Intro to Epistemic Model Checking

Jan van Eijck

jve@cwi.nl

August 7, 2008

Abstract

We will add a mechanism for factual change to the action model update definition. Next, we apply the resulting system to the analysis of epistemic puzzles and protocols.

Today

- Adding Factual Change
- Wise Men Puzzle
- Muddy Children Puzzle
- ...

Module Declaration

```
module IEMC where

import List
import HFKR
import RPAU
import RAMU
```

Change in the World

Following [1], we represent changes in the world as substitutions. A substitution maps proposition letters to formulas. Type of a substitition in Haskell:

```
type Subst = [(Prop,Form)]
```

Action+Change models

```
data ACM state = Acm
               [state]
               [Agent]
               [(state,(Form,Subst))]
               [(Agent,state,state)]
               [state]  deriving (Eq,Show)
```

Functions from Agents to A+C models

```
type FACM state = [Agent] -> ACM state
```

Getting the precondition and the substitution from an A+C model

```
prec :: ACM state -> [(state,Form)]
prec (Acm _ _ ps _ _) =
    map (\ (x,(y,_)) -> (x,y)) ps

subst :: ACM state -> [(state,Subst)]
subst (Acm _ _ ps _ _) =
    map (\ (x,(_,y)) -> (x,y)) ps
```

From Tables to Functions

```
t2f :: Eq a => [(a,b)] -> a -> b
t2f t = \ x -> maybe undefined id (lookup x t)
```

Substitutions as functions:

```
sub :: Eq a => ACM a -> a -> Prop -> Form
sub am s p = let
     sb = t2f (subst am) s in
  if elem p (map (\ (x,_) -> x) sb) then
     t2f sb p
   else (Prop p)
```

Changing the World

Valuation at a world in an epistemic model:

```
val :: Eq a => EpistM a -> a -> [Prop]
val m = t2f (valuation m)
```

New valuation after update with an action model

```
newVal :: (Eq a, Ord a) =>
  EpistM a -> ACM a -> (a,a) -> [Prop]
newVal m am (w,s) = [ p | p <- allprops, subfct p ]
   where
   allprops = (sort.nub)
    ((val  m w) ++
     (map (\ (x,_) -> x) (t2f (subst am) s)))
   subfct p = isTrueAt m w (sub am s p)
```

Updating with an A+C Model

```
upc :: (Eq state, Ord state) =>
       EpistM state -> FACM state
                    -> EpistM (state,state)
```

```
upc m@(Mo worlds ags val rel points) facm =
   Mo worlds' ags' val' rel' points'
   where
    acm@(Acm states ags' ps susp as) = facm ags
    worlds' = [ (w,s) | w <- worlds, s <- states,
                  isTrueAt m w (apply (prec acm) s)]
    val'    = [ ((w,s),newVal m acm (w,s)) |
                                  (w,s) <- worlds' ]
    rel'    = [ (ag1,(w1,s1),(w2,s2)) |
                  (ag1,w1,w2) <- rel,
                  (ag2,s1,s2) <- susp,
                   ag1 == ag2,
                   elem (w1,s1) worlds',
                   elem (w2,s2) worlds'   ]
    points' = [ (p,a) | p <- points, a <- as,
                        elem (p,a) worlds'    ]
```

Update and Simplify

```
upd ::  (Eq state, Ord state) =>
        EpistM state -> FACM state
                     -> EpistM State
upd m a = bisim (upc m a)
```

Public Announcement

See [3, 2].

Update model consists of a single action, with reflexive arrows for all agents.

Precondition is the formula that expresses the content of the announcement.

```
public :: Form -> FACM State
public form ags = Acm [0] ags [(0,(form,[]))]
                      [(a,0,0)| a <- ags ] [0]
```

Example

```
m0 = upc s5example (public p)
m1 = upd s5example (public p)
```

```
IEMC> displayS5 s5example
[0,1,2,3]
[(0,[]),(1,[p]),(2,[q]),(3,[p,q])]
(a,[[0],[1],[2],[3]])
(b,[[0],[1],[2],[3]])
(c,[[0,1,2,3]])
[1]
```

```
IEMC> displayS5 m0
[(1,0),(3,0)]
[((1,0),[p]),((3,0),[p,q])]
(a,[[(1,0)],[(3,0)]])
(b,[[(1,0)],[(3,0)]])
(c,[[(1,0),(3,0)]])
[(1,0)]


IEMC> displayS5 m1
[0,1]
[(0,[p]),(1,[p,q])]
(a,[[0],[1]])
(b,[[0],[1]])
(c,[[0,1]])
[0]
```

Public Change

```
pChange :: Subst -> FACM State
pChange subst ags = Acm [0] ags [(0,(Top,subst))]
                        [(a,0,0)| a <- ags ] [0]
```

Example

```
m2 = upc s5example (pChange [(P 0,q)])
m3 = upd s5example (pChange [(P 0,q)])
```

```
IEMC> displayS5 m2
[(0,0),(1,0),(2,0),(3,0)]
[((0,0),[]),((1,0),[]),((2,0),[p,q]),((3,0),[p,q])]
(a,[[(0,0)],[(1,0)],[(2,0)],[(3,0)]])
(b,[[(0,0)],[(1,0)],[(2,0)],[(3,0)]])
(c,[[(0,0),(1,0),(2,0),(3,0)]])
[(1,0)]


IEMC> displayS5 m3
[0,1]
[(0,[]),(1,[p,q])]
(a,[[0],[1]])
(b,[[0],[1]])
(c,[[0,1]])
[0]
```

Group Announcement

Computing the update for passing a group announcement to a list of agents: the other agents confuse the action with the action where nothing happens.

In the limit case where the message is passed to all agents, the message is a public announcement.

```
groupM :: [Agent] -> Form -> FACM State
groupM gr form agents =
  if sort gr == sort agents
    then public form agents
    else
      (Acm
       [0,1]
        agents
       [(0,(form,[])),(1,(Top,[]))]
        ([ (a,0,0) | a <- agents ]
         ++ [ (a,0,1) | a <- agents \\ gr ]
         ++ [ (a,1,0) | a <- agents \\ gr ]
         ++ [ (a,1,1) | a <- agents          ])
       [0])
```

Example

```
e0 = initM [a,b,c] [P 0,Q 0]
m4 = upc e0 (groupM [a,b] (Neg p))
m5 = upd e0 (groupM [a,b] (Neg p))
```

Example

```
e0 = initM [a,b,c] [P 0,Q 0]
m4 = upc e0 (groupM [a,b] (Neg p))
m5 = upd e0 (groupM [a,b] (Neg p))
```

```
IEMC> displayS5 e0
[0,1,2,3]
[(0,[]),(1,[p]),(2,[q]),(3,[p,q])]
(a,[[0,1,2,3]])
(b,[[0,1,2,3]])
(c,[[0,1,2,3]])
[0,1,2,3]
```

```
IEMC> displayS5 m4
[(0,0),(0,1),(1,1),(2,0),(2,1),(3,1)]
[((0,0),[]),((0,1),[]),((1,1),[p]),((2,0),[q]),((2,1),[q]),(
(a,[[(0,0),(2,0)],[(0,1),(1,1),(2,1),(3,1)]])
(b,[[(0,0),(2,0)],[(0,1),(1,1),(2,1),(3,1)]])
(c,[[(0,0),(0,1),(1,1),(2,0),(2,1),(3,1)]])
[(0,0),(2,0)]


IEMC> displayS5 m5
[0,1,2,3,4,5]
[(0,[]),(1,[]),(2,[p]),(3,[q]),(4,[q]),(5,[p,q])]
(a,[[0,3],[1,2,4,5]])
(b,[[0,3],[1,2,4,5]])
(c,[[0,1,2,3,4,5]])
[0,3]
```

Private Messages

Private messages are a special case of group messages:

```
message :: Agent -> Form -> FACM State
message agent = groupM [agent]
```

Tests

Tests are another special case of group messages:

```
test :: Form -> FACM State
test = groupM []
```

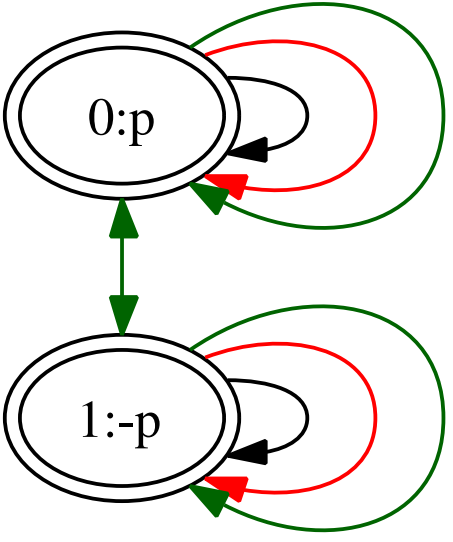
Communications Whether

- informing everone whether φ,
- informing a group whether φ,
- informing an individual whether φ.

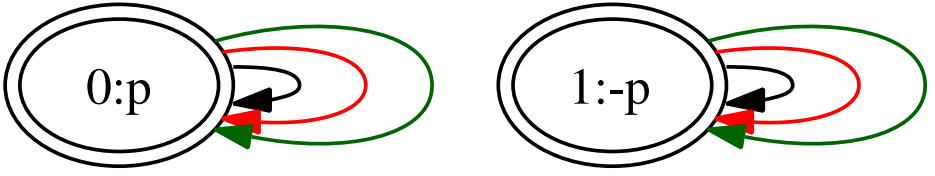
Telling someone whether it rains involves giving her the facts: if it rains you tell her "it rains", if it does not rain you tell her "it does not rain".

In the action model for this there are two actual actions. Which one will lead to a new actual world depends on the facts of the matter, and these are determined by the input model ...

General Form: Group Communication Whether



Informing Everyone Whether p



Implementation

First the negation of a formula:

```
negation :: Form -> Form
negation (Neg form) = form
negation form       = (Neg form)
```

Informing a Group Whether φ

```
info :: [Agent] -> Form -> FACM State
info group form agents =
   Acm
   [0,1]
   agents
   [(0,(form,[])),(1,(negation form,[]))]
   ([ (a,0,0) | a <- agents ]
     ++ [ (a,1,1) | a <- agents ]
     ++ [ (a,0,1) | a <- others ]
     ++ [ (a,1,0) | a <- others ])
   [0,1]
     where others = agents \\ group
```

Example

```
m6 = upc e0 (info [a,b] p)
m7 = upd e0 (info [a,b] p)
```

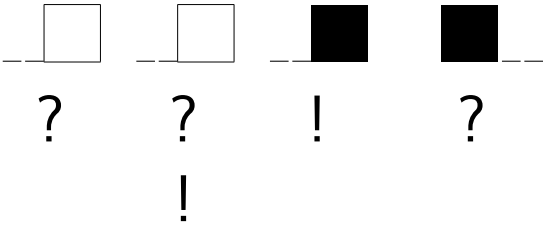
Example

```
m6 = upc e0 (info [a,b] p)
m7 = upd e0 (info [a,b] p)
```

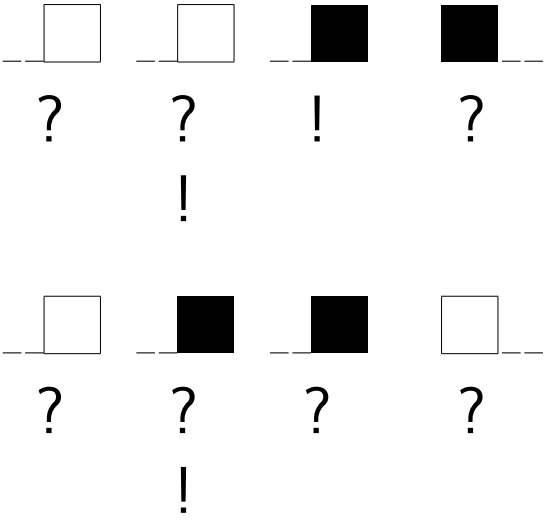
```
IEMC> displayS5 m6
[(0,1),(1,0),(2,1),(3,0)]
[((0,1),[]),((1,0),[p]),((2,1),[q]),((3,0),[p,q])]
(a,[[(0,1),(2,1)],[(1,0),(3,0)]])
(b,[[(0,1),(2,1)],[(1,0),(3,0)]])
(c,[[(0,1),(1,0),(2,1),(3,0)]])
[(0,1),(1,0),(2,1),(3,0)]
```

```
IEMC> displayS5 m7
[0,1,2,3]
[(0,[]),(1,[p]),(2,[q]),(3,[p,q])]
(a,[[0,2],[1,3]])
(b,[[0,2],[1,3]])
(c,[[0,1,2,3]])
[0,1,2,3]
```

Wise Men Puzzle; or: The Riddle of the Caps



Wise Men Puzzle; or: The Riddle of the Caps



Analysis with Model Checking

- Four agents: a, b, c, d, occupying positions $1, 2, 3, 4$.
- Four basic propositions p_1, p_2, p_3, p_4.
- p_i expresses that the guy at position i is wearing a white cap.

Initial model

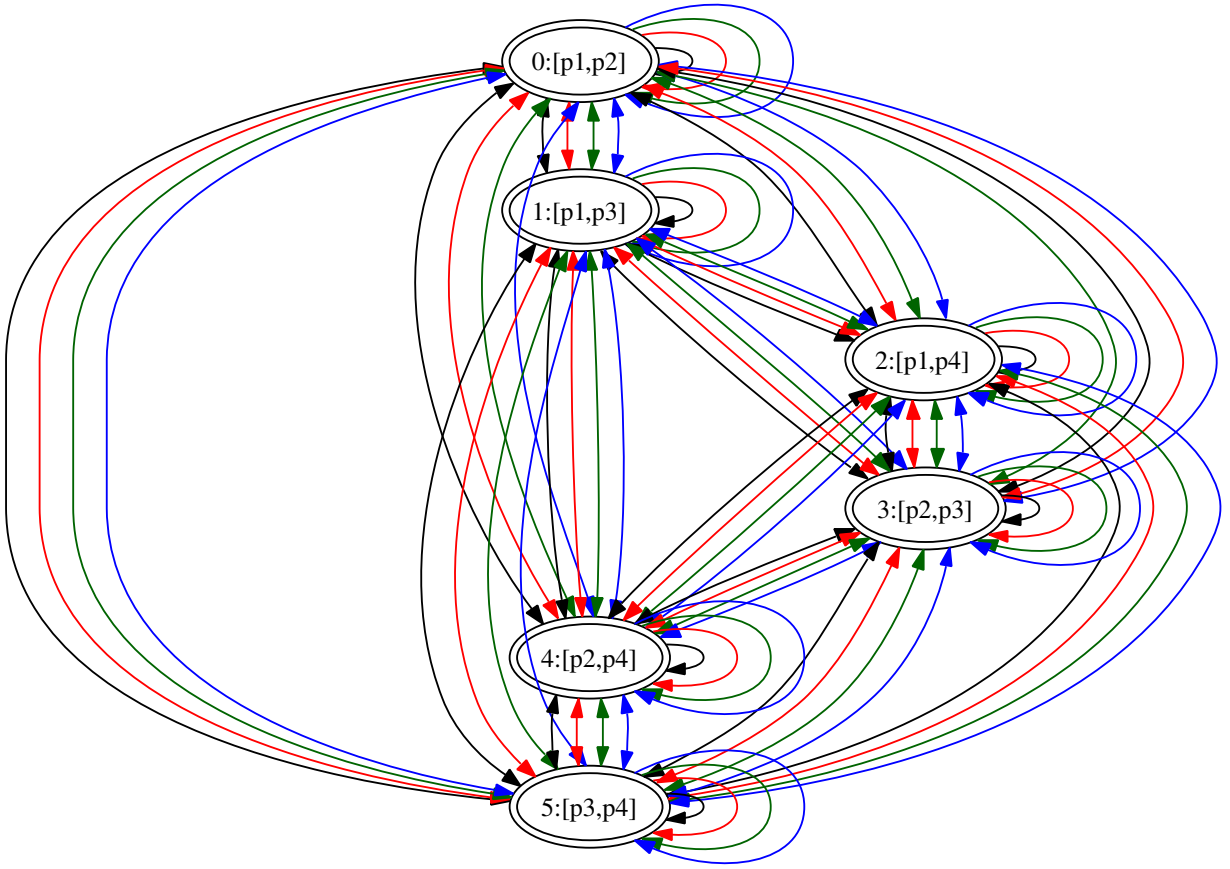
```
mo0 = initM [a,b,c,d] [P 1, P 2, P 3, P 4]
```

```
p1,p2,p3,p4 :: Form
p1 = Prop (P 1); p2 = Prop (P 2)
p3 = Prop (P 3); p4 = Prop (P 4)

capsInfo :: Form
capsInfo =
  Disj [Conj [f, g, Neg h, Neg j] |
              f <- [p1, p2, p3, p4],
              g <- [p1, p2, p3, p4] \\ [f],
              h <- [p1, p2, p3, p4] \\ [f,g],
              j <- [p1, p2, p3, p4] \\ [f,g,h],
              f < g, h < j                      ]
```

```
mo1 = upd mo0 (public capsInfo)
```

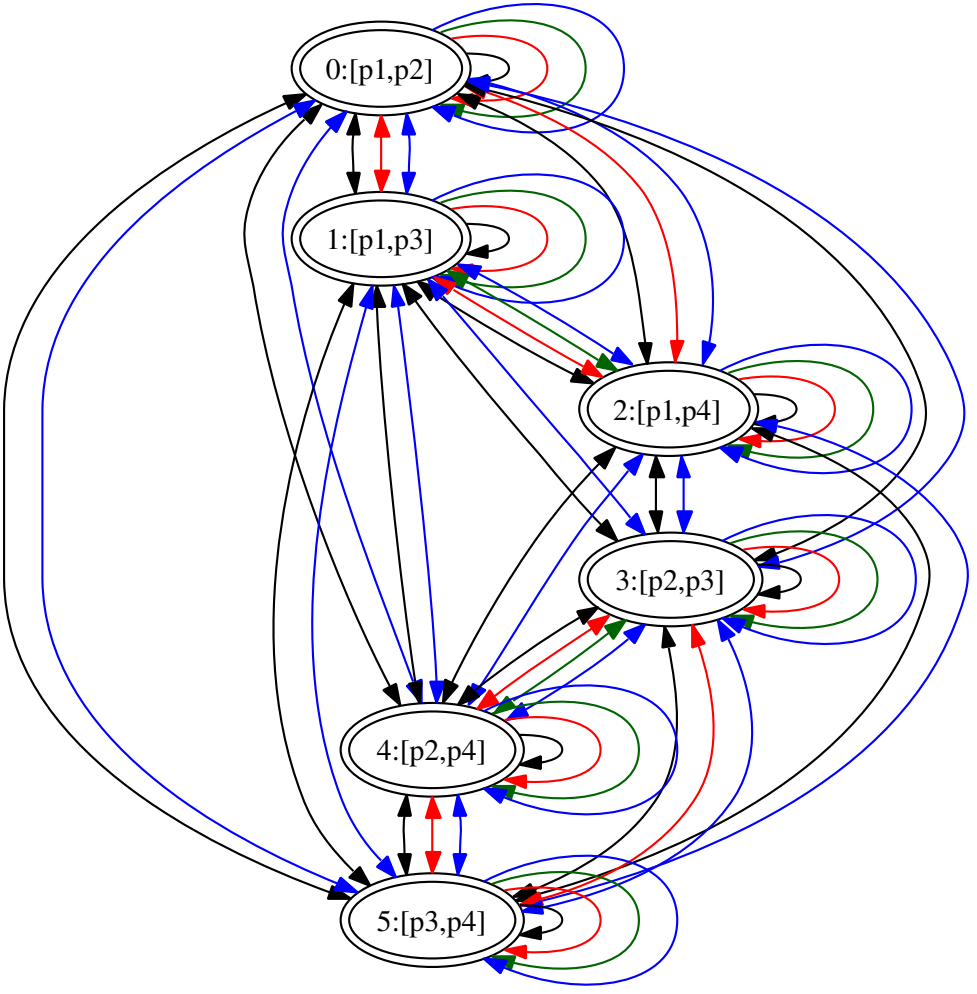
```
LAI14> displayS5 mo1
[0,1,2,3,4,5]
[(0,[p1,p2]),(1,[p1,p3]),(2,[p1,p4]),
 (3,[p2,p3]),(4,[p2,p4]),(5,[p3,p4])]
(a,[[0,1,2,3,4,5]])
(b,[[0,1,2,3,4,5]])
(c,[[0,1,2,3,4,5]])
(d,[[0,1,2,3,4,5]])
[0,1,2,3,4,5]
```


0:[p1,p2]
1:[p1,p3]
2:[p1,p4]
3:[p2,p3]
4:[p2,p4]
5:[p3,p4]

```
awarenessFirstCap  = info [b,c] p1
awarenessSecondCap = info [c]   p2

mo2  = upd (upd mo1 awarenessFirstCap)
                    awarenessSecondCap
```

```
LAI14> displayS5 mo2
[0,1,2,3,4,5]
[(0,[p1,p2]),(1,[p1,p3]),(2,[p1,p4]),
 (3,[p2,p3]),(4,[p2,p4]),(5,[p3,p4])]
(a,[[0,1,2,3,4,5]])
(b,[[0,1,2],[3,4,5]])
(c,[[0],[1,2],[3,4],[5]])
(d,[[0,1,2,3,4,5]])
[0,1,2,3,4,5]
```

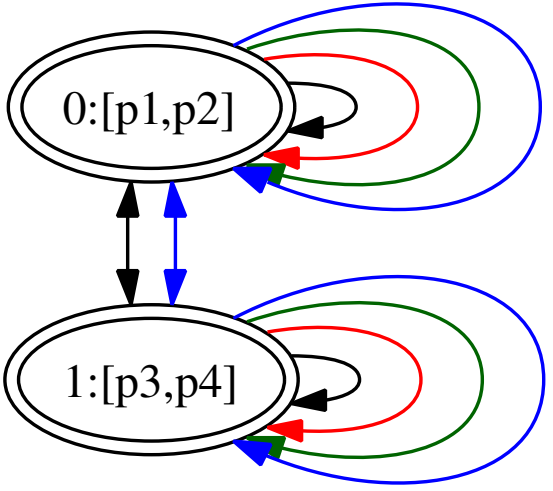

0:[p1,p2]
1:[p1,p3]
2:[p1,p4]
3:[p2,p3]
4:[p2,p4]
5:[p3,p4]

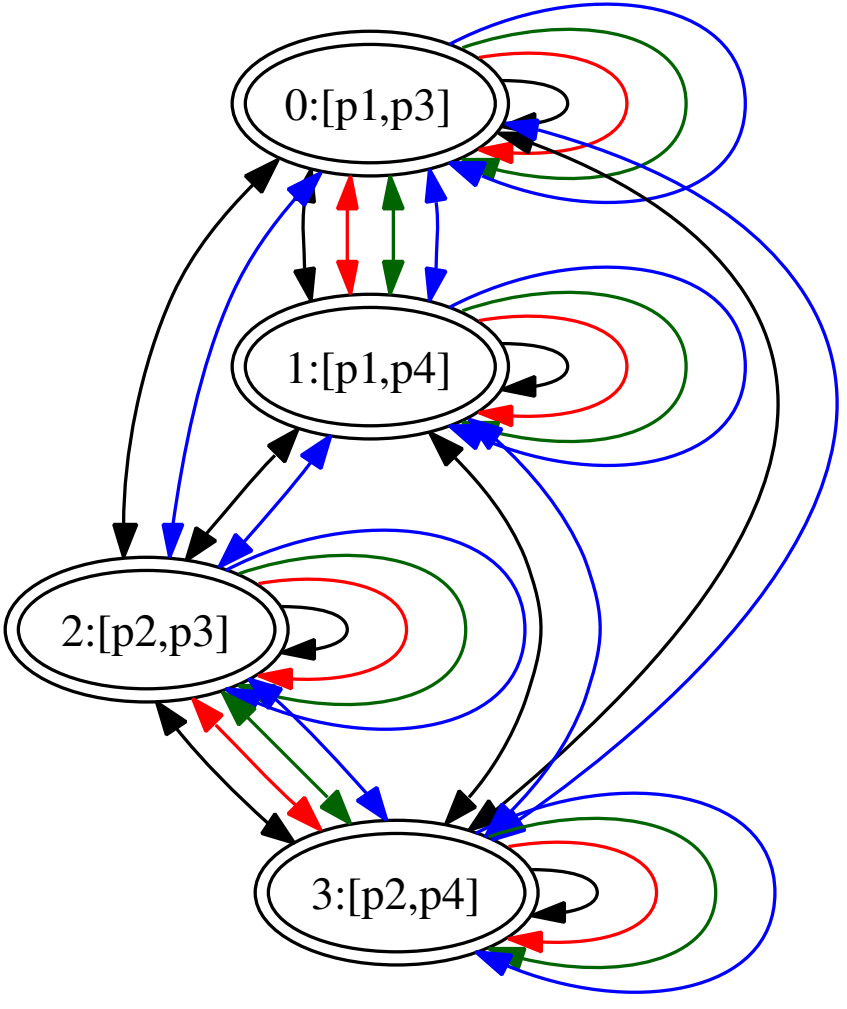
```
bK =  Disj [K b p2, K b (Neg p2)]
cK =  Disj [K c p3, K c (Neg p3)]

mo3a = upd  mo2 (public cK)
mo3b = upd  mo2 (public (Neg cK))
```

```
LAI14> displayS5 mo3a
[0,1]
[(0,[p1,p2]),(1,[p3,p4])]
(a,[[0,1]])
(b,[[0],[1]])
(c,[[0],[1]])
(d,[[0,1]])
[0,1]
```

```
LAI14> displayS5 mo3b
[0,1,2,3]
[(0,[p1,p3]),(1,[p1,p4]),(2,[p2,p3]),(3,[p2,p4])]
(a,[[0,1,2,3]])
(b,[[0,1],[2,3]])
(c,[[0,1],[2,3]])
(d,[[0,1,2,3]])
[0,1,2,3]
```


0:[p1,p2]
1:[p3,p4]


0:[p1,p3]
1:[p1,p4]
2:[p2,p3]
3:[p2,p4]

```
impl :: Form -> Form -> Form
impl form1 form2 = Disj [Neg form1, form2]

equiv :: Form -> Form -> Form
equiv form1 form2 =
  Conj [form1 `impl` form2, form2 `impl` form1]
```

```
test1 = isTrue mo3a bK
test2 = isTrue mo3b bK
test3 = isTrue mo3a (K a (equiv p1 p2))
test4 = isTrue mo3b (K a (equiv p1 (Neg p2)))
```

```
test1 = isTrue mo3a bK
test2 = isTrue mo3b bK
test3 = isTrue mo3a (K a (equiv p1 p2))
test4 = isTrue mo3b (K a (equiv p1 (Neg p2)))
```

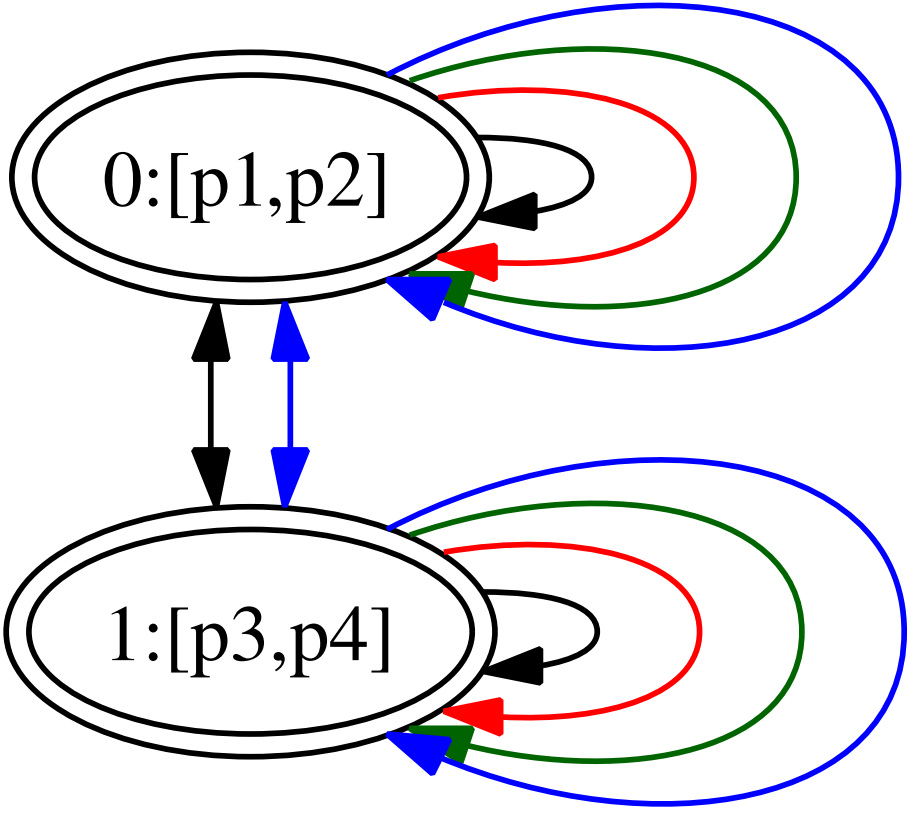
```
IEMC> test1
True
IEMC> test2
True
IEMC> test3
True
IEMC> test4
True
```

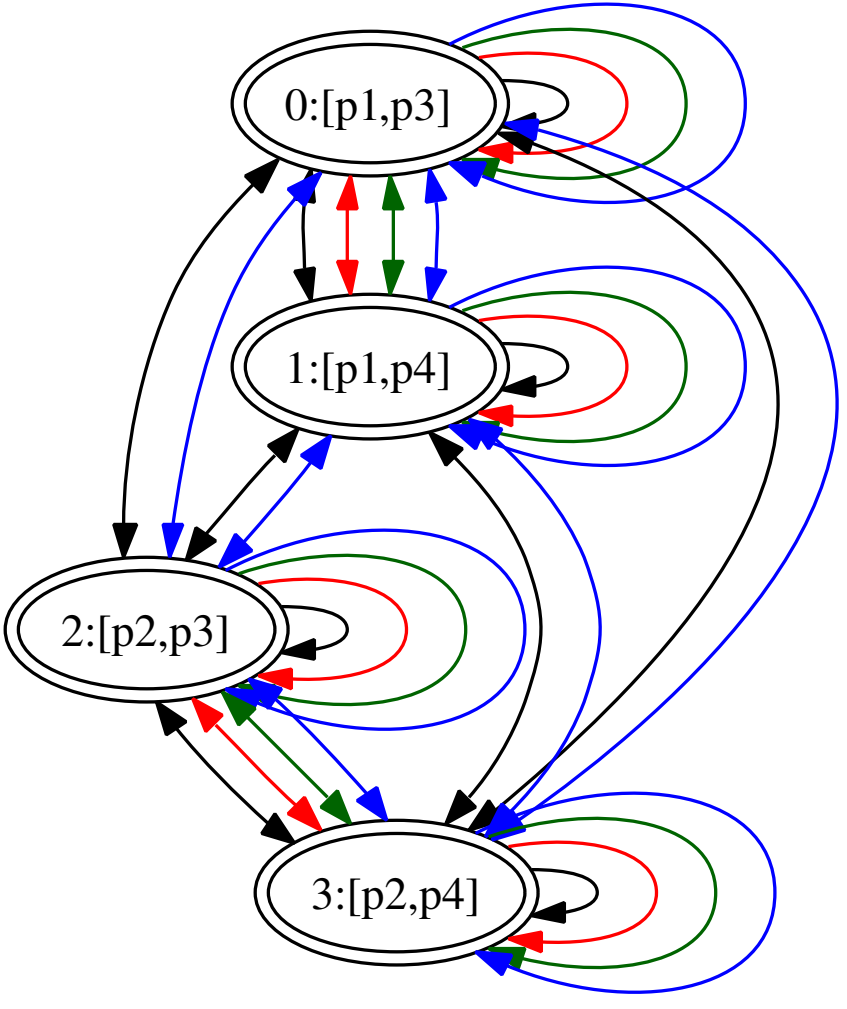
```
mo4a = upd mo3a (public bK)
mo4b = upd mo3b (public bK)
```

```
IEMC> displayS5 mo4a
[0,1]
[(0,[p1,p2]),(1,[p3,p4])]
(a,[[0,1]])
(b,[[0],[1]])
(c,[[0],[1]])
(d,[[0,1]])
[0,1]


IEMC> displayS5 mo4b
```

```
[0,1,2,3]
[(0,[p1,p3]),(1,[p1,p4]),(2,[p2,p3]),(3,[p2,p4])]
(a,[[0,1,2,3]])
(b,[[0,1],[2,3]])
(c,[[0,1],[2,3]])
(d,[[0,1,2,3]])
[0,1,2,3]
```


0:[p1,p2]
1:[p3,p4]


0:[p1,p3]
1:[p1,p4]
2:[p2,p3]
3:[p2,p4]

Muddy Children Puzzle

Your homework for today: implement the initial model for the case of three children, find the appropriate updates, and check the results.

Next, model-check the example of muddy children with factual change from Hans' lecture today.

References

[1] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. Information and Computation, 204(11):1620–1662, 2006.

[2] J. Gerbrandy. Bisimulations on planet Kripke. PhD thesis, ILLC, 1999.

[3] J. A. Plaza. Logics of public communications. In M. L. Emrich, M. S. Pfeifer, M. Hadzikadic, and Z. W. Ras, editors, Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems, pages 201–216, 1989.