MODULE *Heat*

EXTENDS *Naturals*, *TLC*, *Sequences*, *FiniteSets*

CONSTANT N, *Stack*, *deps*

VARIABLES *status*, *locks*, Q

$vars \triangleq \langle status, locks, Q \rangle$

ASSUME $IsFiniteSet(Stack)$

deps is a function of $r \mapsto \{resources\}$

ASSUME $IsFiniteSet(deps)$

$Status \triangleq \{\text{“READY”}, \text{“IN\_PROGRESS”}, \text{“COMPLETE”}\}$

$Locks \triangleq [status : \text{“FREE”}] \cup [status : \text{“BUSY”}, traversal : Nat]$

$Init \triangleq$

Here we define the functions that will let our logic run

status is a function that, given a resource, will tell us its state

$\land\ status = [x \in Stack \mapsto \text{“READY”}]$

locks are the underlying place we store sync points free/busy state

$\land\ locks = [x \in Stack \mapsto \text{“FREE”}]$

Q is just a work queue of resource *IDs*, and stands in for the time between a lock being acquired and the work being completed.

$\land\ Q = \langle\rangle$

$Complete(parent) \triangleq$

For a given resource, it is complete if it and all children are done

$\land \forall\, r \in deps[parent] : status[r] = \text{“COMPLETE”}$

$\land\ status[parent] = \text{“COMPLETE”}$

$ResReady[r \in Stack] \triangleq$

Readiness is defined as all dependencies being ready

$\land\ status[r] = \text{“READY”}$

$\land \forall\, d \in deps[r] : status[d] = \text{“COMPLETE”}$

$BeginAct(r) \triangleq$

Perform an action on a resource, in this (simple) case the only action is to satisfy the resource after ensuring the syncpoint is not in use by another traversal

The action itself is not executed here, but is enqueued to be done by the heat engine

$\land\ locks[r] = \text{“FREE”}$

$\land\ ResReady[r]$

Now we acquire the lock for our traversal

$\land\ locks' = [locks \text{ EXCEPT } ![r] = \text{“BUSY”}]$

And enqueue the resource to be worked on

$\land\ Q' = Append(Q, r)$

$\land$ UNCHANGED $status$

$Act \triangleq$
LET
$r \triangleq Head(Q)$
IN
$\land\ status' = [status \text{ EXCEPT } ![r] = \text{“COMPLETE”}]$
Take off the head item, since we just completed it
$\land\ Q' = Tail(Q)$
Release the syncpoint now that the work is complete
$\land\ locks' = [locks \text{ EXCEPT } ![r] = \text{“FREE”}]$

$TypeOK \triangleq$
$\land \forall\, d \in deps : d \in \text{SUBSET } Stack$
$\land\ Q \in Seq(Stack)$
Ensure a resource can never be *COMPLETE* until all its *deps* are
$\land \neg\exists\, r \in Stack : status[r] = \text{“COMPLETE”} \land \neg(\forall\, d \in deps[r] : ResReady[d])$

$Termination \triangleq$
$\land \forall\, r \in Stack : status[r] = \text{“COMPLETE”}$
$\land$ UNCHANGED $vars$

$Next \triangleq$
The next step is either to ready a resource, or to enqueue an action.
There isn't a qualification on which resources can be acted on
because the syncpoint being free and *deps* being ready are both
preconditions.
$\lor\ (Q \neq \langle\rangle \land Act)$
$\lor\ (\exists\, r \in Stack : BeginAct(r))$
$\lor\ Termination$

Theorems
$NonTriviality \triangleq$
$\land\ Stack \neq \{\}$

$Completeness \triangleq$
$\forall\, r \in Stack : status[r] = \text{“COMPLETE”}$

$Spec \triangleq Init \land \Box[Next]_{vars}$

THEOREM $Spec \Rightarrow TypeOK \land NonTriviality$

* Modification History
* Last modified *Tue Feb* 10 13:54:13 *EST* 2015 by *ryansb*
* Created *Fri Jan* 23 16:29:46 *EST* 2015 by *ryansb*