
$$

$\land$ UNCHANGED $\langle msgs,\ pstate,\ program,\ fseq,\ mseq,\ ppProgram,$
$\qquad\qquad\qquad\quad ppcurStmt,\ incPar,\ decPar\rangle$

Processing a nested remote *async* in the currently running block

$RThreadRunNestedRemoteAsync(here,\ tid) \;\triangleq$

$\quad \land\ pstate =$ "running"

$\quad \land\ thrds[here][tid].status =$ "running"

$\quad \land$ LET $top \;\overset{\Delta}{=}\; Head(thrds[here][tid].stack)$

$\qquad\qquad tail \;\overset{\Delta}{=}\; Tail(thrds[here][tid].stack)$

$\qquad\qquad lstStmt \;\overset{\Delta}{=}\; top.i$

$\qquad\qquad curStmt \;\overset{\Delta}{=}\; top.i + 1$

$\qquad\qquad blk \;\overset{\Delta}{=}\; top.b$

$\qquad\qquad \mathit{fid} \;\overset{\Delta}{=}\; top.\mathit{fid}$

$\qquad\qquad root \;\overset{\Delta}{=}\; GetRootFinishId(\mathit{fid})$

$\qquad\qquad nested \;\overset{\Delta}{=}\; program[blk].stmts[curStmt]$

$\quad$ IN $\quad \land\ program[blk].type \neq$ "expr"

$\qquad\qquad \land\ \mathit{fid} \neq NoParent$

$\qquad\qquad \land\ curStmt \geq 0$

$\qquad\qquad \land\ curStmt \leq program[blk].mxstmt$

$\qquad\qquad \land\ program[nested].type =$ "async"

$\qquad\qquad \land\ program[nested].dst \neq here$

$\qquad\qquad \land\ \lor\ \land\ Finish(\mathit{fid})!NotifySubActivitySpawn(program[nested].dst)$

$\qquad\qquad\qquad\quad \land\ SendMsg([mid \mapsto mseq,$
$\qquad\qquad\qquad\qquad\qquad\qquad src \;\mapsto here,$
$\qquad\qquad\qquad\qquad\qquad\qquad dst \;\mapsto program[nested].dst,$
$\qquad\qquad\qquad\qquad\qquad\qquad type \mapsto$ "async",
$\qquad\qquad\qquad\qquad\qquad\qquad \mathit{fid} \mapsto root,$
$\qquad\qquad\qquad\qquad\qquad\qquad b \mapsto nested])$

$\qquad\qquad\qquad\quad \land\ mseq' = mseq + 1$

$\qquad\qquad\qquad\quad \land\ thrds' = [thrds$ EXCEPT $![here][tid].stack =$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \langle[\ b \;\mapsto top.b,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad i \;\mapsto curStmt,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{fid} \mapsto \mathit{fid}]$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \rangle \circ tail]$

$\qquad\qquad\quad \lor\ \land\ Finish(\mathit{fid})!NotifySubActivitySpawnError(program[nested].dst)$

$\qquad\qquad\qquad\quad \land\ msgs' = msgs$

$\qquad\qquad\qquad\quad \land\ mseq' = mseq$

$\qquad\qquad\qquad\quad \land\ thrds' = [thrds$ EXCEPT $![here][tid].stack =$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \langle[b \;\mapsto top.b,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad i \;\mapsto program[blk].mxstmt,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{fid} \mapsto \mathit{fid}]$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \rangle \circ tail]$

$\wedge$ UNCHANGED $\langle pstate,\ program,\ aseq,\ fseq,\ readyQ,\ ppProgram,$
$\qquad\qquad\qquad ppcurStmt,\ incPar,\ decPar \rangle$

Processing a nested finish in the currently running block
$RThreadRunNestedFinish(here,\ tid) \triangleq$
   $\wedge\ pstate =$ "running"
   $\wedge\ thrds[here][tid].status =$ "running"
   $\wedge$ LET $top \triangleq Head(thrds[here][tid].stack)$
      $tail \triangleq Tail(thrds[here][tid].stack)$
      $lstStmt \triangleq top.i$
      $curStmt \triangleq top.i + 1$
      $blk \triangleq top.b$
      $fid \triangleq top.fid$
      $nested \triangleq program[blk].stmts[curStmt]$
     IN $\wedge\ program[blk].type \neq$ "expr"
        $\wedge\ curStmt \geq 0$
        $\wedge\ curStmt \leq program[blk].mxstmt$
        $\wedge\ program[nested].type =$ "finish"
        $\wedge\ program[nested].dst = here$
        $\wedge\ thrds' = [thrds$ EXCEPT $![here][tid].stack =$
                $\langle [\ \ b\ \mapsto nested,$
                  $i\ \mapsto\ -2,$
                  $fid \mapsto fid],$
                $[\ \ b\ \mapsto top.b,$
                  $i\ \mapsto curStmt,$
                  $fid \mapsto fid]$
               $\rangle \circ tail]$
        $\wedge\ program' = [program$ EXCEPT $![nested].ran = 1]$
   $\wedge$ UNCHANGED $\langle fstates,\ msgs,\ pstate,\ aseq,\ fseq,\ mseq,\ readyQ,$
         $ppProgram,\ ppcurStmt,\ incPar,\ decPar \rangle$

Processing a nested expression in the currently running block
$RThreadRunNestedExpr(here,\ tid) \triangleq$
   $\wedge\ pstate =$ "running"
   $\wedge\ thrds[here][tid].status =$ "running"
   $\wedge$ LET $top \triangleq Head(thrds[here][tid].stack)$
      $tail \triangleq Tail(thrds[here][tid].stack)$
      $lstStmt \triangleq top.i$
      $curStmt \triangleq top.i + 1$
      $blk \triangleq top.b$
      $fid \triangleq top.fid$
      $nested \triangleq program[blk].stmts[curStmt]$
     IN $\wedge\ program[blk].type \neq$ "expr"
        $\wedge\ curStmt \geq 0$
        $\wedge\ curStmt \leq program[blk].mxstmt$

$\land\ program[nested].type =\ \text{"expr"}$
$\land\ program[nested].dst = here$
$\land\ thrds' = [thrds\ \text{EXCEPT}\ ![here][tid].stack =$
$$\langle[\ \ b\ \ \mapsto nested,$$
$$i\ \ \mapsto -1,$$
$$\mathit{fid} \mapsto \mathit{fid}],$$
$$[\ \ b\ \ \mapsto top.b,$$
$$i\ \ \mapsto curStmt,$$
$$\mathit{fid} \mapsto \mathit{fid}]$$
$$\rangle \circ tail]$$
$\land\ program' = [program\ \text{EXCEPT}\ ![nested].ran = 1]$
$\land\ \text{UNCHANGED}\ \langle fstates,\ msgs,\ pstate,\ aseq,\ fseq,\ mseq,\ readyQ,$
$$ppProgram,\ ppcurStmt,\ incPar,\ decPar\rangle$$

$RThreadRunNestedError(here,\ tid)\ \triangleq$
$\quad \land\ pstate =\ \text{"running"}$
$\quad \land\ thrds[here][tid].status =\ \text{"running"}$
$\quad \land\ \text{LET}\ top\ \ \triangleq\ Head(thrds[here][tid].stack)$
$\qquad\qquad tail\ \triangleq\ Tail(thrds[here][tid].stack)$
$\qquad\qquad lstStmt\ \ \triangleq\ top.i$
$\qquad\qquad curStmt\ \triangleq\ top.i + 1$
$\qquad\qquad blk\ \triangleq\ top.b$
$\qquad\qquad \mathit{fid}\ \triangleq\ top.\mathit{fid}$
$\qquad\qquad nested\ \triangleq\ program[blk].stmts[curStmt]$
$\quad\ \ \text{IN}\quad \land\ program[blk].type \notin \{\ \text{"expr"},\ \text{"error"}\ \}$
$\qquad\qquad \land\ curStmt \geq 0$
$\qquad\qquad \land\ curStmt \leq program[blk].mxstmt$
$\qquad\qquad \land\ program[nested].type =\ \text{"error"}$
$\qquad\qquad \land\ program[nested].dst = here$
$\qquad\qquad \land\ thrds' = [thrds\ \text{EXCEPT}\ ![here][tid].stack =\ $ jump to the end of the current block
$$\langle[\ \ b\ \ \mapsto top.b,$$
$$i\ \ \mapsto program[blk].mxstmt,$$
$$\mathit{fid} \mapsto \mathit{fid}]$$
$$\rangle \circ tail]$$
$\qquad\qquad \land\ program' = [program\ \text{EXCEPT}\ ![nested].ran = 1]$
$\qquad\qquad \land\ Finish(\mathit{fid})!PushException([err \mapsto\ \text{"ErrorStmt"},\ from \mapsto here])$
$\quad \land\ \text{UNCHANGED}\ \langle msgs,\ pstate,\ aseq,\ fseq,\ mseq,\ readyQ,$
$$ppProgram,\ ppcurStmt,\ incPar,\ decPar\rangle$$

$CreateRemoteFinish(here)\ \triangleq$
$\quad \land\ pstate =\ \text{"running"}$
$\quad \land\ \text{LET}\ msg\ \triangleq\ FindIncomingMSG(here,\ \text{"async"})$
$\qquad\qquad pid\ \triangleq\ msg.\mathit{fid}$

11

$$
\begin{aligned}
&\qquad\qquad \mathit{fid} \;\;\triangleq\;\; \mathit{GetActiveFID}(\mathit{REMOTE\_FINISH},\, \mathit{here},\, \mathit{pid}) \\
&\quad \text{IN} \quad \wedge\, \mathit{pid} \neq \mathit{NotID} \\
&\qquad\qquad \wedge\, \mathit{fid} \,=\, \mathit{NotID} \\
&\qquad\qquad \wedge\, \mathit{Finish}(\mathit{fseq})!\mathit{Alloc}(\mathit{REMOTE\_FINISH},\, \mathit{here},\, \mathit{pid}) \\
&\qquad\qquad \wedge\, \mathit{fseq}' = \mathit{fseq} + 1 \\
&\quad \wedge\, \text{UNCHANGED}\; \langle \mathit{msgs},\, \mathit{pstate},\, \mathit{program},\, \mathit{aseq},\, \mathit{mseq},\, \mathit{readyQ},\, \mathit{thrds},\, \mathit{ppProgram}, \\
&\qquad\qquad\qquad\qquad\qquad \mathit{ppcurStmt},\, \mathit{incPar},\, \mathit{decPar} \rangle
\end{aligned}
$$

Pushing an incoming *async* to the ready queue

$RecvAsync(here) \;\triangleq\;$

$$
\begin{aligned}
&\quad \wedge\, \mathit{pstate} = \text{``running''} \\
&\quad \wedge\, \text{LET}\; \mathit{msg} \;\triangleq\; \mathit{FindIncomingMSG}(\mathit{here},\, \text{``async''}) \\
&\qquad\qquad\;\; \mathit{pid} \;\triangleq\; \mathit{msg.fid} \\
&\qquad\qquad\;\; \mathit{fid} \;\triangleq\; \mathit{GetActiveFID}(\mathit{REMOTE\_FINISH},\, \mathit{here},\, \mathit{pid}) \\
&\qquad\qquad\;\; \mathit{src} \;\triangleq\; \mathit{msg.src} \\
&\qquad\qquad\;\; \mathit{blk} \;\triangleq\; \mathit{msg.b} \\
&\qquad \text{IN} \quad \wedge\, \mathit{pid} \neq \mathit{NotID} \\
&\qquad\qquad\;\; \wedge\, \mathit{fid} \,\neq\, \mathit{NotID} \\
&\qquad\qquad\;\; \wedge\, \mathit{src} \,\neq\, \mathit{NotPlace} \\
&\qquad\qquad\;\; \wedge\, \mathit{RecvMsg}([\mathit{mid} \mapsto \mathit{msg.mid}, \\
&\qquad\qquad\qquad\qquad\qquad \mathit{src} \;\mapsto\, \mathit{msg.src}, \\
&\qquad\qquad\qquad\qquad\qquad \mathit{dst} \;\mapsto\, \mathit{here}, \\
&\qquad\qquad\qquad\qquad\qquad \mathit{type} \mapsto \text{``async''}, \\
&\qquad\qquad\qquad\qquad\qquad b \mapsto \mathit{blk}, \\
&\qquad\qquad\qquad\qquad\qquad \mathit{fid} \mapsto \mathit{pid}]) \\
&\qquad\qquad\;\; \wedge\, \mathit{SubmitRemoteActivity}(\mathit{here},\, \mathit{src},\, [\mathit{aid} \mapsto \mathit{aseq}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad b \,\mapsto\, \mathit{blk}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\;\; \mathit{fid} \,\mapsto\, \mathit{fid}]) \\
&\qquad\qquad\;\; \wedge\, \mathit{aseq}' = \mathit{aseq} + 1 \\
&\quad \wedge\, \text{UNCHANGED}\; \langle \mathit{pstate},\, \mathit{program},\, \mathit{fseq},\, \mathit{mseq},\, \mathit{thrds},\, \mathit{ppProgram},\, \mathit{ppcurStmt}, \\
&\qquad\qquad\qquad\qquad\qquad \mathit{incPar},\, \mathit{decPar} \rangle
\end{aligned}
$$

Enclosing finish receiving a termination signal from a remote task

$RecvAsyncTerm(here) \;\triangleq\;$

$$
\begin{aligned}
&\quad \wedge\, \mathit{pstate} = \text{``running''} \\
&\quad \wedge\, \text{LET}\; \mathit{msg} \;\triangleq\; \mathit{FindIncomingMSG}(\mathit{here},\, \text{``asyncTerm''}) \\
&\qquad\qquad\;\; \mathit{fid} \;\triangleq\; \mathit{msg.fid} \\
&\qquad\qquad\;\; \mathit{src} \;\triangleq\; \mathit{msg.src} \\
&\qquad \text{IN} \quad \wedge\, \mathit{fid} \,\neq\, \mathit{NotID} \\
&\qquad\qquad\;\; \wedge\, \mathit{src} \,\neq\, \mathit{NotPlace} \\
&\qquad\qquad\;\; \wedge\, \mathit{Finish}(\mathit{fid})!\mathit{ProcessChildTermMsg}(\mathit{msg}) \\
&\quad \wedge\, \text{UNCHANGED}\; \langle \mathit{pstate},\, \mathit{program},\, \mathit{aseq},\, \mathit{fseq},\, \mathit{mseq},\, \mathit{readyQ},\, \mathit{thrds},\, \mathit{ppProgram}, \\
&\qquad\qquad\qquad\qquad\qquad \mathit{ppcurStmt},\, \mathit{incPar},\, \mathit{decPar} \rangle
\end{aligned}
$$

RemoteFinish notifying its *RootFinish* that it terminated

$NotifyParentFinish(fid) \;\triangleq\;$

$\wedge$ $pstate =$ "running"
$\wedge$ $fstates[fid].status =$ "finished"
$\wedge$ LET $type \triangleq fstates[fid].type$
　　　$pid \triangleq fstates[fid].root$
　　　$pidHome \triangleq GetFinishHome(pid)$
　　　$here \triangleq fstates[fid].here$
　IN　　IF $type = ROOT\_FINISH$
　　　THEN　$\wedge$ $msgs' = msgs$
　　　　　　$\wedge$ $mseq' = mseq$
　　　　　　$\wedge$ $fstates' = [fstates$ EXCEPT $![fid].status =$ "forgotten"$]$
　　　ELSE　$\wedge$ $pidHome \neq here$
　　　　　　$\wedge$ $Finish(fid)!SendTermMsg(mseq)$
　　　　　　$\wedge$ $mseq' = mseq + 1$
$\wedge$ UNCHANGED $\langle program, pstate, aseq, fseq, readyQ, thrds, ppProgram,$
　　　　　　　　　$ppcurStmt, incPar, decPar\rangle$

Predicate enumerating all possible next actions

$Next \triangleq$
　$\vee$ $ParseInputProgram$
　$\vee$ Run
　$\vee$ $\exists\, here \in PLACE :$
　　　$\vee$ $IncreaseParallelism(here)$
　　　$\vee$ $DecreaseParallelism(here)$
　　　$\vee$ $CreateRemoteFinish(here)$
　　　$\vee$ $RecvAsync(here)$
　　　$\vee$ $RecvAsyncTerm(here)$
　　　$\vee$ $\exists\, tid \in ThreadID :$
　　　　　$\vee$ $IThreadFetchActivity(here, tid)$
　　　　　$\vee$ $RThreadRunExpr(here, tid)$
　　　　　$\vee$ $RThreadRunAsyncEnd(here, tid)$
　　　　　$\vee$ $RThreadRunFinishEnd(here, tid)$
　　　　　$\vee$ $BThreadUnblock(here, tid)$
　　　　　$\vee$ $RThreadRunFinishFirstStmt(here, tid)$
　　　　　$\vee$ $RThreadRunNestedLocalAsync(here, tid)$
　　　　　$\vee$ $RThreadRunNestedRemoteAsync(here, tid)$
　　　　　$\vee$ $RThreadRunNestedFinish(here, tid)$
　　　　　$\vee$ $RThreadRunNestedExpr(here, tid)$
　　　　　$\vee$ $RThreadRunNestedError(here, tid)$
　　$\vee$ $\exists\, fid \in IDRange :$
　　　　$\vee$ $NotifyParentFinish(fid)$

Asserting fairness properties to all actions

13

$Liveness \triangleq$
  $\wedge \text{WF}_{Vars}(ParseInputProgram)$
  $\wedge \text{WF}_{Vars}(Run)$
  $\wedge \forall\, here \in PLACE :$
      $\text{WF}_{Vars}(IncreaseParallelism(here))$
    $\wedge \text{WF}_{Vars}(DecreaseParallelism(here))$
    $\wedge \text{WF}_{Vars}(CreateRemoteFinish(here))$
    $\wedge \text{WF}_{Vars}(RecvAsync(here))$
    $\wedge \text{WF}_{Vars}(RecvAsyncTerm(here))$
    $\wedge \forall\, tid \in ThreadID :$
        $\wedge \text{WF}_{Vars}(IThreadFetchActivity(here,\ tid))$
        $\wedge \text{WF}_{Vars}(RThreadRunExpr(here,\ tid))$
        $\wedge \text{WF}_{Vars}(RThreadRunAsyncEnd(here,\ tid))$
        $\wedge \text{WF}_{Vars}(RThreadRunFinishEnd(here,\ tid))$
        $\wedge \text{WF}_{Vars}(BThreadUnblock(here,\ tid))$
        $\wedge \text{WF}_{Vars}(RThreadRunFinishFirstStmt(here,\ tid))$
        $\wedge \text{WF}_{Vars}(RThreadRunNestedLocalAsync(here,\ tid))$
        $\wedge \text{WF}_{Vars}(RThreadRunNestedRemoteAsync(here,\ tid))$
        $\wedge \text{WF}_{Vars}(RThreadRunNestedFinish(here,\ tid))$
        $\wedge \text{WF}_{Vars}(RThreadRunNestedExpr(here,\ tid))$
        $\wedge \text{WF}_{Vars}(RThreadRunNestedError(here,\ tid))$
  $\wedge \forall\, fid \in IDRange :$
      $\text{WF}_{Vars}(NotifyParentFinish(fid))$

Specification

$Spec \triangleq\ \ Init \wedge \square[Next]_{Vars} \wedge Liveness$

THEOREM $Spec \Rightarrow \square(TypeOK \wedge PartialCorrectness)$