Exercises 4

```haskell
module E4 where

import Data.List
```

Exercise 4.1: Model Checking PAL

Recall the following functions from the lecture today:

```haskell
(!) :: Eq a => [(a,b)] -> a -> b
(!) v x = let (Just y) = lookup x v in y

(?) :: Eq a => [[a]] -> a -> [a]
(?) lls x = head (filter (x `elem`) lls)

type Prop = Int
type Ag = String
data Form = P Prop | Neg Form | Con Form Form | K Ag Form  deriving (Eq,Ord,Show)

dis :: Form -> Form -> Form
dis f g = Neg (Con (Neg f) (Neg g))

type World = Int
type Relations = [(Ag, [[World]])]
type Valuation = [(World, [Prop])]
data Model = Mo { worlds :: [World]
                , rel :: Relations
                , val :: Valuation }
  deriving (Eq,Ord,Show)

isTrue :: (Model,World) -> Form -> Bool
isTrue (m,w) (P p)     = p `elem` (val m ! w)
isTrue (m,w) (Neg f)   = not (isTrue (m,w) f)
isTrue (m,w) (Con f g) = isTrue (m,w) f && isTrue (m,w) g
isTrue (m,w) (K i f)   = and [ isTrue (m,w') f | w' <- (rel m ! i) ? w ]
```

Implement the `announce` function to make public announcements:

```haskell
announce :: Model -> Form -> Model
announce oldModel@(Mo oldWorlds oldRel oldVal) f = Mo newWorlds newRel newVal where
  newWorlds = undefined
  newRel    = undefined
  newVal    = undefined
```

Congratulations, you now have a simple model checker for Public Announcement Logic (PAL), the simplest version of Dynamic Epistemic Logic (DEL).

Other features you can add:

- the announcement operator such as $[!\phi]\psi$ to the `Form` type.
- the knowing-whether operator
- operators for distributed knowledge and common knowledge of groups of agents (Note: for common knowledge you need a transitive closure. See https://staff.fnwi.uva.nl/d.j.n.vaneijck2/software/demo_s5/EREL.pdf for how to compute it when relations are partitions.)

For more inspiration you can have a look at DEMO, DEMO-S5 and SMCDEL (see course website).

Exercise 4.2: Muddy Children

Use your code from the previous exercise to solve this version of the muddy children puzzle:

> Three children play together. Some of the children get mud on their foreheads. Each can see the mud on others but not on their own forehead. The father says: "At least one of you has mud on your forehead". The father then asks: “Do you know whether you have mud on your own forehead? The please raise your hand.” Nobody reacts. The father repeats the question a second time and now some children raise their hand. How many children are muddy?

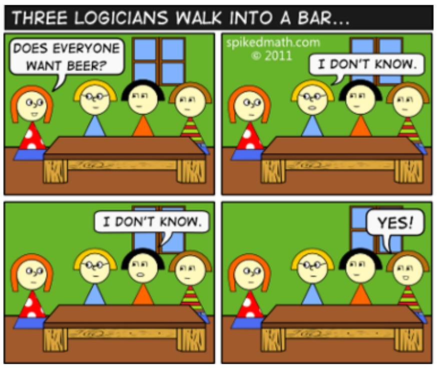
```haskell
start :: Model
start = Mo
  -- worlds:
  [ 0 -- nobody is muddy
  , 1 -- only child 1 is muddy
  , 2 -- only child 2 is muddy
  , 3 -- only child 3 is muddy
  , 4 -- 1 and 2 are muddy
  , 5 -- 2 and 3 are muddy
  , 6 -- 1 and 3 are muddy
  , 7 -- all three are muddy
  ]
  -- relations -- fill in the missing ones!
  [("1",[[0,1],[2,4],[3,6],[5,7]])
  ,("2",undefined)
  ,("3",undefined)]
  -- valuation:
  [(0,[   ]),(1,[1    ]),(2,[  2  ]),(3,[    3])
  ,(4,[1,2]),(5,[  2,3]),(6,[1,  3]),(7,[1,2,3])]

muddy2 :: Model
muddy2 = start `announce` undefined -- hint: use "dis" to say "at least one ..."
```

Evaluate `start` and `muddy2` in ghci to see which worlds are removed by the announcement. Then define a new `muddy3` by making the next announcement, and so on.

Exercise 4.3: Drinking Logicians

The following is dual to the Muddy Children example:



Write a model for the drinking logicians and check:

(Side note: to make your life easier, replace “is common knowledge” with “everyone knows” in the statements below.)

- After the first logician says “I don’t know.” it is common knowledge that she wants beer.
- The sequence of all four announcements is possible iff everyone wants beer.
- After the second logician says “I don’t know.” it is common knowledge that the third logician knows whether everyone wants beer.

Use your model to verify or falsify the following statements:

- After the second logician says "I don't know." it is common knowledge that the third logician knows *that* everyone wants beer.
- If the third logician would say *No!* it would be common knowledge that only the first two want beer and the third does not.

Exercise 4.4: Moore Sentences

(This is a longer exercises. Feel free to skip.)

Intuitively, after announcing something, all agents know it. So we might think that for any ϕ the PAL formula $[!\phi]K_i\phi$ is valid:

```haskell
conjecture :: (Model,Form) -> Bool
conjecture (m,f) = and [ isTrue (m,w) f | w <- worlds m ]
```

However, this conjecture is wrong. Let us use QuickCheck to find a counter example!

First you should write instances of `Arbitrary` for `Model` and `Form`. You might have code from Exercises 2 that can be reused or adapted for this. Another option is to steal from the SMCDEL code, click here and click here.

Then you can run `quickCheck conjecture`.

Can you think of other conjectures and quickcheck them?

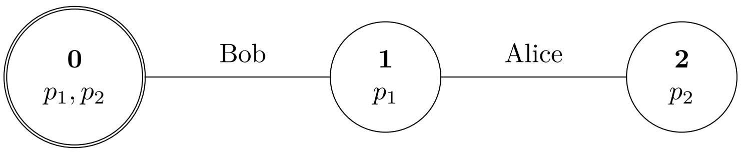
Side note: Why did you not have to write an `Arbitrary` instance for `(Model,Form)`?

Exercise 4.5: Kripke Models and Knowledge Structures

Part (i) Find a Kripke model which is equivalent to this knowledege structure:

$$\mathcal{F} = (V = \{p, q, r\}, \theta = (p \lor q) \to r, O_a = \{p\}, O_b = \{q\})$$

Part (ii) Find a knowledge structure which is equivalent to this Kripke model:



```
KrM [0,1,2]
    [ ("Alice",[[0],[1,2]])
    , ("Bob"  ,[[0,1],[2]]) ]
    [ (0,[(P 1,True ),(P 2,True )])
    , (1,[(P 1,True ),(P 2,False)])
    , (2,[(P 1,False),(P 2,True )]) ]
```

First solve this exercise manually with pen on paper. Then use SMCDEL to check your answers. Part (i) can be done in the web interface at https://w4eg.de/malvin/illc/smcdelweb. For part (ii) you need the whole smcdel Haskell library and find the appropriate function. See https://github.com/jrclogic/SMCDEL or use `stack unpack smcdel`.

Exercise 4.6: Binary Decision Diagrams

Draw binary decision diagrams for the following formulas and statements. First try to do it by hand, then check your results with the BDD package *HasCacBDD* or the SCMDEL web interface.

- $p \lor q$
- $(p \land q) \to (p \land r)$
- At least one of the propositions p, q and r is true.
- Exactly two of the propositions p, q and r are true.

Bonus Exercise 4.7: The Dining Cryptographers

Read about the Dining cryptographers problem on Wikipedia, then look at the SMCDEL formalization of it (available in the web interface). Extend it to the case with four agents.

Bonus Exercise 4.8: Cheryl's Birthday

The following is from the Singapore and Asian Schools Math Olympiad 2015. At some point it went "viral", as they say.

> Albert and Bernard just become friends with Cheryl, and they want to know when her birthday is. Cheryl gives them a list of 10 possible dates:
>
> May 15, May 16, May 19, June 17, June 18, July 14, July 16, August 14, August 15, August 17
>
> Cheryl then tells Albert and Bernard separately the month and the day of her birthday respectively. Then the following dialogue takes place.
>
> Albert: I don't know when Cheryl's birthday is, but I know that Bernard does not know too.
>
> Bernard: At first I don't know when Cheryl's birthday is, but I know now.
>
> Albert: Now I also know when Cheryl's birthday is.
>
> So when is Cheryl's birthday?

Can you solve this puzzle using your own model checking code?

You can find a solution using the explicit model checker DEMO-S5 at https://malv.in/posts/2015-04-20-finding-cheryls-birthday-with-DEMO.html and a solution using SMCDEL at https://malv.in/posts/2019-03-01-symbolically-finding-cheryls-birthday-with-SMCDEL.html.