Starvation-Free Monitors

Jafar Hamin[0000−0002−5701−9111]

imec-DistriNet, Department of Computer Science, KU Leuven, Belgium
jafar.hamin@cs.kuleuven.be

A Monitor Implementing a CLH Lock

This document illustrates how starvation-freedom of a monitor implementing a *CLH lock*, shown in Figure 1, can be verified, as shown in Figure 3, using a publishable ghost list, shown in Figure 2.

```
routine new_tvm()
{
 $l$ := new_lock();
 $v$ := new_cvar();
 $node$ :=
   node($value$:=new_int(0));
 $q$ := queue($first$:=$node$,
   $last$:=$node$);
 tvm($l$:=$l$, $v$:=$v$, $q$:=$q$)
}
```

```
routine enter(tvm $t$){
 $new\_node$ :=
   node($value$:=new_int(1));
 acquire($t.l$);
 $mine$ := $t.q.last$;
 $t.q.last$ := $new\_node$;
 while($mine.value \neq 0$)
   wait($t.v, t.l$);
 $t.q.first$ = $new\_node$;
 release($t.l$)}
```

```
routine leave(tvm $t$)
{
 acquire($t.l$);
 $t.q.first.value$ := 0;
 notifyAll($t.v$);
 release($t.l$)
}
```

Fig. 1. A CLH-based monitor

NewGList
$\{\mathsf{true}\}\ \mathsf{g\_new\_list}\ \{\lambda g.\ \mathsf{glist}(g, [])\}$

PutToList
$\{\mathsf{glist}(g, l)\}\ \mathsf{g\_put}(g, i)\ \{\lambda\_.\ \mathsf{glist}(g, [i]\cdot l)\}$

TakeFromList
$\{\mathsf{glist}(g, l\cdot[i]) * i \mapsto v\}\ \mathsf{g\_take}(g)\ \{\lambda\_.\ \mathsf{glist}(g, l) * i \mapsto v * \mathsf{ptic}(g, 1, 0)\}$

Fig. 2. Ghost lists, where $[i]$ indicate a list with an element i, and $l_1{\cdot}l_2$ appends two lists l_1 and l_2.

```
$\mathsf{tvm}(\mathsf{tvm}\ t, \mathsf{gctr}\ gt, \mathsf{cond}\ v) ::= \mathsf{lock}(t.l) * \exists nodes.\ \mathsf{cond}(v, \mathsf{ptic}(t.gp), \{v\}) \wedge$
 $\mathsf{R}(t.l) < \mathsf{R}(v) \wedge \mathsf{L}(v){=}t.l \wedge \mathsf{I}(t.l) = \mathsf{linv}(t) \wedge gt = t.gt \wedge v = t.v$

$\mathsf{linv}'(\mathsf{tvm}\ t, \mathsf{list}{<}\mathsf{node}{>}\ nodes) ::= \lambda Wt.\ \lambda Ot.\ \lambda It.$
 $\exists q, first, last.\ t.q \mapsto q * q.first \mapsto first * m.last \mapsto last \wedge$
 $(last = 0\ ?\ first = last : last \in nodes) *$
 $\mathsf{nodes}([first]{\cdot}nodes) * \exists fvalue.\ first.value \xmapsto{.5} fvalue \wedge$
 $(ones = (fvalue = 1\ ?\ [first]{\cdot}nodes : nodes)) * \mathsf{g\_list}(t.gp, ones) \wedge$
 $(0 < |ones| \Rightarrow 0 < Ot(t.v)) \wedge It = \{(v, id) \mid id \in ones\} *$
 $\exists Tr.\ \mathsf{ctr}(t.gt, Tr) \wedge (1 < |ones| \Rightarrow ones - 1 \leqslant Tr \wedge Tr \leqslant |It|)$

$\mathsf{linv}(\mathsf{tvm}\ t) ::= \lambda Wt.\ \lambda Ot.\ \lambda It.\ \exists nodes.\ \mathsf{linv}'(t, nodes)(Wt, Ot, It)$

$\mathsf{nodes}(\mathsf{list}{<}\mathsf{node}{>}\ nodes) ::=$
 $\exists head, tail.\ nodes = [head]{\cdot}tail\ ?\ head.value \xmapsto{.5} 1 * \mathsf{nodes}(tail) : \mathsf{true}$

routine new_tvm(){
req : {true}
 $l := \mathsf{new\_lock}();$  $gt := \mathsf{g\_new\_ctr};$  $gp := \mathsf{g\_new\_list};$
 $v := \mathsf{new\_cvar}();$  $gt := \mathsf{g\_initc};$
 $node := \mathsf{node}(value{:=}\mathsf{new\_int}(0));$
 $q := \mathsf{queue}(first{:=}node, last{:=}node);$
 $t := \mathsf{tvm}(l{:=}l, v{:=}v, q{:=}q, gt{:=}gt, gp{:=}gp);$  $\mathsf{g\_initl}(l);\ t$
ens : $\lambda t.\ \mathsf{tvm}(t, gt, v) \wedge \mathsf{R}(v){=}r\}$

routine enter(tvm $t$){
req : $\{\mathsf{obs}(O) * \mathsf{tvm}(t, gt, v) \wedge v \prec O\}$
 $new\_node := \mathsf{node}(value{:=}\mathsf{new\_int}(1));$
 $\mathsf{acquire}(t.l);$
 $mine := t.q.last;$
 $t.q.last := new\_node;$  $\mathsf{g\_put}(t.gp, new\_node);$  $\mathsf{g\_ctr\_inc}(gt);$
 $\mathsf{if}(mine.value = 0)\ \mathsf{g\_charge}(v);$
 $\mathsf{while}(mine.value \neq 0)\{$
 inv :  $\mathsf{tvm}(t, gt, v) * \mathsf{tic}(gt) * \exists value.\ newnode.value \xmapsto{.5} value *$
   $\exists nodes, takens.\ \mathsf{g\_list}(t.gp, nodes) * \mathsf{ptic}(t.gp, 0, takens) *$
   $newnode \in nodes * \exists\, Wt, Ot, It.\ \mathsf{locked}(l,\ Wt, Ot, It) *$
   $mine.value = 0\ ?\ \mathsf{obs}(O{\uplus}\{l, v\}) * \mathsf{linv}'(t, nodes)(Wt, Ot, It) :$
   $\mathsf{obs}(O{\uplus}\{l\}) * \mathsf{linv}'(t, nodes)(Wt{\uplus}\{v\}, Ot, It{\cup}\{(v, mine)\})$
 var : $\mathsf{fronts}(nodes, newnode)$
   $\mathsf{wait}(t.v, t.l)\};$
 $t.q.first = new\_node;$
 $\mathsf{release}(t.l)$
ens : $\{\mathsf{obs}(O{\uplus}\{v\}) * \mathsf{tvm}(t, gt, v) * \mathsf{tic}(gt)\}\}$

routine leave(tvm $t$)
{
req : $\mathsf{obs}(O{\uplus}\{v\}) * \mathsf{tvm}(t, gt, v) * \mathsf{tic}(gt) \wedge v \prec O$
 $\mathsf{acquire}(t.l);$
 $t.q.first.value := 0;$  $\mathsf{g\_take}(t.gp);$  $\mathsf{if}(t.q.first = t.q.last)\ \mathsf{g\_discharge}(v);$
 $\mathsf{notifyAll}(t.v);$  $\mathsf{g\_ctr\_dec}(gt);$
 $\mathsf{release}(t.l)$
req : $\{\mathsf{obs}(O) * \mathsf{tvm}(t, gt, v)\}\}$
```

Fig. 3. Verification of termination of the monitor shown in Figure 1