Modelação e Análise de um sistema de tempo-real em UPPAAL

Pedro Moura & Bruno Antunes

17 de Junho de 2020

Introdução

Problema

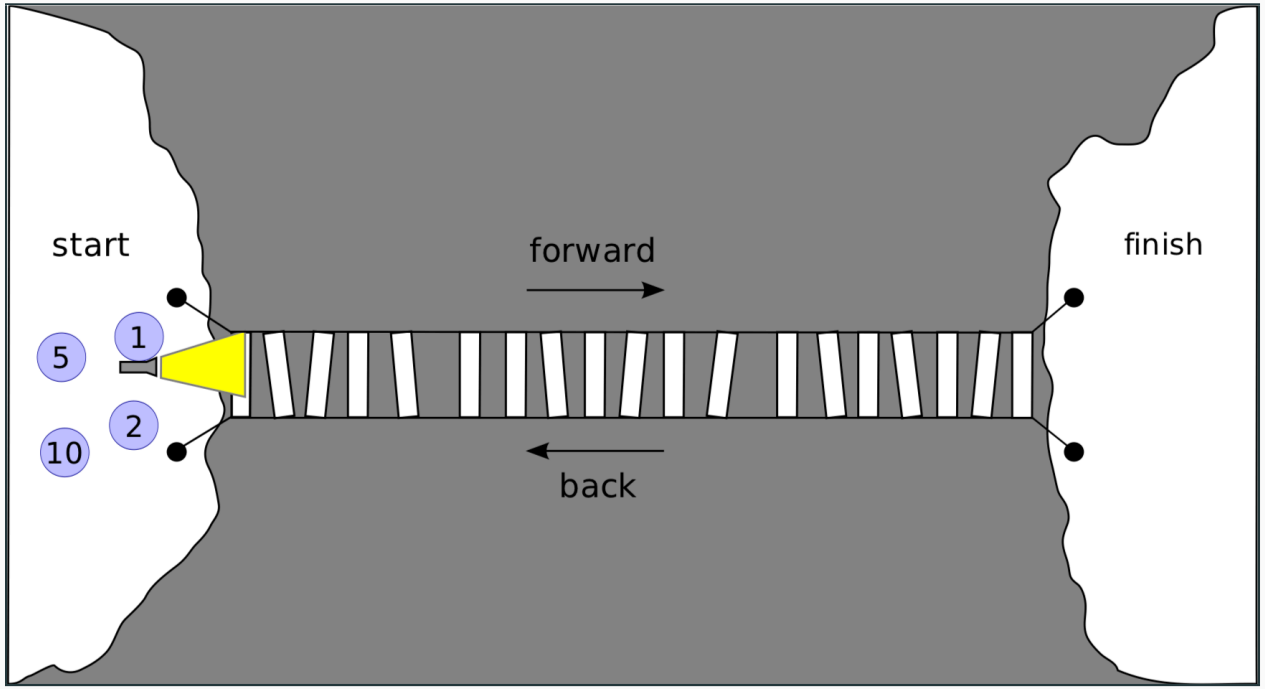

Figura 1: Problema da ponte.

Objetivo

Mostrar que:

- É possível que todos os aventureiros estejam do outro lado da ponte em 17 minutos.
- É impossível que todos os aventureiros estejam do outro lado da ponte em menos de 17 minutos.

Modelação

Lista de Durações

Set Constructor: $$T_X = X \mapsto (\mathbb{N} \times X)^*$$

Unit: $$\eta_X = singl \cdot \langle \underline{0}, id \rangle$$

Kleisli function: $$\frac{f : X \to (\mathbb{N} \times Y)}{f^* = concat \cdot map\ (\lambda(t, x) \to map\ wait_t\ (f\ x))}$$

Monads - Lista de Durações

```haskell
data ListDur a = LD [Duration a]
   deriving Show

instance Monad ListDur where
   return = LD . singl . return
   l >>= f = LD . concat . map (\(Duration (t,x))
                  -> map (wait t) (remLD (f x))) . remLD $ l
   --l >>= f = LD $ do
   --             Duration (d,x) <- remLD l
   --             map (wait d) (remLD (f x))
```

Figura 2: Lista de durações - código.

Monads - Lista de Durações com registo

Set Constructor: $(T_A)_X = X \mapsto (A^* \times (\mathbb{N} \times X))^*$

Unit: $\eta_X = singl \cdot \langle \underline{[]}, \langle \underline{0}, id \rangle \rangle$

Kleisli function:

$$\frac{f : X \to (A^* \times (\mathbb{N} \times Y))^*}{f^* = concat \cdot map\ (\lambda(l, (t, x)) \to map\ ((l +\!\!+) \times wait_t)\ (f\ x))}$$

Monads - Lista de Durações com registo

```haskell
data LogListDur t a = LLD [([t], Duration a)]
    deriving (Show, Eq)

instance Monad (LogListDur t) where
    return = LLD . singl . split nil return
    l >>= f = LLD . concatMap (\(l', Duration (t,x))
                  -> map ((l' ++) >< (wait t)) (remLLD (f x))) . remLLD $ l
    --l >>= f = LLD $ do
    --             (l1, Duration (t, x)) <- remLLD l
    --             let u = remLLD (f x) in
    --                 map (\(l2, d) -> (l1 ++ l2, wait t d)) u
```

Figura 3: Lista de durações com registo - código.

Resolução

```haskell
isCrossValid :: [Adventurer] → State → Bool
isCrossValid ps s = length ps ⩽ 2
                  && all (\p → s (Left p) == s (Right ())) ps


cross :: [Adventurer] → State → ListDur State
cross ps s = if isCrossValid ps s
             then LD [ Duration (maxT, mChangeState ((Right ()) : map Left ps) s) ]
             else LD []
        where maxT = maximum (map getTimeAdv ps)
```

Figura 4: Função *cross*.

Resolução

```haskell
oneCrossing :: State → ListDur State
oneCrossing s = manyChoice [ cross [p] s | p ← advList ]

twoCrossing :: State → ListDur State
-- twoCrossing s = manyChoice [ cross [p1,p2] s | p1 ← advList,  p2 ← advList, p1 /= p2 ]
twoCrossing s = manyChoice [ cross p s | p ← possiblePairs ] -- more efficient
        where possiblePairs = [[P1,P2], [P1,P5], [P1,P10], [P2,P5], [P2, P10], [P5,P10]]

{-- For a given state of the game, the function presents all the
possible moves that the adventurers can make.  --}
allValidPlays :: State → ListDur State
allValidPlays s = manyChoice [ oneCrossing s, twoCrossing s ]
```

Figura 5: Função *allValidPlays*.

Resolução

```haskell
{-- For a given number n and initial state, the function calculates
all possible n-sequences of moves that the adventures can make --}
exec :: Int → State → ListDur State
exec 0 s = return s
exec n s = allValidPlays s >>= exec (n-1)
```

Figura 6: Função exec.

Verificação

Propriedades

```haskell
{-- Is it possible for all adventurers to be on the other side
in ⩽17 min and not exceeding 5 moves ? --}
-- To implement
leq17 :: Bool
leq17 = any (\(Duration (d,s)) → d ⩽ 17 && s == const True) (remLD (exec 5 gInit))

{-- Is it possible for all adventurers to be on the other side
in < 17 min ? --}
-- To implement
l17 :: Bool
l17 = any (\(Duration (d,s)) → d < 17 && s == const True) (remLD (exec 5 gInit))

solution :: Duration State
solution = head $ filter (\(Duration (d,s)) → d ⩽ 17 && s == const True) (remLD (exec 5 gInit))
```

Figura 7: **Propriedades** - código.

Solução

```
[*Adventurers> solution
Duration (17,["True","True","True","True","True"])
```

```
[*AdventurersLog> solution                                                    ]
([False ---[P1,P2] t=2⟶ True,False ⟵[P1] t=1--- True,False ---[P5,P10] t=10⟶ True,False ⟵[
P2] t=2--- True,False ---[P1,P2] t=2⟶ True],Duration (17,["True","True","True","True","True"]))
```

UPPAAL vs HASKELL

UPPAAL vs HASKELL

- Criação do modelo
- Visualização
- Verificação

Modelação e Análise de um sistema de tempo-real em UPPAAL

Pedro Moura & Bruno Antunes

17 de Junho de 2020