Ask me anything

Mentimeter

4 questions
4 upvotes

How would you fix the TWOCOIN system to satisfy the REPICK safety property?

Mentimeter

```
TWOCOIN = (pick->COIN|pick->TRICK),
TRICK   = (toss->heads->TRICK),
COIN    = (toss->heads->COIN|toss->tails->COIN).

// How can you satisy this property?
property REPICK = (pick -> toss -> REPICK).

progress HEADS = {heads}
progress TAILS = {tails}
progress HEADSorTAILS = {heads,tails}
```

TRICK = (toss -> heads -> TWOCOIN),COIN = (toss -> heads -> TWOCOIN|toss->tails->TWOCOIN).

The trick coin should not be a terminal state

TWOCOIN = (pick->COIN | pick->TRICK | goback->TWOCOIN)

{ON,OFF}

{Runnable,NotRunnable,STOP}

{MAKER,USER}

6

What are the terminal sets of this process?

Mentimeter

{STOP}

STOP

{once}

```
ONESHOT = (once->STOP).
```

3

What are the terminal sets of this process?

Mentimeter

```
SWITCH = OFF,
OFF    = (on -> ON),
ON     = (off-> OFF).
```

{}	{OFF,ON}	ON
(ON, OFF)	{}	{ON, OFF}
{on, off}	there are none	{ON , OFF}

10

What are the terminal sets of this process?

Mentimeter

(OFF,ON)

```
SWITCH = OFF,
OFF    = (on -> ON),
ON     = (off-> OFF).
```

10

What are the terminal sets of this process?

Mentimeter

```
Thread =
    ( start -> Runnable ),
Runnable =
    ( yield -> Runnable
    | {sleep, wait, blockio} -> NotRunnable
    | stop -> STOP ),
NotRunnable =
    ( {awake, notify, unblockio} -> Runnable ).
```

{STOP}

{Runnable, NotRunnable,STOP}

{STOP}

{NotRunnable, Runnable}

{RUNNABLE, NotRunnable, STOP}

{Runnable, NotRunnable, STOP}

{ Stop}

STOP

STOP

11

What are the terminal sets of this process?

Mentimeter

{STOP}

{STOP}

```
Thread =
    ( start -> Runnable ),
Runnable =
    ( yield -> Runnable
    | {sleep, wait, blockio} -> NotRunnable
    | stop -> STOP ),
NotRunnable =
    ( {awake, notify, unblockio} -> Runnable ).
```

11

What are the terminal sets of this process?

Mentimeter

```
MAKER = (make->ready->MAKER).
USER  = (ready->use->USER).

||MAKER_USER = (MAKER || USER).
```

{MAKER, USER}

MAKER,USER

{MAKER, USER}

{MAKER, USER}

{USER} and {MAKER}

{MAKER, USER}

{}

{MAKER, MAKE, READY, USER}

What are the terminal sets of this process?

Mentimeter

{STOP}

{STOP}

{VAR[3]}

{ Stop}

{STOP}

```
const N = 3
range T = 0..N
set VarAlpha = {read[T],write[T]}

VAR      = VAR[0],
VAR[u:T] = ( read[u]   ->VAR[u]
           | write[v:T]->VAR[v]).

LOCK = (acquire->release->LOCK).

INC = (acquire -> read[v:0..N-1] -> write[v+1] -> release -> STOP)+VarAlpha.

||ParInc2 = ({a,b}::VAR || {a,b}::LOCK || a:INC || b:INC).
```

5

What would be a useful safety property for the dining philosophers?

Mentimeter

```
PHIL0 = (sitdown->left.get->right.get
         ->eat->left.put->right.put
         ->arise->PHIL0).

PHIL = (sitdown->right.get->left.get
         ->eat->left.put->right.put
         ->arise->PHIL).

FORK = (get -> put -> FORK).

||DINERS(N=5)=
  (PHIL0||
   forall [i:1..N-1]
   (phil[i]:PHIL
   ||{phil[i].left,phil[((i-1)+N)%N].right}::FORK)).
```

Progress

Guarded Method?

property SAFE=(eat->SAFE).

No two philosophers ever acquire the same fork

property Safe = (sitdown -> think -> Safe).

property SAFE_PHIL = (get->get->put->get->get->put)

left.put / right.put should always be possible

7

Are the Dining Philosopher solutions fair?

Mentimeter

Yes, No

The first solution is fair if we have a round table. The second solution very much depends on how we allow the philosophers into the queue

one philosopher might starve as he is too slow.

Break the cycle

> Number the forks. Philosophers grab the *lowest numbered fork* first.

> One philosopher grabs forks in the reverse order.

Philosophers queue to sit down

> allow *no more than four* at a time to sit down

3

How can you selectively wake up different threads or groups of waiting threads?

- notify();
- Using a Condition on a lock
- Conditions of locks or maybe you can use different objects to synchronize

Mentimeter

Last chance for questions