

Modelling Effects of Communication

Jan van Eijck

jve@cwi.nl

January 18, 2006

Abstract

The goal of logics of communication is to allow reasoning about communication and knowledge change, and to model the effects of communication.

We will first implement a set of useful update action models, and next use these to analyse situations involving communication and knowledge change.

Module Declaration

```
module LAI14 where

import List
import Char
import LAI9
import LAI10
import LAI11
import LAI12
import LAI13
```

Recall ...

- AM is the type of action models,
- FAM is the type of functions from agent lists to action models,
- up is the update function, the function that computes a new epistemic model from an epistemic model and an action model.

Updating, ... and simplifying the result

Update, and take the bisimulation minimal version of the update result:

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

Implementing Communicative Actions

Now we need implementations of various communicative actions:

- public announcements,
- group messages,
- private (individual) messages,
- tests.

These are **communications that**.

(Below, we will also look at **communications whether**.)

Public Announcement

See [2, 1].

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 -> FAM State
public form ags =
  Am [0] ags [(0,form)] [(a,0,0) | a <- ags ] [0]
```

Example

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

```
LAI12> 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]
```

LAI12> displayS5 m0

$[(1,0), (3,0)]$

$[((1,0), [p]), ((3,0), [p,q])]$

$(a, [[(1,0)], [(3,0)]])$

$(b, [[(1,0)], [(3,0)]])$

$(c, [[(1,0), (3,0)]])$

$[(1,0)]$

LAI12> displayS5 m1

$[0,1]$

$[(0, [p]), (1, [p,q])]$

$(a, [[0], [1]])$

$(b, [[0], [1]])$

$(c, [[0,1]])$

$[0]$

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 -> FAM State
groupM gr form agents =
  if sort gr == sort agents
  then public form agents
  else
    (Am
     [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]
m2 = up e0 (groupM [a,b] (Neg p))
m3 = upd e0 (groupM [a,b] (Neg p))
```

```
LAI14> 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]
```

LAI14> displayS5 m2

$[(0,0), (0,1), (1,1), (2,0), (2,1), (3,1)]$

$[(0,0), []], [(0,1), []], [(1,1), [p]], [(2,0), [q]],$
 $[(2,1), [q]], [(3,1), [p,q]]]$

$(a, [[(0,0), (2,0)], [(0,1), (1,1), (2,1), (3,1)]]]$

$(b, [[(0,0), (2,0)], [(0,1), (1,1), (2,1), (3,1)]]]$

$(c, [[(0,0), (0,1), (1,1), (2,0), (2,1), (3,1)]]]$

$[(0,0), (2,0)]$

$[0,1,2,3]$

LAI14> displayS5 m3

$[0,1,2,3,4,5]$

$[(0, []), (1, []), (2, [p]), (3, [q]), (4, [q]), (5, [p,q])]$

$(a, [[0,3], [1,2,4,5]])$

$(b, [[0,3], [1,2,4,5]])$

$(c, [[0,1,2,3,4,5]])$

$[0,3]$

Private Messages

Private messages are a special case of group messages:

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

Example

```
m4 = up e0 (message c (Disj [p,q]))
m5 = upd e0 (message c (Disj [p,q]))
```

```
LAI14> displayS5 m4
```

```
[(0,1), (1,0), (1,1), (2,0), (2,1), (3,0), (3,1)]
[((0,1), []), ((1,0), [p]), ((1,1), [p]), ((2,0), [q]),
 ((2,1), [q]), ((3,0), [p,q]), ((3,1), [p,q])]
(a, [[(0,1), (1,0), (1,1), (2,0), (2,1), (3,0), (3,1)])]
(b, [[(0,1), (1,0), (1,1), (2,0), (2,1), (3,0), (3,1)])]
(c, [[(0,1), (1,1), (2,1), (3,1)], [(1,0), (2,0), (3,0)]]
[(1,0), (2,0), (3,0)]
```

```
LAI14> displayS5 m5
```

```
[0,1,2,3,4,5,6]
```

```
[(0, []), (1, [p]), (2, [p]), (3, [q]), (4, [q]),
```

```
  (5, [p,q]), (6, [p,q])]
```

```
(a, [[0,1,2,3,4,5,6]])
```

```
(b, [[0,1,2,3,4,5,6]])
```

```
(c, [[0,2,4,6], [1,3,5]])
```

```
[1,3,5]
```

Tests

Tests are another special case of group messages:

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

Example

```
m6 = up e0 (test (Neg p))  
m7 = upd e0 (test (Neg p))
```

```
LAI14> displayS5 m6
```

```
[(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]), ((3,1), [p,q])]
```

```
(a, [[(0,0), (0,1), (1,1), (2,0), (2,1), (3,1)])]
```

```
(b, [[(0,0), (0,1), (1,1), (2,0), (2,1), (3,1)])]
```

```
(c, [[(0,0), (0,1), (1,1), (2,0), (2,1), (3,1)])]
```

```
[(0,0), (2,0)]
```

```
LAI14> displayS5 m7
```

```
[0,1,2,3]
```

```
[(0, []), (1, [p]), (2, [q]), (3, [p, q])]
```

```
(a, [[0,1,2,3]])
```

```
(b, [[0,1,2,3]])
```

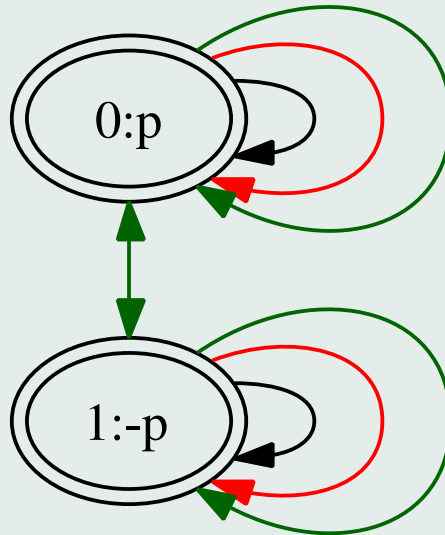
```
(c, [[0,1,2,3]])
```

```
[0,2]
```

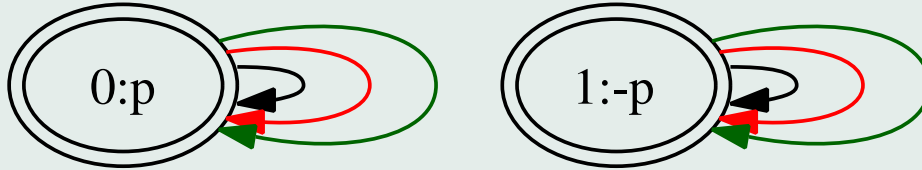
Communications Whether

- informing everyone **whether** φ ,
- informing a group **whether** φ ,
- informing an individual **whether** φ .

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 -> FAM State
info ags form agents =
  Am
  [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 \\ ags
```

Example

```
m8 = up e0 (info [a,b] p)
m9 = upd e0 (info [a,b] p)
```

```
LAI14> displayS5 m8
```

```
[(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)]
```

```
LAI14> displayS5 m9
```

```
[0,1,2,3]
```

```
[(0, []), (1, [p]), (2, [q]), (3, [p, q])]
```

```
(a, [[0,2], [1,3]])
```

```
(b, [[0,2], [1,3]])
```

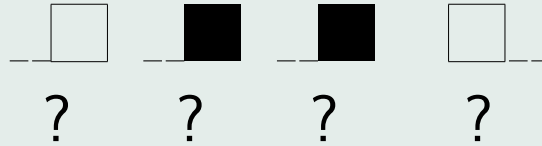
```
(c, [[0,1,2,3]])
```

```
[0,1,2,3]
```

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])]
```

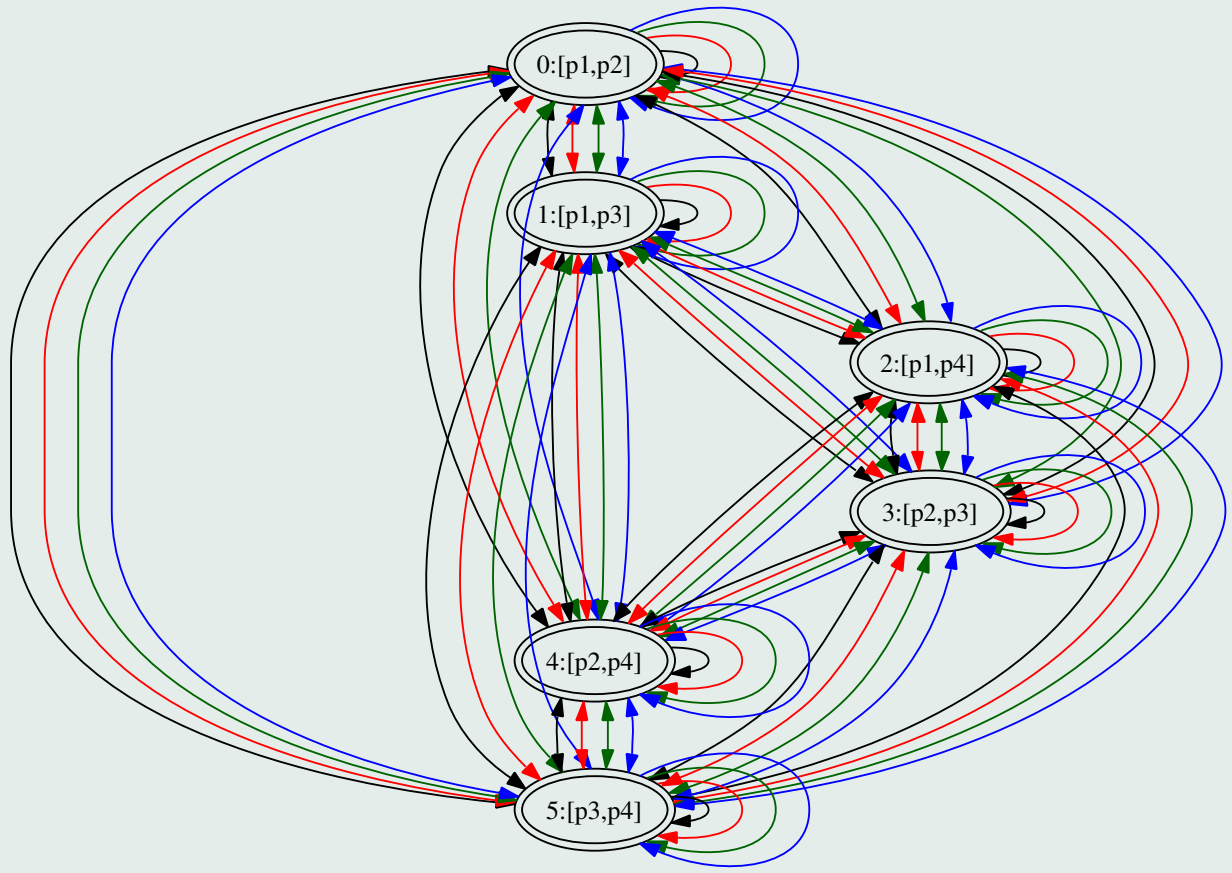
```
(a, [[0,1,2,3,4,5]])
```

```
(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]
```

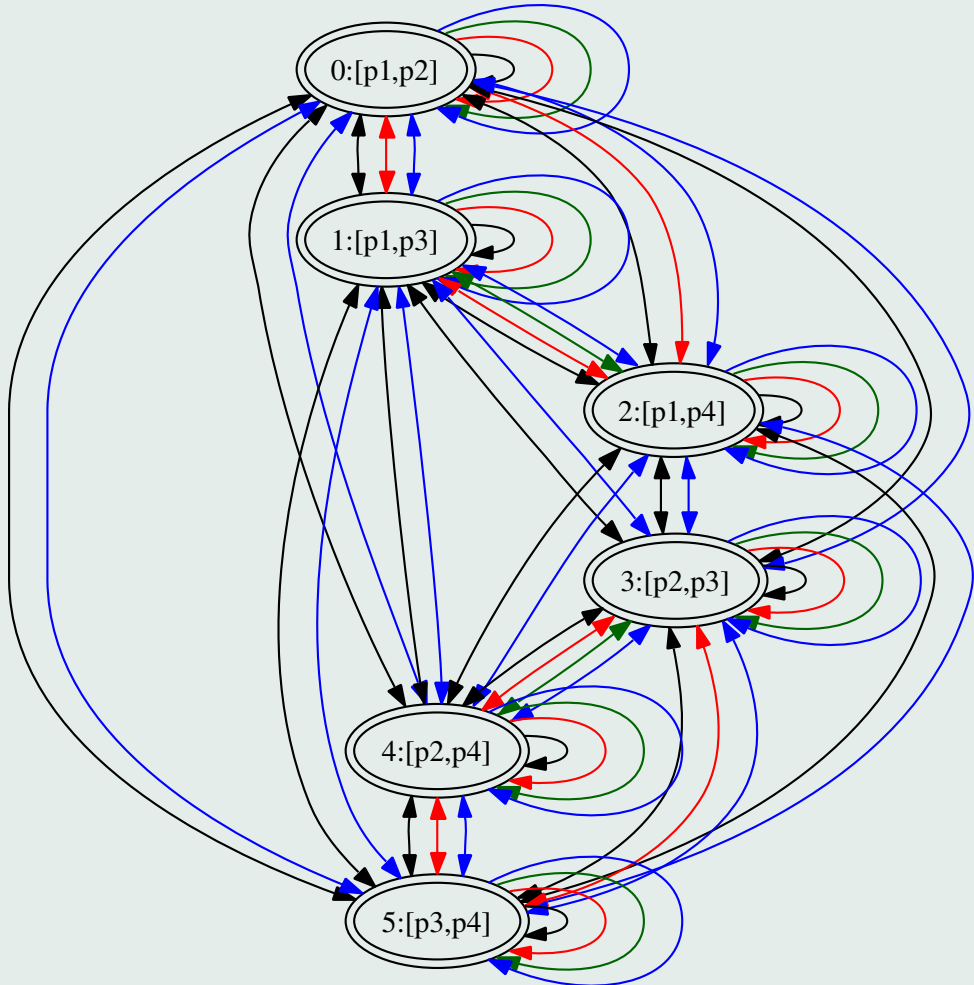


```
awarenessFirstCap = info [b,c] p1
```

```
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]
```



```
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])]
```

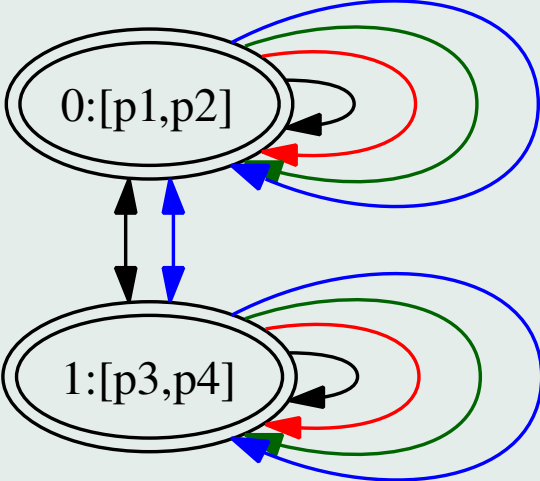
```
(a, [[0,1,2,3]])
```

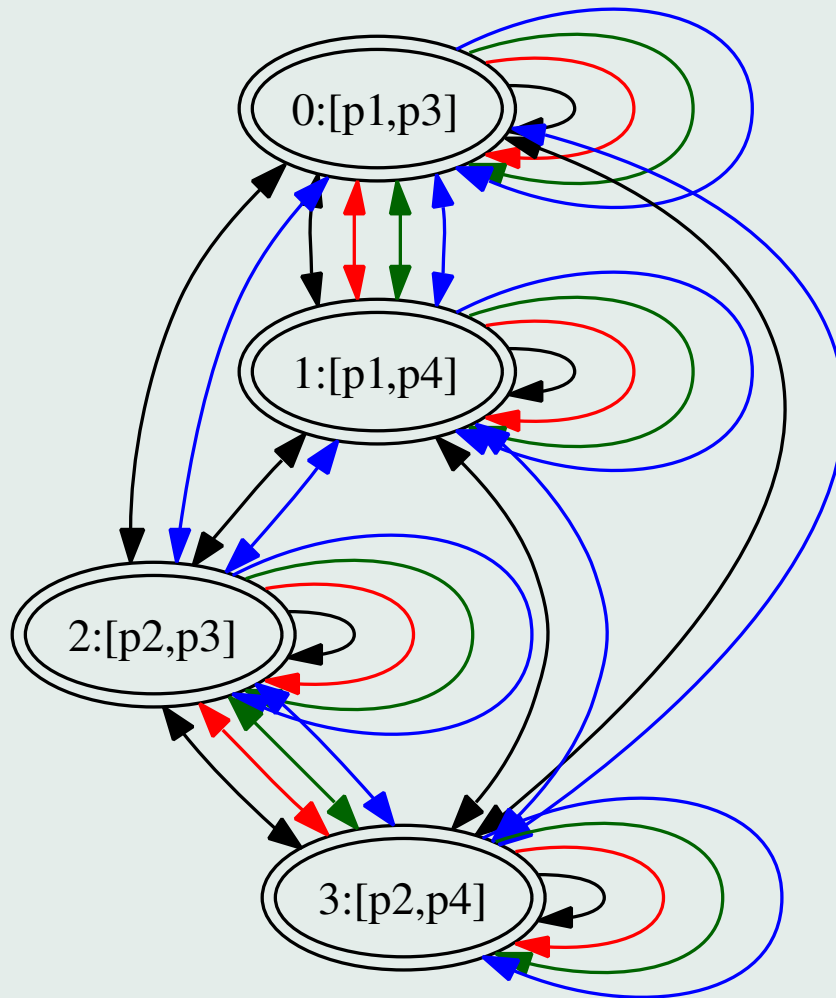
```
(b, [[0,1], [2,3]])
```

```
(c, [[0,1], [2,3]])
```

```
(d, [[0,1,2,3]])
```

```
[0,1,2,3]
```





```
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 p2))
```

```
LAI14> test1
```

```
True
```

```
LAI14> test2
```

```
True
```

```
LAI14> test3
```

```
True
```

```
LAI14> test4
```

```
False
```

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

```
LAI14> displayS5 mo4a
```

```
[0,1]
```

```
[(0, [p1,p2]), (1, [p3,p4])]
```

```
(a, [[0,1]])
```

```
(b, [[0],[1]])
```

```
(c, [[0],[1]])
```

```
(d, [[0,1]])
```

```
[0,1]
```

```
LAI14> displayS5 mo4b
```

[0,1,2,3]

[(0, [p1,p3]), (1, [p1,p4]), (2, [p2,p3]), (3, [p2,p4])]

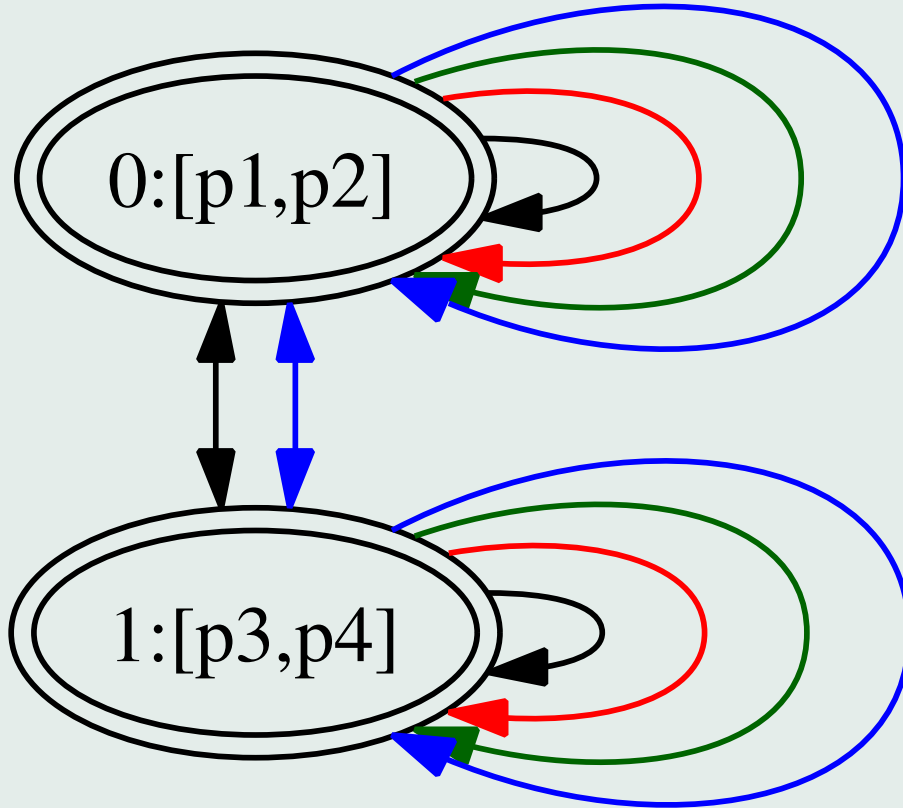
(a, [[0,1,2,3]])

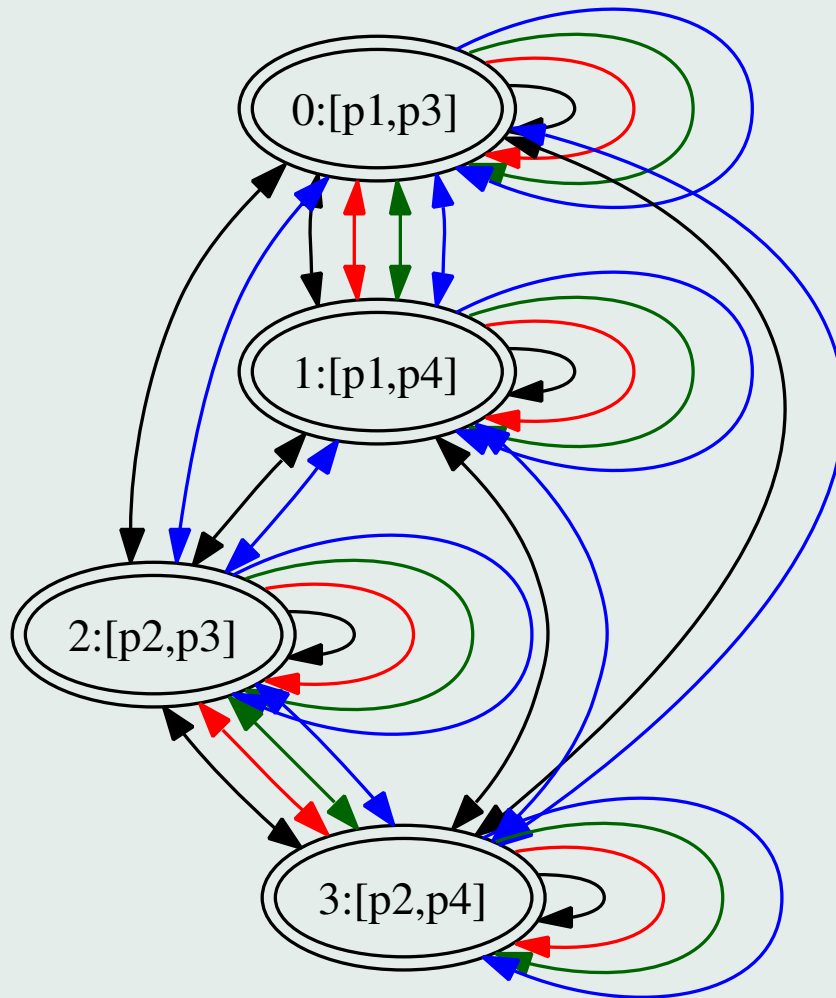
(b, [[0,1], [2,3]])

(c, [[0,1], [2,3]])

(d, [[0,1,2,3]])

[0,1,2,3]





References

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