

Exercises Week 3 — Solutions

Some preliminary code:

```
module LabExerc3 where

import List
import LAI9
import LAI10
import LAI11
```

1. Define functions

`myModel1 :: EpistM State` and `myModel2 :: EpistM State`

that generate the models from exercises 1 and 3 in the Pencil and Paper exercises of this week. Next compare `myModel1` with `bisim myModel1`, and `myModel2` with `bisim myModel2`, and check if the answers agree with your findings of the Pencil and Paper exercises.

Answer:

```
myModel1 :: EpistM State
myModel1 = Mo [0..6]
            [a,b]
            [(0, []), (1, []), (2, []), (3, [P 0]),
             (4, [P 0]), (5, []), (6, [P 0])]
            [(a,0,1), (a,1,1), (a,0,2), (a,2,5), (a,5,5),
             (b,1,3), (b,2,4), (b,5,6)]
            [0]

myModel2 :: EpistM State
myModel2 = Mo [0..7]
            [a,b]
            [(0, []), (1, []), (2, []), (3, []),
             (4, []), (5, []), (6, []), (7, [])]
            [(a,0,1), (a,1,1), (a,0,2), (a,2,5), (a,5,5),
             (b,1,3), (b,2,4), (b,5,6), (b,0,7)]
            [0]
```

We get:

```

LabExerc3> bisim myModel1
Mo [0,1,2] [a,b] [(0,[]),(1,[]),(2,[p])] [(a,0,1),(a,1,1),(b,1,2)] [0]
LabExerc3> bisim myModel2
Mo [0,1] [a,b] [(0,[]),(1,[p])] [(a,0,0),(b,0,1)] [0]

```

2. Use `myModel1` and `myModel2` from the previous exercise to check how the partition refinement algorithm handles these models. What are the initial partitions? How many refinement steps are there? What happens in these steps? Compare with your pencil and paper findings.

Answer:

For the first model we get:

```

LabExerc3> initPartition myModel1
[[0,1,2,5],[3,4,6]]
LabExerc3> refineStep myModel1 [[0,1,2,5],[3,4,6]]
[[0],[1,2,5],[3,4,6]]
LabExerc3> refineStep myModel1 [[0],[1,2,5],[3,4,6]]
[[0],[1,2,5],[3,4,6]]

```

In the second refinement step no splitting takes place, so we have reached the fixpoint.

For the second model we get:

```

LabExerc3> initPartition myModel2
[[0,1,2,5],[3,4,6,7]]
LabExerc3> refineStep myModel2 [[0,1,2,5],[3,4,6,7]]
[[0,1,2,5],[3,4,6,7]]

```

This shows that in this case the initial partition is already a fixpoint for the refinement operation.

3. Define a function

```
gsm :: Ord state => EpistM state -> EpistM state
```

that maps an epistemic model to its generated submodel (see exercise 5 in Pencil and Paper exercises of this week).

Solution:

```

reachable :: Ord state => EpistM state -> [state]
reachable (Mo states agents val rel actual) =
  (sort.nub)
    (concat [ commonAlts rel agents states w | w <- actual ])

gsm :: Ord state => EpistM state -> EpistM state
gsm m@(Mo states agents val rel actual) =
  Mo states' agents val' rel' actual
  where
    states' = reachable m
    val'    = [(x,y) | (x,y) <- val, elem x states' ]
    rel'    = [(a,x,y) | (a,x,y) <- rel,
                      elem x states', elem y states']

```

An example to test this out:

```

gsm_example =
  Mo [0..3]
    [a..c]
    [(0,[]), (1,[P 0]), (2,[Q 0]), (3,[P 0, Q 0])]
    [( (a,x,x) | x <- [0..3] ] ++
      [ (b,x,x) | x <- [0..3] ] ++
      [ (c,x,x) | x <- [0..3] ])
  [1]

```

This gives:

```

LabExerc3> gsm gsm_example
Mo [1] [a,b,c] [(1,[p])] [(a,1,1),(b,1,1),(c,1,1)] [1]

```

4. HOMEWORK Define a function

```
part2pairs :: [[a]] -> Rel a
```

that maps a list partition to the corresponding relation (list of pairs).

Answer:

```

part2pairs :: [[a]] -> [(a,a)]
part2pairs [] = []
part2pairs (b1:blocks) =
  [(x,y) | x <- b1, y <- b1] ++ part2pairs blocks

```

Another possibility (with thanks to Bas Steunebrink) that is even simpler but perhaps somewhat less intuitive, is:

```

part2pairs :: [[a]] -> [(a,a)]
part2pairs blocks = [(x,y) | b1 <- blocks, x <- b1, y <- b1 ]

```

5. HOMEWORK Use the function from the previous exercise to define a function

```
maxBisim :: Eq state => EpistM state -> Rel state
```

that computes the maximal bisimulation on an epistemic model.

Answer:

```

maxBisim :: Ord state => EpistM state -> Rel state
maxBisim m = part2pairs partition
  where (Mo partition _ _ _ _) = minimalModel m

```