Lenshen He

43.5/50

```
-- Since(X,Y) is true precisely when X has been true at some point,
-- and Y has been continuously true afterwards
node HasHappened(X : bool) returns (Y : bool);
let
  Y = X -> (X or (pre Y));
tel

node Since( X, Y : bool ) returns ( Z : bool );
let
  Z =  X or (Y and (false -> pre Z));
tel
node UpCounter( Reset: bool ) returns ( C: int );

let
  C = 0 -> if Reset then 0 else (pre C) + 1 ;
tel

node UpCounter4( Reset: bool ) returns ( C: int );

let
  C = 0 -> if Reset and pre C > 3  then 0 else (pre C) + 1 ;
tel

node Counter( Input : bool ) returns ( C: int );

let
  C = 0 -> if pre C = 3 then 1 else if Input then pre C + 1 else pre C ;
tel



node Keypad( Digit: int; Press: bool ) returns ( Request: bool; Code: int );

var first, second, now, count : int ; P ,valid: bool ;

let
  valid = Digit > -1 and Digit < 10 ;
  count = Counter (  valid and Press ) ;
  now = 0-> if valid and Press then Digit else pre now;
    second = 0 -> if valid and Press then pre now else pre second ;
    first = 0 ->if valid and Press then pre second else pre first;
    Request = if ( count = 3 and Press) then true else false ;
    Code = 0  -> if Request then ( first * 100 + second *10 + now ) else pre Code;
    P = valid =>  if Digit > -1 and Digit < 10 and Code >-1 and Code <  1000 then true else false;

    --%PROPERTY P;
tel
```

when it's "always" between 0 and 9.

redundant

redundant

having to "issue a request"

-2

```
node ReqKeypad( Digit: int; Press: bool ) returns (R1 ,R2 : bool);
var Request : bool ; Code ,time : int;
let
  Request, Code = Keypad( Digit , Press );
    time =  UpCounter(false);
    -- R1 The component will not send an unlock requests before its third cycle.
    -- I assume cycle  means  clock cycle
    R1 = if time < 3 then not Request else true ;
    --R2 Any two distinct unlock requests are separated by at least 2 clock cycles.
  R2 = true -> Request => pre Request = false and pre ( pre Request) =false ;

    -- %MAIN;
-- %PROPERTY R1;
-- %PROPERTY R2;
tel
```

-1

when you have pre pre .. you should define firest 2 state, not only the first 1.

```
node Control( Request : bool; Code : int; MasterKey : bool ) returns ( CurrentCode : int; Granted : bool
```

(-1)

```
);
var isSet : bool ;
let
  -- initial CurrentCode as -1 for a resonable value
  CurrentCode = -1 -> if Request and MasterKey and pre MasterKey and pre pre MasterKey then Code else pre
CurrentCode ;
  Granted = if Request and ( Code = CurrentCode ) and isSet and not MasterKey then true else false ;
    -- in order to set the CurrentCode , Request and MasterKey must have been issued.
    isSet = HasHappened(Request and MasterKey ) ;
tel



node ReqControl(  Request : bool; Code : int; MasterKey : bool) returns (R1 ,R2 ,R3 , R4: bool);
var CurrentCode , time : int; Granted ,isSet , beingChanged: bool ;
let
  time =  UpCounter(false);
    CurrentCode , Granted = Control (Request , Code, MasterKey);
    beingChanged = MasterKey;
    isSet = HasHappened(Request and MasterKey ) ; -- in order to set code a MasterKey and Request must be
issued.
    --R1 Until the internal access code is first set, the door cannot be unlocked.
    R1 = not isSet => not Granted ;
   -- R2 Unless the internal access code is being changed, an unlock request is granted whenever the
provided code equals the internal access code
  R2 = ( not beingChanged => ( Code = CurrentCode and Request => Granted ) ) and (beingChanged => not
Granted ) ;
    -- R3 An unlock request is granted only if the provided code equals the internal access code.
    R3 = Granted => ( Code = CurrentCode ) ;
-- R4 Once it has been set, the internal access code changes only when the master key is inserted.
    R4 = isSet and  (CurrentCode <> pre CurrentCode ) => MasterKey;

    --assert Code < 1000 and Code > -1 ;

-- %MAIN
--%PROPERTY R1;
--%PROPERTY R2;
--%PROPERTY R3;
--%PROPERTY R4;
tel

--This node will hold the door unlocked for exactly 4 clock cycles once an unlock request is accepted.
node Lock ( Digit : int ; Press : bool ; MasterKey : bool)
returns ( Unlocking : bool) ;
var time , CurrentCode ,Code : int ; Granted ,Request : bool ;
let
  Request , Code =  Keypad ( Digit , Press);
  CurrentCode , Granted = Control ( Request , Code , MasterKey);
    time = UpCounter4( Granted and not pre Granted) ;
  Unlocking = false -> if time < 4 then pre Unlocking or Granted else false;

tel

node ReqLock(Digit : int ; Press : bool ; MasterKey : bool ) returns (R1, R2 :bool );
var Unlocking : bool ; time : int ;
let
  Unlocking = false -> Lock(Digit, Press, MasterKey);
    time =  0-> UpCounter(Unlocking and not pre Unlocking) ;
    R1 =  if time > 3 then not Unlocking else true ;
    R2 = Since ( false -> ( pre Unlocking and not Unlocking) , not Press) => not Unlocking;
   --%MAIN
--%PROPERTY R1;
--%PROPERTY R2;
```

OK

unless is not "iff"

-1

No idea why you need write like this. This is equivalent to Granted in this case.

-1

wrong based on your implementation of Uplcounter 4.

-0.5

```
tel


--R1 The door will never be kept unlocked for more than 4 clock cycles at a time.
--R2 Once a door locks (after being unlocked) it remains locked as long as no one uses the keypad.
/* TODO:

First is "changes the access code into the newly entered code—while keeping the door locked". This is a
hidden requirement not described in the Modeling. That means when change the access code door should be
locked and Key should insterted. And the formalized req:
CurrentCode = -1 -> if Request and MasterKey and pre MasterKey and pre pre MasterKey then Code else pre
CurrentCode ;
Granted = if Request and ( Code = CurrentCode ) and isSet and not MasterKey then true else false ;
Also there is no initial requirement for CurrentCode, so I choose the initial CurrentCode as -1. -1 is
not in the scope of the keypad input, it is safe for user to use and the isSet(currentCode has been set)
is used to ensure Until the internal access code is first set, the door cannot be unlocked.

Also I assume the Digit should be in the range [0..9], as I formalzie it in my code
valid = Digit > -1 and Digit < 10 ;
And I use this to ensure my Code in the Keypad won't have dangerous value.
Also I assume request will only stay for one clock cycle even if no key pressed.




*/
```