MODULE *OptimisticCommons*

Constants and common utility actions

EXTENDS *Integers*

CONSTANTS *LEVEL*,
WIDTH,
NUM_PLACES,
MAX_KILL

VARIABLES *exec_state*,
tasks,
f_set,
lf_set,
rf_set,
msgs,
nxt_finish_id,
nxt_task_id,
nxt_remote_place,
killed,
rec_child,
rec_to,
rec_from,
rec_from_waiting

$FIRST\_PLACE\_ID \triangleq 0$
$PlaceID \triangleq FIRST\_PLACE\_ID \,..\, (NUM\_PLACES - 1)$
$NOT\_PLACE\_ID \triangleq -1$

$ROOT\_FINISH\_ID \triangleq 0$
$MAX\_FINISH\_ID \triangleq ((1 - (WIDTH)^{(LEVEL+1)}) \div (1 - WIDTH))$ the power series
$NOT\_FINISH\_ID \triangleq -1$
$FinishID \triangleq ROOT\_FINISH\_ID \,..\, MAX\_FINISH\_ID$

$ROOT\_TASK\_ID \triangleq 0$
$MAX\_TASK\_ID \triangleq MAX\_FINISH\_ID$
$NOT\_TASK\_ID \triangleq -1$
$TaskID \triangleq ROOT\_TASK\_ID \,..\, MAX\_TASK\_ID$

$BranchID \triangleq 0 \,..\, WIDTH$
$LevelID \triangleq 0 \,..\, LEVEL$

$TASK\_STATUS \triangleq \{$"waitingForPublish", "waitingForTransit", "sent", "dropped", "running", "blocked", "termi

$TASK\_TYPE \triangleq \{$"normal", terminates at any time
"finishMainTask" terminates after finish creates all its branches
$\}$

$FINISH\_STATUS \triangleq \{\text{“active”}, \text{“waitingForPublish”}, \text{“waitingForRelease”}, \text{“released”}\}$

$$
\begin{aligned}
Task \triangleq\ & [id : TaskID, \\
& pred\_id : TaskID \cup \{NOT\_TASK\_ID\}, \quad \text{predecessor task, used for debugging only} \\
& src : PlaceID,\ dst : PlaceID, \\
& finish\_id : FinishID, \\
& level : LevelID, \\
& last\_branch : BranchID, \\
& status : TASK\_STATUS, \\
& type : TASK\_TYPE, \\
& finish\_type : \{\text{“global”}, \text{“local”}, \text{“N/A”}\}]
\end{aligned}
$$

$$
\begin{aligned}
RootTask \triangleq\ & [id \mapsto ROOT\_TASK\_ID, \\
& pred\_id \mapsto NOT\_TASK\_ID, \\
& src \mapsto FIRST\_PLACE\_ID, \\
& dst \mapsto FIRST\_PLACE\_ID, \\
& finish\_id \mapsto ROOT\_FINISH\_ID, \\
& level \mapsto 0, \\
& last\_branch \mapsto WIDTH, \\
& status \mapsto \text{“blocked”}, \\
& type \mapsto \text{“normal”}, \\
& finish\_type \mapsto \text{“global”}]
\end{aligned}
$$

$$
\begin{aligned}
NOT\_TASK \triangleq\ & [id \mapsto NOT\_TASK\_ID, \\
& src \mapsto NOT\_PLACE\_ID, \\
& dst \mapsto NOT\_PLACE\_ID, \\
& level \mapsto -1, \\
& finish\_id \mapsto NOT\_FINISH\_ID, \\
& finish\_type \mapsto \text{“N/A”}]
\end{aligned}
$$

$$Place1D \triangleq [PlaceID \to Nat]$$
$$Place2D \triangleq [PlaceID \to [PlaceID \to Nat]]$$

$$Place1DZeros \triangleq [i \in PlaceID \mapsto 0]$$
$$Place2DZeros \triangleq [i \in PlaceID \mapsto [j \in PlaceID \mapsto 0]]$$

$$Place1DTerminateTask(src, cnt) \triangleq [i \in PlaceID \mapsto \text{IF } i = src \text{ THEN } cnt \text{ ELSE } 0]$$

$$Place2DInitResilientFinish(home) \triangleq [i \in PlaceID \mapsto [j \in PlaceID \mapsto \text{IF } i = home \wedge j = home \text{ THEN } 1 \text{ ELSE}$$

$$
\begin{aligned}
Finish \triangleq\ & [id : FinishID \setminus \{ROOT\_FINISH\_ID\}, \\
& pred\_id : TaskID \cup \{NOT\_TASK\_ID\}, \quad \text{predecessor task} \\
& home : PlaceID, \\
& origin : PlaceID, \\
& parent\_finish\_id : FinishID, \\
& status : FINISH\_STATUS, \\
& lc : Nat]
\end{aligned}
$$

$$
\begin{aligned}
RootFinish \triangleq\ & [id \mapsto ROOT\_FINISH\_ID + 1,\\
& pred\_id \mapsto RootTask.id,\\
& home \mapsto FIRST\_PLACE\_ID,\\
& origin \mapsto FIRST\_PLACE\_ID,\\
& parent\_finish\_id \mapsto ROOT\_FINISH\_ID,\\
& status \mapsto \text{“active”},\\
& lc \mapsto 1]
\end{aligned}
$$

$$
\begin{aligned}
RootFinishTask \triangleq\ & [id \mapsto ROOT\_TASK\_ID + 1,\\
& pred\_id \mapsto ROOT\_TASK\_ID,\\
& dst \mapsto FIRST\_PLACE\_ID,\\
& src \mapsto FIRST\_PLACE\_ID,\\
& finish\_id \mapsto RootFinish.id,\\
& status \mapsto \text{“running”},\\
& level \mapsto 1,\\
& last\_branch \mapsto 0,\\
& type \mapsto \text{“finishMainTask”},\\
& finish\_type \mapsto \text{“global”}]
\end{aligned}
$$

$$
\begin{aligned}
NOT\_FINISH \triangleq\ & [id \mapsto NOT\_FINISH\_ID,\\
& home \mapsto NOT\_PLACE\_ID,\\
& origin \mapsto NOT\_PLACE\_ID,\\
& parent\_finish\_id \mapsto NOT\_FINISH\_ID,\\
& status \mapsto \text{“”},\\
& lc \mapsto 0]
\end{aligned}
$$

$$
\begin{aligned}
LFinish \triangleq\ & [id : FinishID \setminus \{ROOT\_FINISH\_ID\},\\
& home : PlaceID,\\
& lc : Nat,\\
& reported : Place1D,\\
& received : Place1D,\\
& deny : Place1D]
\end{aligned}
$$

$$
\begin{aligned}
RFinish \triangleq\ & [id : FinishID \setminus \{ROOT\_FINISH\_ID\},\\
& home : PlaceID,\\
& origin : PlaceID,\\
& parent\_finish\_id : FinishID,\\
& transOrLive : Place2D,\\
& sent : Place2D,\\
& gc : Nat,\\
& ghost\_children : \text{SUBSET } FinishID,\\
& isAdopted : \text{BOOLEAN }]
\end{aligned}
$$

$$
\begin{aligned}
Message \triangleq\ & [from : \{\text{“f”}, \text{“rf”}, \text{“src”}, \text{“dst”}, \text{“lf”}\},\\
& to : \{\text{“f”}, \text{“rf”}, \text{“src”}, \text{“dst”}, \text{“lf”}\},\\
& tag : \{\text{“transit”}, \text{“transitDone”}, \text{“transitNotDone”},
\end{aligned}
$$

```
                "terminateTask", "terminateGhost",
                "task",
                "publish", "publishDone",
                "release",
                "countDropped", "countDroppedDone"},
            src : PlaceID,
            dst : PlaceID,
            finish_id : FinishID,
            ghost_finish_id : FinishID,
            task_id : TaskID,
            term_tasks_by_src : Place1D,   termination only
            term_tasks_dst : PlaceID,     termination only
            num_sent : Nat,
            num_dropped : Nat
           ]

NOT_MESSAGE ≜ [from ↦ "N/A", to ↦ "N/A", tag ↦ "N/A",
               src ↦ NOT_PLACE_ID, dst ↦ NOT_PLACE_ID,
               finish_id ↦ NOT_FINISH_ID,
               task_id ↦ NOT_TASK_ID,
               ghost_finish_id ↦ NOT_FINISH_ID,
               term_tasks_by_src ↦ Place1DZeros,
               term_tasks_dst ↦ NOT_PLACE_ID]

FindRunningTask(maxLevel) ≜
  LET tset ≜ {task ∈ tasks : ∧ task.status = "running"
                             ∧ task.last_branch < WIDTH
                             ∧ task.level ≤ maxLevel}
  IN  IF tset = {} THEN NOT_TASK
      ELSE CHOOSE t ∈ tset : TRUE

FindRunningTaskWithFinishType(maxLevel, fin_type) ≜
  LET tset ≜ {task ∈ tasks : ∧ task.status = "running"
                             ∧ task.last_branch < WIDTH
                             ∧ task.level ≤ maxLevel
                             ∧ task.finish_type = fin_type}
  IN  IF tset = {} THEN NOT_TASK
      ELSE CHOOSE t ∈ tset : TRUE

FindFinishById(id) ≜
  CHOOSE f ∈ f_set : f.id = id

FindResilientFinishById(id) ≜
  CHOOSE f ∈ rf_set : f.id = id

FindTaskById(id) ≜
  CHOOSE t ∈ tasks : t.id = id
```

$ActiveFinishSet \triangleq \{\text{“active”}, \text{“waitingForRelease”}\}$

$FindActiveFinish(id, home) \triangleq$
LET $fset \triangleq \{finish \in f\_set : \land finish.status \in ActiveFinishSet$
$\land finish.id = id$
$\land \lor \land home \neq NOT\_PLACE\_ID$
$\land finish.home = home$
$\lor \land home = NOT\_PLACE\_ID\}$
IN IF $fset = \{\}$ THEN NOT_FINISH
ELSE CHOOSE $f \in fset$: TRUE

$FindPendingRemoteTask(finish\_id, status) \triangleq$
LET $tset \triangleq \{task \in tasks : \land task.status = status$
$\land task.src \neq task.dst$
$\land task.finish\_type = \text{“local”}$
$\land task.finish\_id = finish\_id$
$\}$
IN IF $tset = \{\}$ THEN NOT_TASK
ELSE CHOOSE $t \in tset$: TRUE

$IsPublished(finish\_id) \triangleq$
$\exists rf \in rf\_set : \land rf.id = finish\_id$
$\land \exists f \in f\_set : \land f.id = finish\_id$
$\land f.status \in ActiveFinishSet$
$LocalFinishExists(place, finish\_id) \triangleq$
$\exists lf \in lf\_set : \land lf.id = finish\_id$
$\land lf.home = place$

$ResilientFinishExists(finish\_id) \triangleq$
$\exists rf \in rf\_set : rf.id = finish\_id$

$FindLocalFinish(place, finish\_id) \triangleq$
CHOOSE $f \in lf\_set : f.home = place \land f.id = finish\_id$

$FindFinishToRelease(finish\_id) \triangleq$
CHOOSE $f \in f\_set : f.id = finish\_id \land f.status = \text{“waitingForRelease”} \land f.lc = 0$

a task can terminate if cannot branch further - at last level or at last branch number
$FindTaskToTerminate(fin\_type) \triangleq$
LET $tset \triangleq \{task \in tasks : \land task.status = \text{“running”}$
$\land task.finish\_type = fin\_type$
$\land \lor task.level = LEVEL$
$\lor task.last\_branch = WIDTH$
$\land$ IF $fin\_type = \text{“global”}$
THEN $FindActiveFinish(task.finish\_id, task.src) \neq NOT\_FINISH$
ELSE TRUE
$\}$

```
  IN   IF tset = {} THEN NOT_TASK
       ELSE CHOOSE t ∈ tset : TRUE

FindBlockedTask(task_id) ≜
  LET tset ≜ {task ∈ tasks : ∧ task.status = "blocked"
                             ∧ task.id = task_id
             }
  IN   IF tset = {} THEN NOT_TASK
       ELSE CHOOSE t ∈ tset : TRUE

SendMsg(m) ≜
  Add message to the msgs set
  msgs' = msgs ∪ {m}

RecvMsg(m) ≜
  Delete message from the msgs set
  msgs' = msgs \ {m}

ReplaceMsg(toRemove, toAdd) ≜
  Remove an existing message and add another one
  msgs' = (msgs \ {toRemove}) ∪ {toAdd}

FindMessageToActivePlaceWithTag(to, tag) ≜
  Return a message matching the given criteria, or NOT_MESSAGE otherwise
  LET mset ≜ {m ∈ msgs : ∧ m.to = to
                         ∧ m.tag = tag
                         ∧ IF m.to = "rf"
                           THEN TRUE
                           ELSE m.dst ∉ killed}
  IN   IF mset = {} THEN NOT_MESSAGE
       ELSE (CHOOSE x ∈ mset : TRUE)

Sum(place1D) ≜
    IF NUM_PLACES = 0 THEN 0
     ELSE IF NUM_PLACES = 1 THEN place1D[0]
     ELSE IF NUM_PLACES = 2 THEN place1D[0] + place1D[1]
     ELSE IF NUM_PLACES = 3 THEN place1D[0] + place1D[1] + place1D[2]
     ELSE IF NUM_PLACES = 4 THEN place1D[0] + place1D[1] + place1D[2] + place1D[3]
     ELSE IF NUM_PLACES = 5 THEN place1D[0] + place1D[1] + place1D[2] + place1D[3] + place1D[4]
     ELSE IF NUM_PLACES = 6 THEN place1D[0] + place1D[1] + place1D[2] + place1D[3] + place1D[4] +
     ELSE -1

NextRemotePlace(here) ≜
  IF nxt_remote_place[here] = here
   THEN (nxt_remote_place[here] + 1)%NUM_PLACES
   ELSE nxt_remote_place[here]
```

$ShiftNextRemotePlace(here) \triangleq$
IF $nxt\_remote\_place[here] = here$
THEN $nxt\_remote\_place' = [nxt\_remote\_place \text{ EXCEPT } ![here] = (nxt\_remote\_place[here] + 2)\%NUM\_PLA$
ELSE $nxt\_remote\_place' = [nxt\_remote\_place \text{ EXCEPT } ![here] = (nxt\_remote\_place[here] + 1)\%NUM\_PLA$

$FindLostTasks(dead) \triangleq$
{
$task \in tasks : \lor \land task.status \in \{\text{“waitingForPublish”}, \text{“waitingForTransit”}\}$
$\land task.src = dead$
$\lor \land task.status \in \{\text{“running”}, \text{“blocked”}\}$
$\land task.dst = dead$
}

$FindLostFinishes(dead) \triangleq$
{
$finish \in f\_set : \land finish.status \neq \text{“released”}$
$\land finish.home = dead$
}

$FindLostLocalFinishes(dead) \triangleq$
{
$local\_fin \in lf\_set : \land local\_fin.home = dead$
}

$FindImpactedResilientFinish(victim) \triangleq$
{
$id \in FinishID : \exists\, rf \in rf\_set : \land rf.id = id$
$\land \lor \exists\, i \in PlaceID : rf.transOrLive[i][victim] > 0$
$\lor \exists\, j \in PlaceID : rf.transOrLive[victim][j] > 0$
}

$FindImpactedResilientFinishToDead(victim) \triangleq$
{
$id \in FinishID : \exists\, rf \in rf\_set : \land rf.id = id$
$\land \exists\, i \in PlaceID : rf.transOrLive[i][victim] > 0$
}

$FindImpactedResilientFinishFromDead(victim) \triangleq$
{
$id \in FinishID : \exists\, rf \in rf\_set : \land rf.id = id$
$\land \exists\, j \in PlaceID : rf.transOrLive[victim][j] > 0$
}

$GetSrc(rf) \triangleq$
CHOOSE $i \in PlaceID$: CHOOSE $j \in killed : rf.transOrLive[i][j] > 0$

$GetDst(rf,\, src) \triangleq$

$\text{CHOOSE } j \in killed : rf.transOrLive[src][j] > 0$

$$GetAdoptedGhostChildren(fin\_id) \triangleq$$
$$\{$$
$$id \in FinishID : \exists\, rf \in rf\_set : \begin{array}{l} \wedge\, rf.id = id \\ \wedge\, rf.home \in killed \\ \wedge\, rf.parent\_finish\_id = fin\_id \\ \wedge\, rf.isAdopted = \text{TRUE} \end{array}$$
$$\}$$

$$GetNonAdoptedGhostChildren(fin\_id) \triangleq$$
$$\{$$
$$id \in FinishID : \exists\, rf \in rf\_set : \begin{array}{l} \wedge\, rf.id = id \\ \wedge\, rf.home \in killed \\ \wedge\, rf.parent\_finish\_id = fin\_id \\ \wedge\, rf.isAdopted = \text{FALSE} \end{array}$$
$$\}$$

$$IsRecoveringTasksToDeadPlaces(fin\_id) \triangleq$$
$$\begin{array}{ll} \vee \exists\, task \in rec\_child & : task.finish\_id = fin\_id \\ \vee \exists\, task \in rec\_to & : task.finish\_id = fin\_id \end{array}$$

$$ConvTask \triangleq [finish\_id : FinishID,\ from : PlaceID,\ to : PlaceID]$$

$$GetChildrenTask \triangleq [finish\_id : FinishID,\ victim : PlaceID,\ markingDone : \text{BOOLEAN}\ ]$$

$$NOT\_REQUEST \triangleq [finish\_id \mapsto NOT\_FINISH\_ID]$$

$$ChildRequestExists(fin\_id) \triangleq$$
$$\exists\, creq \in rec\_child : fin\_id = creq.finish\_id$$

$$ToRequestExists(fin\_id) \triangleq$$
$$\exists\, treq \in rec\_to : fin\_id = treq.finish\_id$$

$$FindMarkGhostChildrenRequest \triangleq$$
$$\begin{array}{l} \text{LET } rset \triangleq \{r \in rec\_child : r.markingDone = \text{FALSE}\} \\ \text{IN} \quad \text{IF } rset = \{\} \text{ THEN } NOT\_REQUEST \\ \qquad \text{ELSE } (\text{CHOOSE } x \in rset : \text{TRUE}) \end{array}$$

$$FindAddGhostChildrenRequest \triangleq$$
$$\begin{array}{l} \text{LET } rset \triangleq \{r \in rec\_child : r.markingDone = \text{TRUE}\} \\ \text{IN} \quad \text{IF } rset = \{\} \text{ THEN } NOT\_REQUEST \\ \qquad \text{ELSE } (\text{CHOOSE } x \in rset : \text{TRUE}) \end{array}$$

$$ChooseGhost(ghosts) \triangleq$$
$$\text{IF } ghosts = \{\} \text{ THEN } NOT\_FINISH \text{ ELSE } \text{CHOOSE } x \in rf\_set : x.id \in ghosts$$

$$FindToDeadRequest \triangleq$$
$$\text{IF } rec\_to = \{\} \text{ THEN } NOT\_REQUEST$$

$$
\begin{array}{l}
\quad \text{ELSE IF } \exists\, a \in rec\_to : \neg ChildRequestExists(a.finish\_id) \\
\qquad \text{THEN } (\text{CHOOSE } b \in rec\_to : \neg ChildRequestExists(b.finish\_id)) \\
\qquad \text{ELSE } NOT\_REQUEST
\end{array}
$$

$$
\begin{array}{l}
FindFromDeadRequest \triangleq \\
\quad \text{IF } rec\_from = \{\} \text{ THEN } NOT\_REQUEST \\
\quad \text{ELSE IF } \exists\, a \in rec\_from : \land \neg ChildRequestExists(a.finish\_id) \\
\qquad\qquad\qquad\qquad\qquad\quad \land \neg ToRequestExists(a.finish\_id) \\
\qquad \text{THEN } (\text{CHOOSE } b \in rec\_from : \land \neg ChildRequestExists(b.finish\_id) \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \land \neg ToRequestExists(b.finish\_id)) \\
\qquad \text{ELSE } NOT\_REQUEST
\end{array}
$$

$$
\begin{array}{l}
FindFromDeadWaitingRequest(fin\_id,\, from,\, to) \triangleq \\
\quad \text{CHOOSE } x \in rec\_from\_waiting : \land x.finish\_id = fin\_id \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \land x.from = from \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \land x.to = to
\end{array}
$$

$$
\begin{array}{l}
ApplyTerminateSignal(rf,\, rf\_updated,\, msg) \triangleq \\
\quad \text{IF } rf\_updated.gc = 0 \land rf\_updated.ghost\_children = \{\} \\
\qquad \text{THEN IF } rf.isAdopted \\
\qquad\qquad \text{THEN } \land ReplaceMsg(msg,\, [from \mapsto \text{“rf”},\, to \mapsto \text{“rf”},\, tag \mapsto \text{“terminateGhost”}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad finish\_id \mapsto rf.parent\_finish\_id, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad ghost\_finish\_id \mapsto rf.id, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad dst \mapsto NOT\_PLACE\_ID]) \;\; \text{rf.id is enough} \\
\qquad\qquad\qquad \land rf\_set' = rf\_set \setminus \{rf\} \\
\qquad\qquad \text{ELSE } \land ReplaceMsg(msg,\, [from \mapsto \text{“rf”},\, to \mapsto \text{“f”},\, tag \mapsto \text{“release”}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad finish\_id \mapsto rf.id, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad dst \mapsto rf.home]) \\
\qquad\qquad\qquad \land rf\_set' = rf\_set \setminus \{rf\} \\
\qquad \text{ELSE } \land RecvMsg(msg) \\
\qquad\qquad \land rf\_set' = (rf\_set \setminus \{rf\}) \cup \{rf\_updated\}
\end{array}
$$

$$
\begin{array}{l}
ApplyTerminateSignal2(rf,\, rf\_updated) \triangleq \\
\quad \text{IF } rf\_updated.gc = 0 \land rf\_updated.ghost\_children = \{\} \\
\qquad \text{THEN IF } rf.isAdopted \\
\qquad\qquad \text{THEN } \land SendMsg([from \mapsto \text{“rf”},\, to \mapsto \text{“rf”},\, tag \mapsto \text{“terminateGhost”}, \\
\qquad\qquad\qquad\qquad\qquad\quad finish\_id \mapsto rf.parent\_finish\_id, \\
\qquad\qquad\qquad\qquad\qquad\quad ghost\_finish\_id \mapsto rf.id, \\
\qquad\qquad\qquad\qquad\qquad\quad dst \mapsto NOT\_PLACE\_ID]) \;\; \text{rf.id is enough} \\
\qquad\qquad\qquad \land rf\_set' = rf\_set \setminus \{rf\} \\
\qquad\qquad \text{ELSE } \land SendMsg([from \mapsto \text{“rf”},\, to \mapsto \text{“f”},\, tag \mapsto \text{“release”}, \\
\qquad\qquad\qquad\qquad\qquad\quad finish\_id \mapsto rf.id, \\
\qquad\qquad\qquad\qquad\qquad\quad dst \mapsto rf.home]) \\
\qquad\qquad\qquad \land rf\_set' = rf\_set \setminus \{rf\} \\
\qquad \text{ELSE } \land msgs' = msgs \\
\qquad\qquad \land rf\_set' = (rf\_set \setminus \{rf\}) \cup \{rf\_updated\}
\end{array}
$$

$$RecvTerminateSignal(msg) \triangleq$$
$$\quad \land RecvMsg(msg)$$
$$\quad \land rf\_set' = rf\_set$$

$$RecvCountDroppedResponse(msg) \triangleq$$
$$\quad \land RecvMsg(msg)$$
$$\quad \land rf\_set' = rf\_set$$