

Epistemic Propositional Dynamic Logic With an Appendix on Confusions, Secrets and Lies

Jan van Eijck

jve@cwi.nl

January 25, 2006

Abstract

Baltag, Moss and Solecki [1] showed how to treat the effects of epistemic actions like public announcements, group messages, private messages, and so on, in a common framework, in terms of a very general notion of updating with action models, of epistemic states. Van Benthem, Van Eijck and Kooi [2] showed how dynamic epistemic logic can be streamlined and simplified and brought within the framework of epistemic propositional dynamic logic (PDL). The key idea here is **reduction**.

Internal Structure of Accessibilities

We have already seen that the accessibility relation for common knowledge (or common belief) among a group of agents

$$B = \{b_1, \dots, b_n\}$$

has internal structure: it is the relation

$$(R_{b_1} \cup \dots \cup R_{b_n})^*.$$

It is built in terms of relation union and reflexive transitive closure.

Similarly, reflexive transitive closure can be viewed as an infinite union of compositions:

$$R^* = I \cup R \cup (R \circ R) \cup (R \circ R \circ R) \cup \dots$$

Diagonal Relation, Relativized Common Knowledge Relation

If C is a set, the diagonal relation based on C , notation $\Delta(C)$, is given by:

$$\Delta(C) = \{(x, x) \mid x \in C\}.$$

If φ is a formula, $\llbracket\varphi\rrbracket$ denotes the set of all worlds in a model where the formula is true. The relation $\Delta(\llbracket\varphi\rrbracket)$ is the **test relation** defined by φ .

Relativized common knowledge (or relativized common belief) uses test, composition and reflexive transitive closure. The relativized common knowledge operator $C_B(\varphi, -)$ is interpreted as the following accessibility (assuming $B = \{b_1, \dots, b_n\}$):

$$\Delta(\llbracket\varphi\rrbracket) \circ ((R_{b_1} \cup \dots \cup R_{b_n}) \circ \Delta(\llbracket\varphi\rrbracket))^*$$

This defines all paths of B steps along worlds where φ is true.

Equivalent way of expressing this:

$$(\Delta(\llbracket \varphi \rrbracket) \circ (R_{b_1} \cup \dots \cup R_{b_n}))^* \circ \Delta(\llbracket \varphi \rrbracket)$$

Reduction Again, for a Language with Structured Relations

The logic of test, choice and sequence:

$$\varphi ::= \perp \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\pi]\varphi_2$$

$$\pi ::= a \mid \varphi? \mid \pi_1 \cup \pi_2 \mid \pi_1; \pi_2$$

Note: the relations have internal structure.

Let a model $\mathbf{M} = (W, V, R, U)$ be given. Semantics:

$$\llbracket \perp \rrbracket^{\mathbf{M}} = \emptyset$$

$$\llbracket p \rrbracket^{\mathbf{M}} = \{w \in W \mid p \in V(w)\}$$

$$\llbracket \neg \varphi \rrbracket^{\mathbf{M}} = W - \llbracket \varphi \rrbracket^{\mathbf{M}}$$

$$\llbracket \varphi_1 \wedge \varphi_2 \rrbracket^{\mathbf{M}} = \llbracket \varphi_1 \rrbracket^{\mathbf{M}} \cap \llbracket \varphi_2 \rrbracket^{\mathbf{M}}$$

$$\llbracket [\pi] \varphi \rrbracket^{\mathbf{M}} = \{w \in W \mid \forall v (\text{if } (w, v) \in [\pi]^{\mathbf{M}} \text{ then } v \in \llbracket \varphi \rrbracket^{\mathbf{M}})\}$$

$$\llbracket a \rrbracket^{\mathbf{M}} = R_a$$

$$\llbracket \varphi? \rrbracket^{\mathbf{M}} = \{(w, w) \in W \times W \mid w \in \llbracket \varphi \rrbracket^{\mathbf{M}}\}$$

$$\llbracket [\pi_1; \pi_2] \rrbracket^{\mathbf{M}} = \llbracket \pi_1 \rrbracket^{\mathbf{M}} \circ \llbracket \pi_2 \rrbracket^{\mathbf{M}}$$

$$\llbracket [\pi_1 \cup \pi_2] \rrbracket^{\mathbf{M}} = \llbracket \pi_1 \rrbracket^{\mathbf{M}} \cup \llbracket \pi_2 \rrbracket^{\mathbf{M}}$$

This logic is reducible to multimodal logic. How?

Via translation:

$$\begin{aligned}\perp^\bullet &:= \perp \\ p^\bullet &:= p \\ (\neg\varphi)^\bullet &:= \neg\varphi^\bullet \\ (\varphi_1 \wedge \varphi_2)^\bullet &:= \varphi_1^\bullet \wedge \varphi_2^\bullet \\ ([a]\varphi)^\bullet &:= [a]\varphi^\bullet \\ ([\varphi_1?]\varphi_2)^\bullet &:= \varphi_1^\bullet \rightarrow \varphi_2^\bullet \\ ([\pi_1 \cup \pi_2]\varphi)^\bullet &:= ([\pi_1]\varphi)^\bullet \wedge ([\pi_2]\varphi)^\bullet \\ ([\pi_1; \pi_2]\varphi)^\bullet &:= ([\pi_1][\pi_2]\varphi)^\bullet.\end{aligned}$$

The translation suggests a set of reduction axioms, yielding an easy completeness proof.

Logics for Epistemic Updates: Public Announcement Logic

The logic of public announcements:

$$\varphi ::= \perp \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid K_a\varphi \mid [\varphi_1!]\varphi_2$$

Again, translation yields an easy completeness proof:

A sound and complete proof system for PAL is that for multi-modal S5 epistemic logic plus the following reduction axioms:

- At** $[\varphi!]p \leftrightarrow (\varphi \rightarrow p)$ (atoms)
- PF** $[\varphi_1!]\neg\varphi_2 \leftrightarrow (\varphi_1 \rightarrow \neg[\varphi_1!]\varphi_2)$ (partial functionality)
- Dist** $[\varphi_1!](\varphi_2 \wedge \varphi_3) \leftrightarrow ([\varphi_1!]\varphi_2 \wedge [\varphi_1!]\varphi_3)$ (distribution)
- KA** $[\varphi_1!]K_a\varphi_2 \leftrightarrow (\varphi_1 \rightarrow K_a[\varphi_1!]\varphi_2)$ (knowledge-announcement)

as well as an inference rule of necessitation for all announcement modalities.

Problem: Reduction method breaks down if we add common knowledge

The logic of public announcements with common knowledge :

$$\varphi ::= \perp \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid K_a\varphi \mid C_N\varphi \mid [\varphi_1!]\varphi_2$$

Interpretation of $C_N\varphi$:

$$M, w \models C_N\varphi \text{ iff for all } v \text{ with } (w, v) \in R, M, v \models \varphi, \\ \text{where } R = (\bigcup_{a \in N} R_a)^*.$$

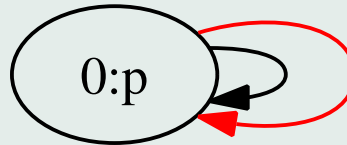
This does **not** reduce to multimodal logic.

Why not?

Theorem 1 *The Logic of Public Announcement with Common Knowledge has Greater Expressive Power than Multimodal Logic.*

Proof:

Consider the formula $C_{\{a,b\}}p$. The following model makes this true.



Suppose there is a multimodal formula φ equivalent with $C_{\{a,b\}}p$. Define the **modal depth** of φ by means of:

$$d(\perp) = d(p) = 0$$

$$d(\neg\varphi) = d(\varphi)$$

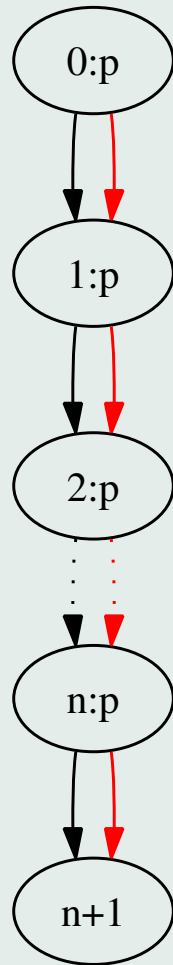
$$d(\varphi_1 \wedge \varphi_2) = \max(d(\varphi_1), d(\varphi_2))$$

$$d(K_a\varphi) = d(\varphi) + 1$$

It can be proved by induction that evaluation of a formula of depth n at a world w involves only worlds at 'distance' at most n from w .

Let n be the modal depth of φ .

Then φ will not see the difference between state 0 in the model above and state 0 in the following model, but $C_{\{a,b\}}p$ will be false in state 0 in the new model. Contradiction with the assumption that $C_{\{a,b\}}$ and φ are equivalent.



Still, common knowledge captures the essence to what goes in in multi-agent epistemic logic.

Solution: strengthen the base logic.

Epistemic PDL

$$\varphi ::= \perp \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\pi]\varphi$$

$$\pi ::= a \mid \varphi? \mid \pi_1; \pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^*$$

This language is well-known from computer science. It was defined by Pratt in [9, 10] as a generic language for reasoning about computation. Axiomatisations were given independently by Segerberg [11], Fisher/Ladner [5], and Parikh [8].

Note that this is **stronger** than multimodal logic, since $C_{\{a,b\}}p$ is expressible as $[(a \cup b)^*]p$.

Epistemic PDL with Updates

$$\varphi ::= \perp \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\pi]\varphi \mid [A, \mathbf{s}]\varphi$$

$$\pi ::= a \mid \varphi? \mid \pi_1; \pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^*$$

- the individual knowledge operator $K_a\varphi$ (or $\Box_a\varphi$) takes the shape $[a]\varphi$,
- the general knowledge operator $E_B\varphi$ takes the shape $[B]\varphi$, where B is shorthand for $b_1 \cup b_2 \cup \dots \cup b_n$.
- the common knowledge operator $C_B\varphi$ takes the shape $[B^*]\varphi$,
- $[A, \mathbf{s}]\varphi$ describes worlds w with the property that after an update with action model A , φ holds in (w, \mathbf{s}) .

Let $\mathbf{M} = (W, V, R, U)$ and $A = (E, \text{pre}, R, F)$.

$$\llbracket \perp \rrbracket^{\mathbf{M}} = \emptyset$$

$$\llbracket p \rrbracket^{\mathbf{M}} = \{w \in W \mid p \in V(w)\}$$

$$\llbracket \neg\varphi \rrbracket^{\mathbf{M}} = W - \llbracket \varphi \rrbracket^{\mathbf{M}}$$

$$\llbracket \varphi_1 \wedge \varphi_2 \rrbracket^{\mathbf{M}} = \llbracket \varphi_1 \rrbracket^{\mathbf{M}} \cap \llbracket \varphi_2 \rrbracket^{\mathbf{M}}$$

$$\llbracket [\pi]\varphi \rrbracket^{\mathbf{M}} = \{w \in W \mid \forall v (\text{if } (w, v) \in [\pi]^{\mathbf{M}} \text{ then } v \in \llbracket \varphi \rrbracket^{\mathbf{M}})\}$$

$$\llbracket [A, s]\varphi \rrbracket^{\mathbf{M}} = \{w \in W \mid \text{if } \mathbf{M}, w \models \text{pre}(s) \text{ then } (w, s) \in \llbracket \varphi \rrbracket^{\mathbf{M} \circ A}\}$$

$$\llbracket a \rrbracket^{\mathbf{M}} = R(a)$$

$$\llbracket \varphi? \rrbracket^{\mathbf{M}} = \{(w, w) \in W \times W \mid w \in \llbracket \varphi \rrbracket^{\mathbf{M}}\}$$

$$\llbracket \pi_1; \pi_2 \rrbracket^{\mathbf{M}} = \llbracket \pi_1 \rrbracket^{\mathbf{M}} \circ \llbracket \pi_2 \rrbracket^{\mathbf{M}}$$

$$\llbracket \pi_1 \cup \pi_2 \rrbracket^{\mathbf{M}} = \llbracket \pi_1 \rrbracket^{\mathbf{M}} \cup \llbracket \pi_2 \rrbracket^{\mathbf{M}}$$

$$\llbracket \pi^* \rrbracket^{\mathbf{M}} = (\llbracket \pi \rrbracket^{\mathbf{M}})^*$$

Epistemic PDL with Updates Reduces to Epistemic PDL

Program transformation [4]:

$$T_{ij}^A(a) = \begin{cases} \text{pre}(s_i)?; a & \text{if } s_i \xrightarrow{a} s_j, \\ \perp? & \text{otherwise} \end{cases}$$

$$T_{ij}^A(\varphi?) = \begin{cases} \varphi? & \text{if } i = j, \\ \perp? & \text{otherwise} \end{cases}$$

$$T_{ij}^A(\pi_1; \pi_2) = \bigcup_{k=0}^{n-1} (T_{ik}^A(\pi_1); T_{kj}^A(\pi_2))$$

$$T_{ij}^A(\pi_1 \cup \pi_2) = T_{ij}^A(\pi_1) \cup T_{ij}^A(\pi_2)$$

$$T_{ij}^A(\pi^*) = K_{ijn}^A(\pi)$$

where $K_{ijk}^A(\pi)$ is a (transformed) program for all the π^* paths from s_i to s_j that can be traced through A while avoiding a pass through intermediate states s_k and higher.

In particular:

- $K_{ij0}^A(\pi)$ is a program for all the π^* paths from s_i to s_j that can be traced through A without stopovers at intermediate states, i.e., if $i = j$ it either is the skip action or a direct π loop, and otherwise it is a direct π step.
- $K_{ijn}^A(\pi)$ is a program for all the π^* paths from s_i to s_j that can be traced through A , for stopovers at any s_k ($0 \leq k \leq n - 1$) are allowed.
- Note that it is immaterial **how many times** a stopover is made at a particular intermediate state.
- Note the connection with the proof method of Kleene's theorem: the language recognized by a finite automaton is regular.

$K_{ijk}^A(\pi)$ is defined by recursing on k , as follows:

$$K_{ij0}^A(\pi) = \begin{cases} \top? \cup T_{ij}^A(\pi) & \text{if } i = j, \\ T_{ij}^A(\pi) & \text{otherwise} \end{cases}$$

$$K_{ij(k+1)}^A(\pi) = \begin{cases} (K_{kkk}^A(\pi))^* & \text{if } i = k = j, \\ (K_{kkk}^A(\pi))^*; K_{kjk}^A(\pi) & \text{if } i = k \neq j, \\ K_{ikk}^A(\pi); (K_{kkk}^A(\pi))^* & \text{if } i \neq k = j, \\ K_{ijk}^A(\pi) \cup (K_{ikk}^A(\pi); (K_{kkk}^A(\pi))^*; K_{kjk}^A(\pi)) & \text{otherwise} \\ & (i \neq k \neq j). \end{cases}$$

Kleene Path lemma

Suppose $(w, w') \in \llbracket T_{ij}^A(\pi) \rrbracket^{\mathbf{M}}$ iff there is a π path from (w, s_i) to (w', s_j) in $\mathbf{M} \circ A$.

Then $(w, w') \in \llbracket K_{ijn}^A(\pi) \rrbracket^{\mathbf{M}}$ iff there is a π^* path from (w, s_i) to (w', s_j) in $\mathbf{M} \circ A$.

Program Transformation Lemma

Assume A has n states s_0, \dots, s_{n-1} . Then:

$$\mathbf{M} \models_w [A, s_i][\pi]\varphi \text{ iff } \mathbf{M} \models_w \bigwedge_{j=0}^{n-1} [T_{ij}^A(\pi)][A, s_j]\varphi.$$

This yields the desired reduction axioms, and completeness follows from the completeness of the proof system for Epistemic PDL.

Special case: public announcement

The reduction axiom for the public announcement action P_φ with respect to the program for common knowledge among agents B , works out as follows:

$$\begin{aligned} [P_\varphi, s_0][B^*]\psi &\leftrightarrow [T_{00}^{P_\varphi}(B^*)][P_\varphi, s_0]\psi \\ &\leftrightarrow [K_{001}^{P_\varphi}(B)][P_\varphi, s_0]\psi \\ &\leftrightarrow [(\varphi?; B)^*][P_\varphi, s_0]\psi. \end{aligned}$$

This expresses that every B path consisting of φ worlds ends in a $[P_\varphi, s_0]\psi$ world.

Related Work

- [1] shows in a non-constructive way that logics of finite updates for particular action signatures can be embedded in PDL.
- [7] shows how generic updates with epistemic actions can be axiomatized in automata PDL [6, Chapter 10.3].
- [4] shows that the detour through automata is not necessary.
- If you are looking for a generic logic of communication, Epistemic Update PDL might be your choice [2].
- DEMO gives an implementation of Epistemic Update PDL: [3]. Also see the DEMO homepage, at <http://www.cwi.nl/~jve/demo>.

Module Declaration

```
module LAI16 where

import List
import Char
import LAI9
import LAI10
import LAI11
import LAI12
import LAI13
import LAI14
import LAI15
```

Language of PDL with Public Announcements

```
data Frm = Bot
  | Prp Prop
  | Ng Frm
  | Cnj [Frm]
  | Dsj [Frm]
  | Rl Rl Frm
  | Publ Frm Frm
deriving (Eq,Ord)
```

```
top = Ng Bot
```

```
imp f1 f2 = Ng (Cnj[f1,Ng f2])
```

```
data R1 = Ag Agent  
        | Test Frm  
        | Cmp [R1]  
        | Cup [R1]  
        | Star R1  
        deriving (Eq,Ord)
```

```
instance Show Frm where
  show Bot = "B" ; show (Prp p) = show p
  show (Ng Bot) = "T"
  show (Ng (Cnj [f,Ng g])) =
    '(' : show f ++ "=>" ++ show g ++ ")"
  show (Ng f) = '-' : (show f)
  show (Cnj fs) = '&' : show fs
  show (Dsj fs) = 'v' : show fs
  show (Rl r f) = '[' : show r ++ "]" ++ show f
  show (Publ f f') = '[' : show f ++ "!" ++ show f'
```

```
instance Show R1 where
  show (Ag ag) = show ag
  show (Test f) = show f ++ "?"
  show (Cmp []) = ""
  show (Cmp [r]) = show r
  show (Cmp (r:rs)) =
    show r ++ ";" ++ show (Cmp rs)
  show (Cup []) = ""
  show (Cup [r]) = show r
  show (Cup (r:rs)) =
    show r ++ " U " ++ show (Cup rs)
  show (Star r) = '(' : show r ++ ")*"
```

Truth Definition

```
isTrAt :: Ord state =>
  EpistM state -> state -> Frm -> Bool
isTrAt m w Bot = False
isTrAt m@(Mo _ _ val _ _) w (Prp p) =
  elem p (apply val w)
isTrAt m w (Ng f) = not (isTrAt m w f)
isTrAt m w (Cnj fs) = and (map (isTrAt m w) fs)
isTrAt m w (Dsj fs) = or (map (isTrAt m w) fs)
isTrAt m w (Rl r f) =
  and [ isTrAt m v f | v <- rightS (semRl m r) w ]
isTrAt m w (Publ f1 f2) = not (isTrAt m w f1)
  || isTrAt (upd_publ m f1) w f2
```

Updates with Public Announcements Again

```
upd_publ :: Ord state =>
           EpistM state -> Frm -> EpistM state
upd_publ m@(Mo states agents val rels actual) f =
  (Mo states' agents val' rels' actual')
  where
    states' = [ s | s <- states, isTrAt m s f ]
    val'    = [(s,p) | (s,p) <- val,
                    s 'elem' states' ]
    rels'   = [(a,x,y) | (a,x,y) <- rels,
                       x 'elem' states',
                       y 'elem' states' ]
    actual' = [ s | s <- actual, isTrAt m s f ]
```

Interpreting Relational Expressions

```
semR1 :: Ord state =>
        EpistM state -> R1 -> Rel state
```

The single agent relation:

```
semR1 (Mo _ _ _ rel _) (Ag b) =
    [(x,y) | (ag,x,y) <- rel, ag == b ]
```

The test is interpreted using isTrAt:

```
semR1 m@(Mo worlds _ _ _ _) (Test f) =
    [(x,x) | x <- worlds, isTrAt m x f ]
```

Cmp is interpreted as relation composition:

```
semR1 m (Cmp []) = []
semR1 m (Cmp [r]) = semR1 m r
semR1 m (Cmp (r:rs)) =
  (semR1 m r) @@ (semR1 m (Cmp rs))
```

Cup is interpreted as relation union:

```
semR1 m (Cup []) = []
semR1 m (Cup [r]) = semR1 m r
semR1 m (Cup (r:rs)) =
  (nub.sort) ((semR1 m r) ++ (semR1 m (Cup rs)))
```

Star is interpreted as reflexive transitive closure:

```
semR1 m@(Mo worlds _ _ _ _) (Star r) =  
  rtc worlds (semR1 m r)
```

Translating from PDL with Public Announcements to PDL

Formula transformation

$$\perp^\bullet := \perp$$

$$p^\bullet := p$$

$$(\neg\varphi)^\bullet := \neg\varphi^\bullet$$

$$(\varphi_1 \wedge \varphi_2)^\bullet := \varphi_1^\bullet \wedge \varphi_2^\bullet$$

$$([\pi]\varphi)^\bullet := [\pi^\circ]\varphi^\bullet$$

This uses relation expression translation $^\circ$.

Translating Formulas in the Scope of $[\varphi!]$

$$\begin{aligned}([\varphi!]\perp)^\bullet &:= \neg(\varphi)^\bullet \\([\varphi!]p)^\bullet &:= (\varphi)^\bullet \rightarrow p \\([\varphi!](\psi \wedge \chi))^\bullet &:= ([\varphi!]\psi)^\bullet \wedge ([\varphi!]\chi)^\bullet \\([\varphi!][\pi]\psi)^\bullet &:= [T_\varphi(\pi)]([\varphi!]\psi)^\bullet \\([\varphi!][\psi!]\chi)^\bullet &:= ([\varphi!]([\psi!]\chi)^\bullet)^\bullet\end{aligned}$$

This uses relation expression transformation T_φ .

Translating Relation Expressions

$$a^\circ := a$$

$$(\varphi?)^\circ := (\varphi^\bullet)?$$

$$(\pi_1; \pi_2)^\circ := \pi_1^\circ; \pi_2^\circ$$

$$(\pi_1 \cup \pi_2)^\circ := \pi_1^\circ \cup \pi_2^\circ$$

$$(\pi^*)^\circ := (\pi^\circ)^*$$

Public Announcements as Relation Transformers

$$T_\varphi(a) := \varphi^\bullet?; a$$

$$T_\varphi(\psi?) := (\varphi^\bullet \wedge ([\varphi!]\psi)^\bullet)?$$

$$T_\varphi(\pi_1; \pi_2) := T_\varphi(\pi_1); T_\varphi(\pi_2)$$

$$T_\varphi(\pi_1 \cup \pi_2) := T_\varphi(\pi_1) \cup T_\varphi(\pi_2)$$

$$T_\varphi(\pi^*) := (T_\varphi(\pi))^*$$

Implementation

```
tr :: Frm -> Frm
tr Bot = Bot
tr (Prp p) = Prp p
tr (Ng f) = Ng (tr f)
tr (Cnj fs) = Cnj (map tr fs)
tr (Dsj fs) = Dsj (map tr fs)
tr (Rl r f) = Rl (trr r) (tr f)
```

```
tr (Publ f (Bot)) = Ng (tr f)
tr (Publ f (Prp p)) = imp (tr f) (Prp p)
tr (Publ f (Ng f')) =
  imp (tr f) (Ng (tr (Publ f f')))
tr (Publ f (Cnj fs)) =
  Cnj (map tr [ Publ f f' | f' <- fs ])
tr (Publ f (Dsj fs)) =
  Dsj (map tr [ Publ f f' | f' <- fs ])
tr (Publ f (Rl r f')) =
  Rl (transform f r) (tr (Publ f f'))
tr (Publ f (Publ f2 f3)) =
  tr (Publ f (tr (Publ f2 f3)))
```

Translating relational expressions

```
trr :: R1 -> R1
trr (Ag a) = Ag a
trr (Test f) = Test (tr f)
trr (Cmp rs) = Cmp (map trr rs)
trr (Cup rs) = Cup (map trr rs)
trr (Star r) = Star (trr r)
```

Computing the transform of a relation by a public announcement

```
transform :: Frm -> Rl -> Rl
transform f (Ag a)      = Cmp [Test (tr f), Ag a]
transform f (Test f') =
    Test (Cnj [tr f, tr(Publ f f')])
transform f (Cmp rs)   =
    Cmp (map (transform f) rs)
transform f (Cup rs)   =
    Cup (map (transform f) rs)
transform f (Star r)   = Star (transform f r)
```

Example: Common Knowledge after Public Announcement

```
frm1 = Publ (Prp (P 0))  
      (R1 (Star (Cup [Ag a, Ag b]))) (Prp (Q 0)))
```

```
LAI16> frm1  
[p!][a U b]*q
```

This expresses that after public announcement of p it is common knowledge among a, b that q .

```
LAI16> tr frm1  
[(p?; a U p?; b)*] (p=>q)
```

This expresses that at the end of every path consisting of a or b steps along states where p holds, the implication $p \Rightarrow q$ holds.

Example 2: Common Knowledge after Public Announcement

```
frm2 = Publ (Prp (P 0))  
      (R1 (Star (Cup [Ag a, Ag b])) (Prp (P 0)))
```

```
LAI16> frm  
[p!][[a U b]*]p
```

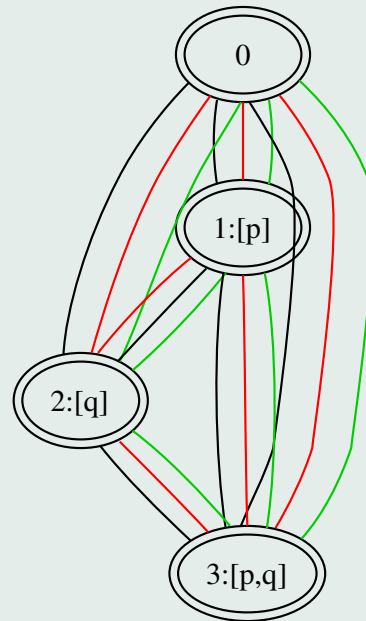
This expresses that after public announcement of p it is common knowledge among a, b that p .

```
LAI16> tr frm2  
[(p?; a U p?; b)*] (p=>p)
```

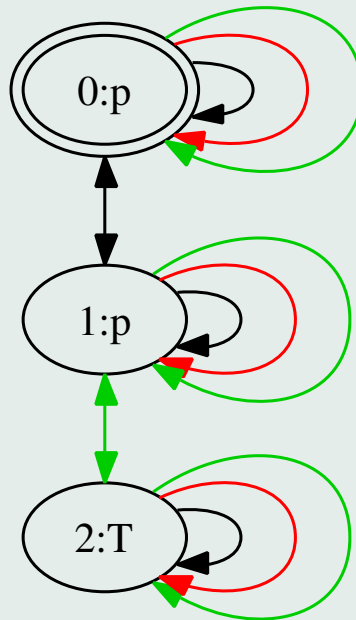
This is equivalent with $[(p?; a \cup p?; b)^*]\top$, which is again equivalent with \top . In other words, $[p!][[a \cup b]^*]p$ is a logical truth.

Everybody learns p , but a suspects that c does not

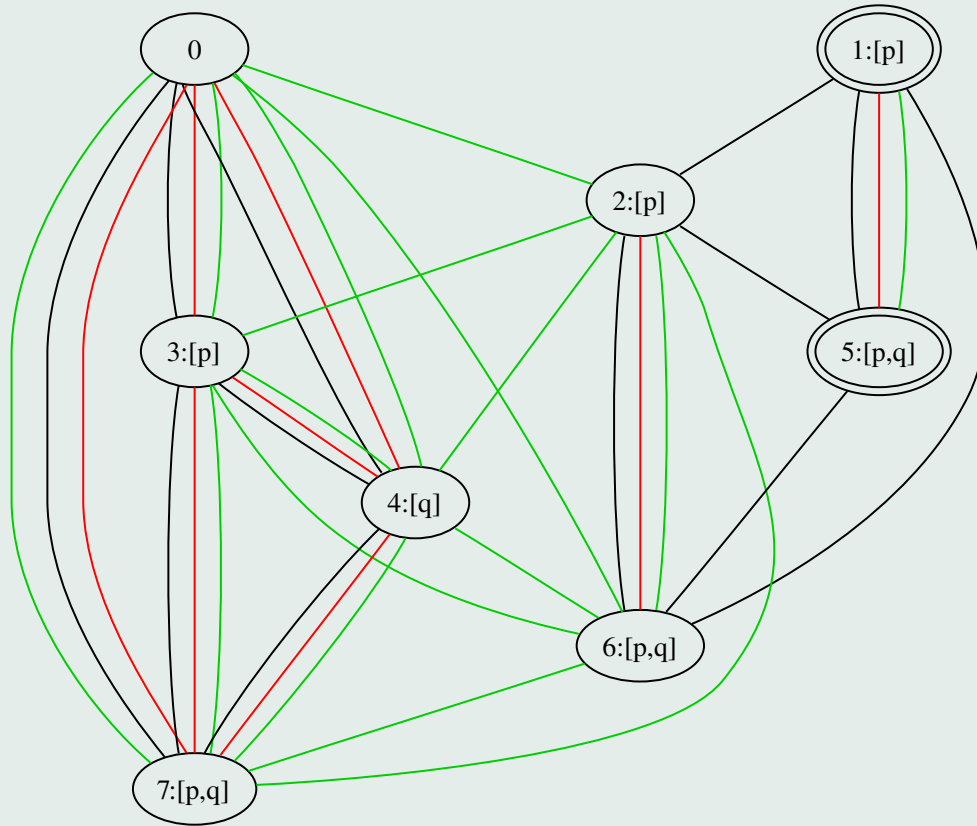
Back to modelling. A situation of blissful ignorance about p and q :



An action model for updating with the information that p gets publicly announced, but a is not sure that c gets the message:



Result of updating with this:



Implementation

```
initm = initM [a,b,c] [P 0,Q 0]

am ags =
  Am [0,1,2]
  ags
  [(0,p), (1,p), (2,Top)]
  ([[ag,x,x] | ag <- ags, x <- [0,1,2]] ++
   [(a,0,1), (a,1,0), (c,1,2), (c,2,1)])
  [0]
```

```
LAI16> displayS5 initm
```

```
[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]
```

```
LAI16> displayS5 (upd initm am)
```

```
[0,1,2,3,4,5,6,7]
```

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

```
(4, [q]), (5, [p,q]), (6, [p,q]), (7, [p,q])]
```

```
(a, [[0,3,4,7], [1,2,5,6]])
```

```
(b, [[0,3,4,7], [1,5], [2,6]])
```

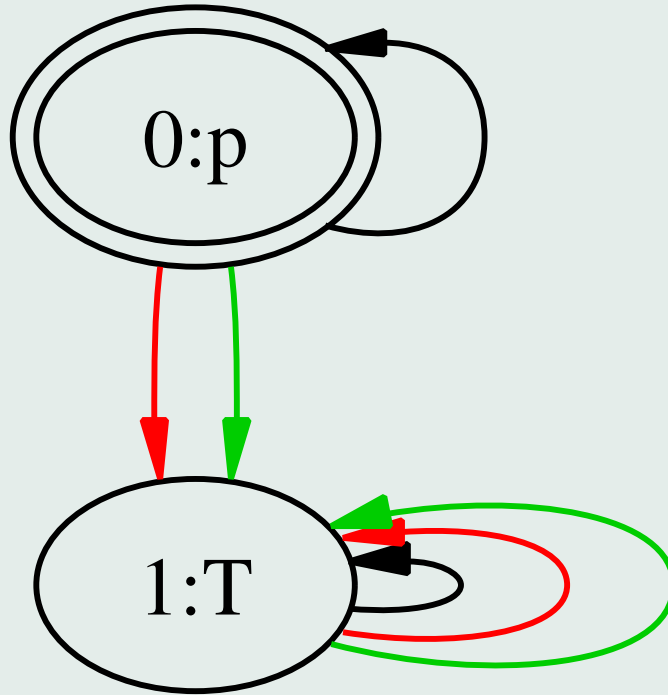
```
(c, [[0,2,3,4,6,7], [1,5]])
```

```
[1,5]
```

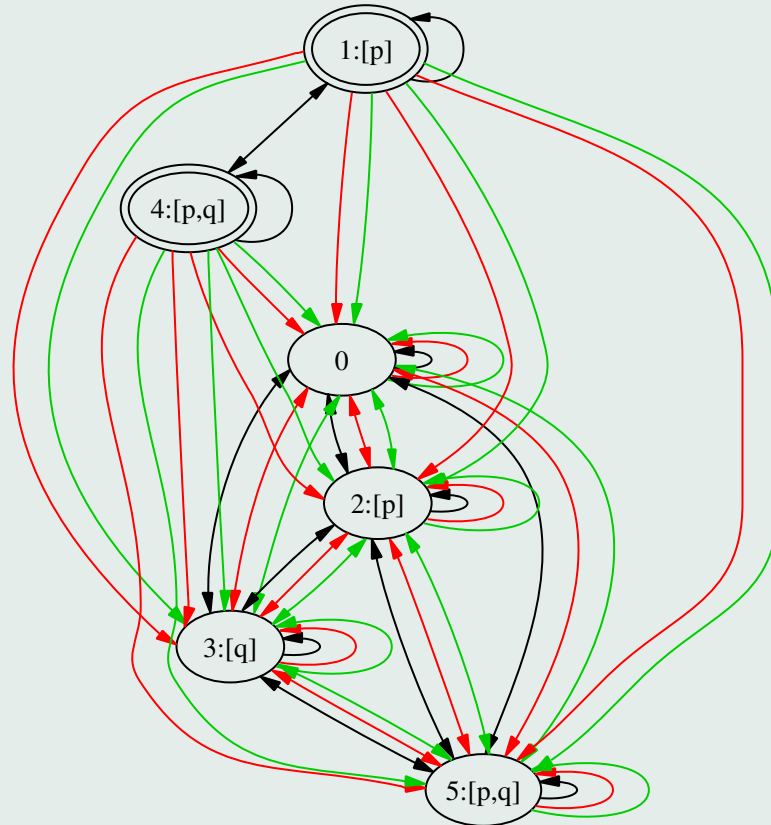
Secrets

Computing the update for passing a **secret** message to a list of agents: the other agents remain unaware of the fact that something goes on. In the limit case where the secret is divulged to all agents, the secret becomes a public update.

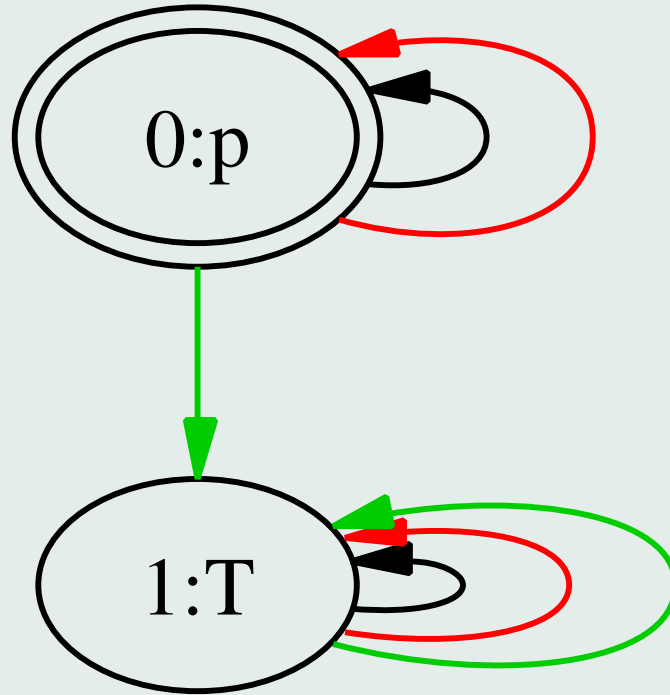
Telling a Secret to an Individual



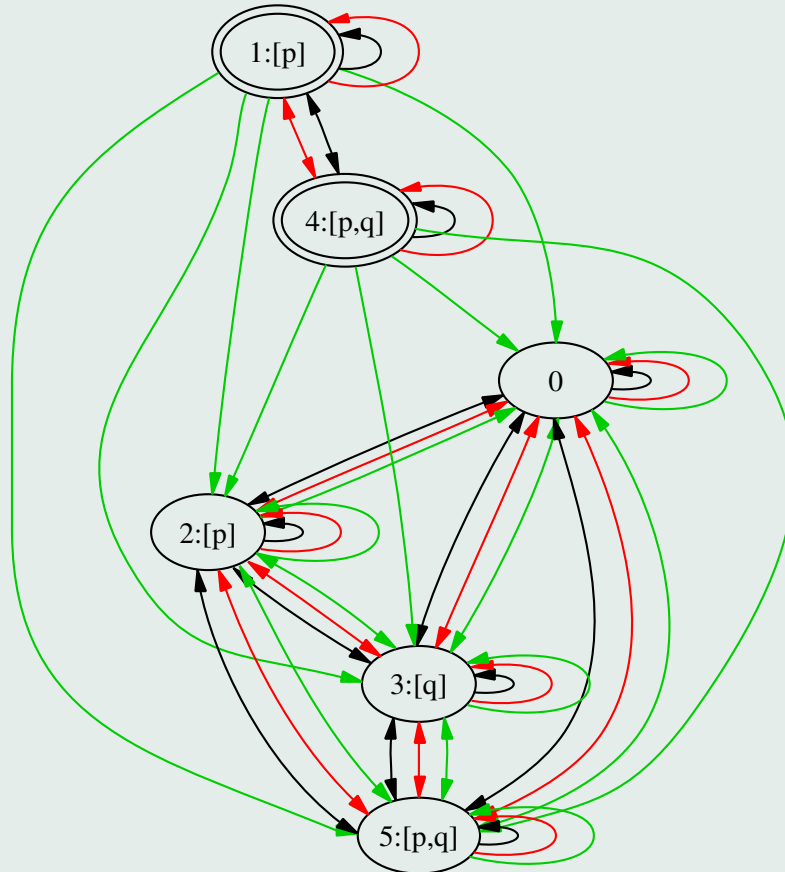
Effects of Telling a Secret



Telling a Secret to a Group



And its Effects



Implementation

```
secret :: [Agent] -> Form -> FAM State
secret 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 <- gr ]
      ++ [ (a,0,1) | a <- agents \\ gr ]
      ++ [ (a,1,1) | a <- agents      ])
     [0])
```

Lies

Difference between an untruth and a lie:

Untruth You tell something that is false, but you are not aware that it is false.

Lie You are telling something that is false, and you are aware of the fact that it is false.

Lie: update with $K_a\varphi$ in a situation where $K_a\neg\varphi$ is the case. This is bad.

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. To appear in **Information and Computation**, 2006. Available from www.cwi.nl/~jve/papers/05/lcc/.
- [3] Jan van Eijck. Dynamic epistemic modelling. Technical Report SEN-E0424, CWI, Amsterdam, December 2004. Available from <http://db.cwi.nl/rapporten/>.
- [4] Jan van Eijck. Reducing dynamic epistemic logic to PDL by program transformation. Technical Report SEN-E0423, CWI, Am-

sterdam, December 2004. Available from <http://db.cwi.nl/rapporten/>.

- [5] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.
- [6] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic. Foundations of Computing*. MIT Press, Cambridge, Massachusetts, 2000.
- [7] B. Kooi and J. van Benthem. Reduction axioms for epistemic actions. In R. Schmidt, I Pratt-Hartmann, M. Reynolds, and H. Wansing, editors, *AiML-2004: Advances in Modal Logic*, number UMCS-04-9-1 in Technical Report Series, pages 197–211. University of Manchester, 2004.
- [8] R. Parikh. The completeness of propositional dynamic logic. In

Mathematical Foundations of Computer Science 1978, pages 403–415. Springer, 1978.

- [9] V. Pratt. Semantical considerations on Floyd–Hoare logic. **Proceedings 17th IEEE Symposium on Foundations of Computer Science**, pages 109–121, 1976.
- [10] V. Pratt. Application of modal logic to programming. **Studia Logica**, 39:257–274, 1980.
- [11] K. Segerberg. A completeness theorem in the modal logic of programs. In T. Traczyk, editor, **Universal Algebra and Applications**, pages 36–46. Polish Science Publications, 1982.

The End

Suppose I try saying something.

What way do I have of knowing
that if I say I know something
I don't really not know it?

Or what way do I have of knowing
that if I say I don't know something
I don't really in fact know it?

Chuang Tzu