

Action Models and Action Updates

Jan van Eijck

jve@cwi.nl

January 16, 2006

Abstract

Action models are a special kind of Kripke models, where the valuations are replaced by precondition formulas.

Communicative actions can be modelled as updates with action models.

Action Models

Action models are like Kripke models, but with the valuation function replaced by a **precondition function**.

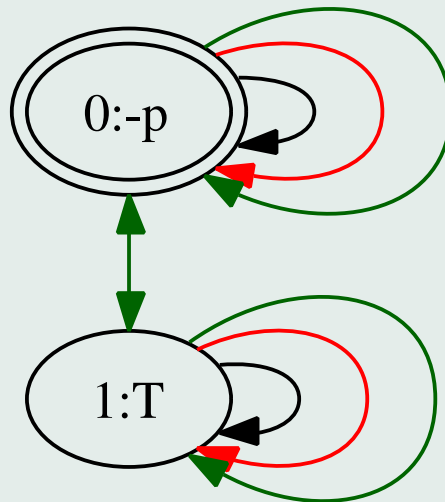
Their components are called **actions**.

If an action has precondition φ then the action can be applied to worlds where φ holds.

See [1, 2].

Representing Updates as Action Models

Suppose there are three agents, **a**, **b**, **c**.



Which update is this?

Updating with an Action Model

The result of updating with an action model is defined as the product of the epistemic model and the action model, restricted to the pairs (w, u) where

- w satisfies the precondition of action u , and
- the accessibility relations hold between pairs (w, u) and (w', u') just in case they hold both between w and w' and between u and u' .

The valuation of (v, u) in the new model is that of v in the old epistemic model.

See [1, 2].

Update Execution: Formal Definition

Given an epistemic model

$$\mathbf{M} = (W, V, R, U)$$

where $U \subseteq W$ is the set of actual worlds, and an action model

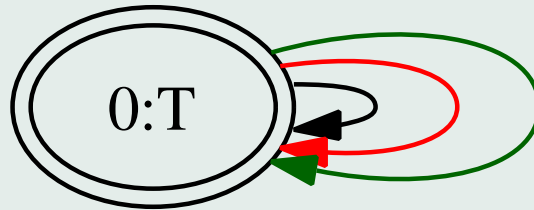
$$A = (E, \text{pre}, R, F)$$

where $F \subseteq E$ is the set of actual actions, we say that the result of **executing** A , **s in** M , w is the model $M \circ A = (W', V', R', U')$ where

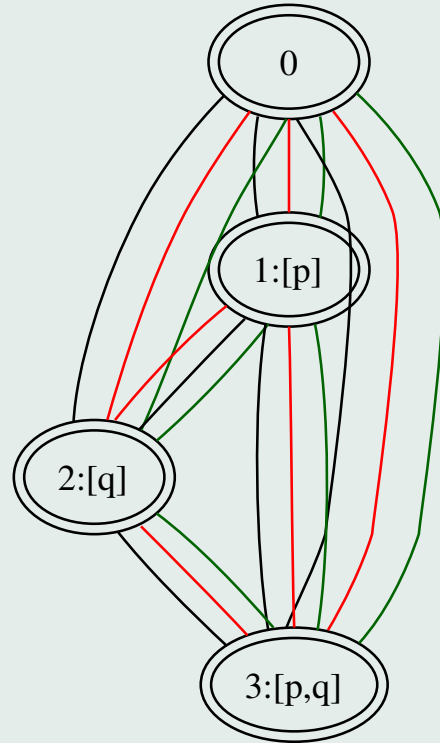
- $W' = \{(v, t) \mid M, v \models \text{pre}(t)\}$,
- $V'(v, b) = V(v)$,
- $R'(a) = \{((v, t), (u, u)) \mid (v, u) \in R(a) \text{ and } (t, u) \in R(a)\}$,
- $U' = \{(v, t) \in W' \mid v \in U, t \in F\}$.

The Action that Changes Nothing

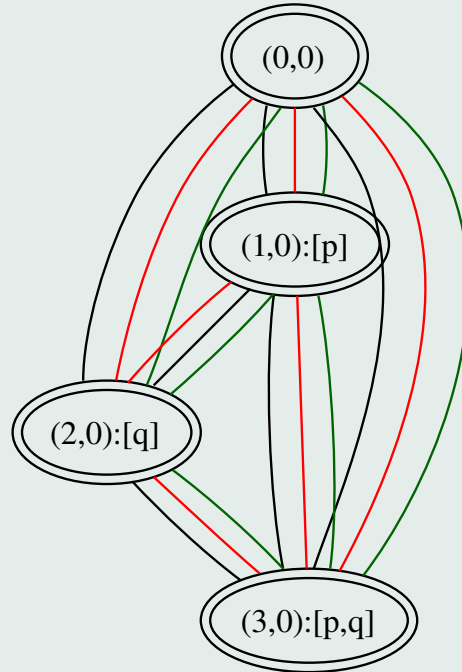
Suppose there are three agents, **a**, **b**, **c**.



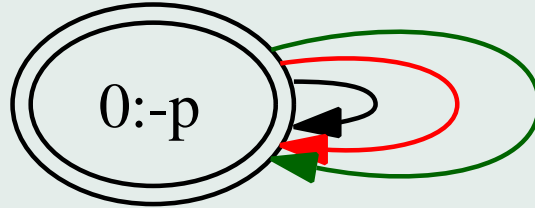
Blissful Ignorance



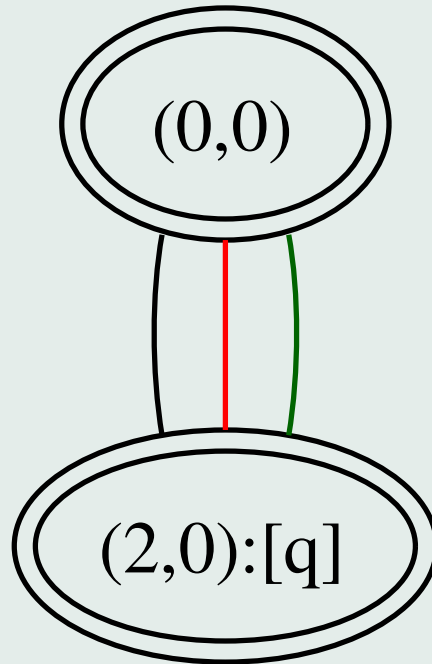
After Update with 'The Action That Changes Nothing'



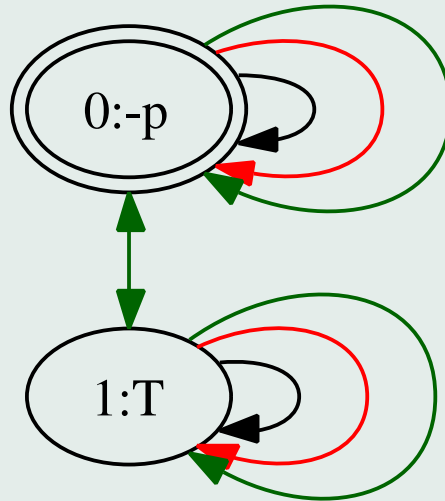
Public Announcement of $\neg p$



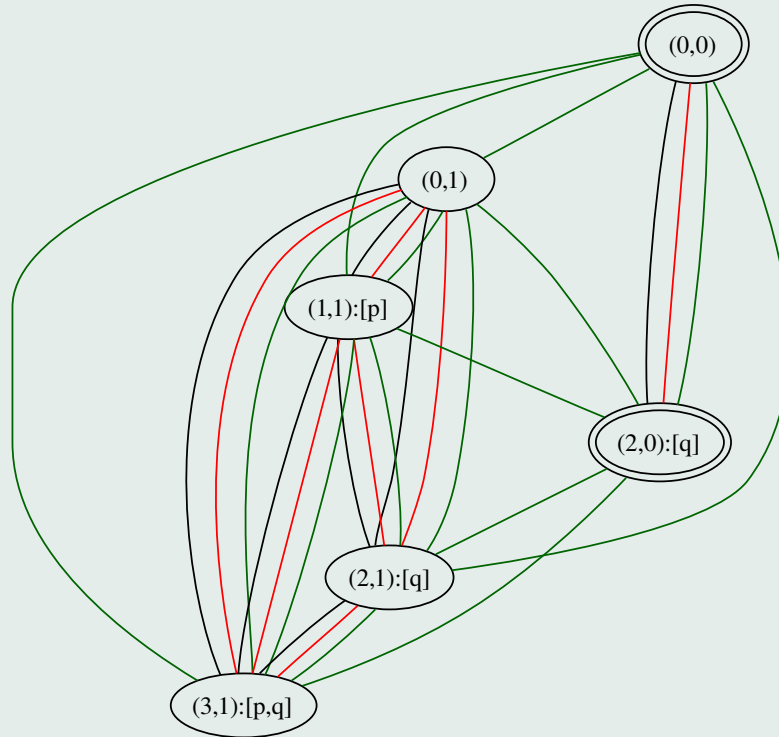
After Update with Public Announcement of $\neg p$



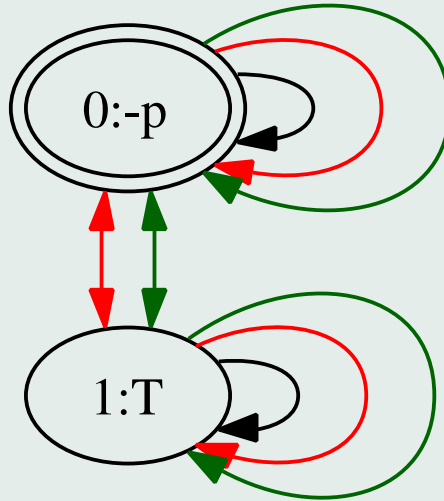
Group Message to a, b that $\neg p$



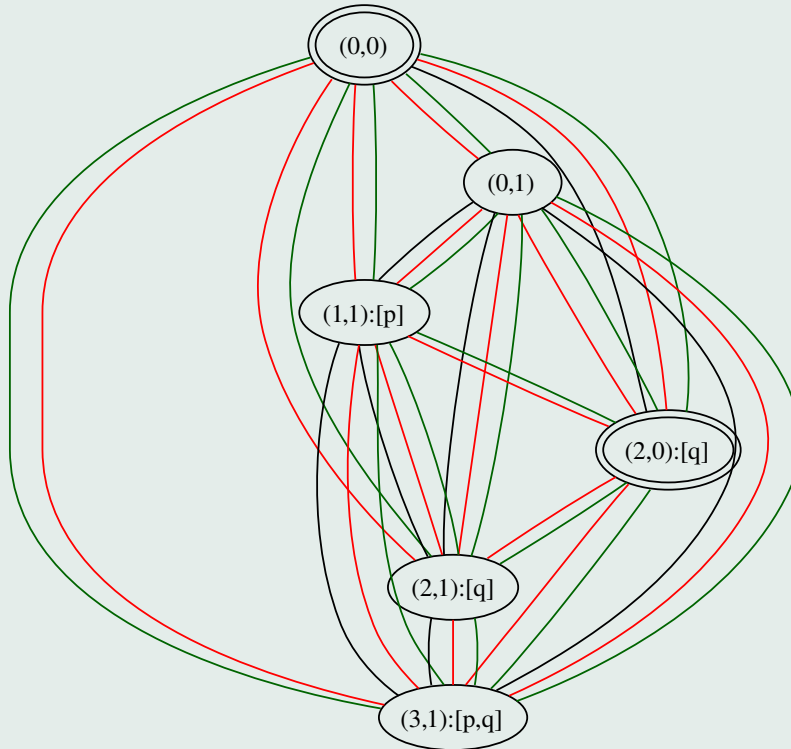
After Update with Group Message to a, b that $\neg p$



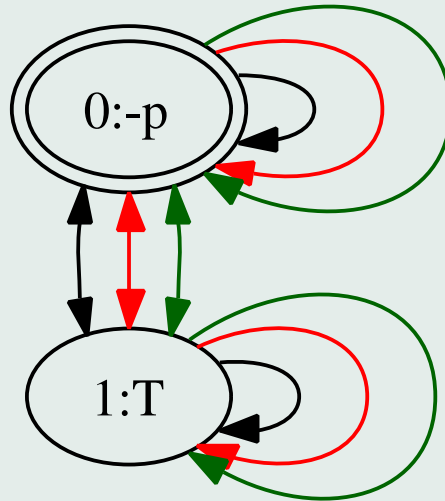
Private Message to a that $\neg p$



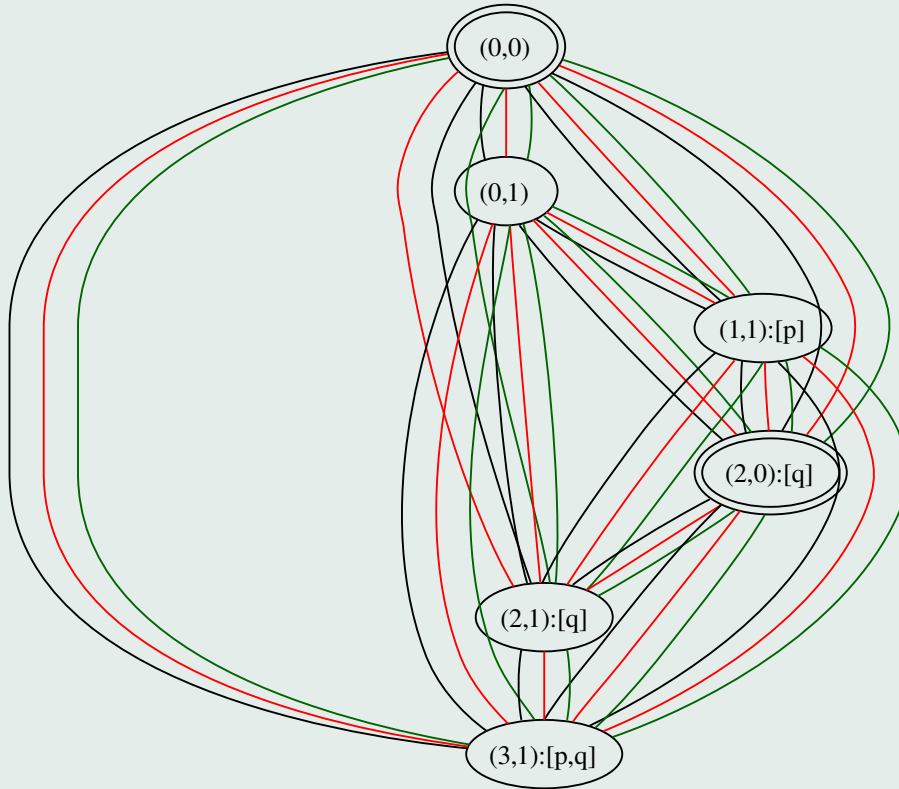
After Update with Private Message to a that $\neg p$



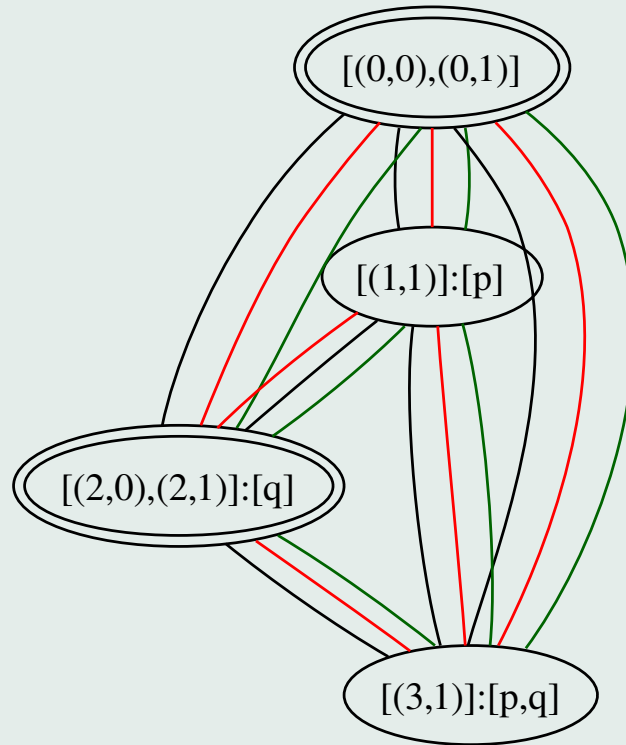
Test that $\neg p$



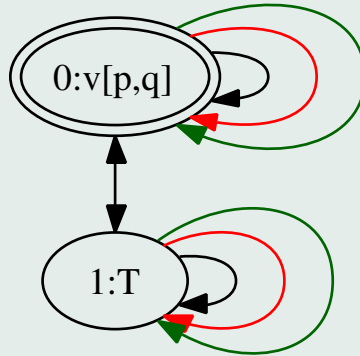
After Update with Test that $\neg p$



Bisimulation Minimal Version



Group Message to b, c that $p \vee q$



Public Announcement, Group Announcement, Private Message, Test

- All special cases of Group Announcements.
- Public Announcement: Group Announcement to All Agents
- Private Message: 'Group' Announcement to a Single Agent
- Test: Group Announcement to the Empty Group

Module Declaration

```
module LAI13 where
```

```
import List
```

```
import Char
```

```
import LAI9
```

```
import LAI10
```

```
import LAI11
```

```
import LAI12
```

Haskell Code for Cartesian Product — For Building Confidence

How to implement the cartesian product of two sets as a relation (set of pairs)?

Representing the sets as lists, we get the following type specification:

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

Next, recall the definition:

$$X \times Y =_{\text{def}} \{(x, y) \mid x \in X, y \in Y\}$$

```
cartProduct xs ys = [ (x,y) | x <- xs, y <- ys]
```

How to Test This?

```
LAI13> cartProduct [1..3] [4..6]
```

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

Haskell Code for n -fold Product

Definition:

$$\begin{aligned}R^1 &= R \\ R^{n+1} &= R \circ R^n\end{aligned}$$

Type declaration:

```
composeR :: Eq a => Rel a -> Int -> Rel a
```

We need the equality class because it is required in the definition of relational composition.

Follow the definition as closely as possible:

```
composeR r 1 = r
composeR r (n+1) = r @@ (composeR r n)
```

How to Test This?

Like this:

```
*LAI13> cf2list (\ x y -> y == succ x) [1..10]
[(1,2), (2,3), (3,4), (4,5), (5,6), (6,7), (7,8), (8,9), (9,10)]
*LAI13> composeR (cf2list (\ x y -> y == succ x) [1..10]) 2
[(1,3), (2,4), (3,5), (4,6), (5,7), (6,8), (7,9), (8,10)]
*LAI13> composeR (cf2list (\ x y -> y == succ x) [1..10]) 3
[(1,4), (2,5), (3,6), (4,7), (5,8), (6,9), (7,10)]
*LAI13> composeR (cf2list (\ x y -> y == succ x) [1..10]) 4
[(1,5), (2,6), (3,7), (4,8), (5,9), (6,10)]
*LAI13> composeR (cf2list (\ x y -> y == succ x) [1..10]) 5
[(1,6), (2,7), (3,8), (4,9), (5,10)]
```

Datatype for Action Models

```
data AM state = Am
    [state]
    [Agent]
    [(state,Form)]
    [(Agent,state,state)]
    [state] deriving (Eq,Show)
```

Action Models

In updating with an action model, we will have to make sure that the set of agents of the action model is the same as that of the epistemic model that gets updated.

Therefore, it makes sense to update with functions from agents to action models:

```
type FAM state = [Agent] -> AM state
```

Updating with an Action Model

```
up :: (Eq state, Ord state) =>
      EpistM state -> FAM state
      -> EpistM (state,state)
```

```

up m@(Mo worlds ags val rel points) fam =
  Mo worlds' ags' val' rel' points'
  where
    Am states ags' pre susp actuals = fam ags
    worlds' = [ (w,s) | w <- worlds, s <- states,
                 isTrueAt m w (apply pre s)    ]
    val'     = [ ((w,s),props) | (w,props) <- val,
                               s           <- states,
                               elem (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 <- actuals,
                 elem (p,a) worlds'      ]

```

Next Time

Implementing communicative actions.

References

- [1] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. Technical report, Dept of Cognitive Science, Indiana University and Dept of Computing, Oxford University, 2003.
- [2] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. Under submission, 2005. Available from www.cwi.nl/~jve/papers/05/lcc/.