Guarded Actions

Jan van Eijck

CWI and ILLC, Amsterdam, Uil-OTS, Utrecht

December 7, 2004

Abstract

Guarded actions are changes with preconditions acting as a guard. Guarded action models are multimodal Kripke models with the valuations replaced by guarded actions. Call guarded action logic the result of adding product updates with guarded action models to PDL (propositional dynamic logic). We show that guarded action logic reduces to PDL.

keywords Dynamic epistemic logic, logic of communication, logic of change, propositional dynamic logic, program transformation.

ACM Classification (1998) E 4, F 4.1, H 1.1.

1 Introduction

It was shown in [6] how generic updating with finite epistemic actions can be axiomatized in automata PDL [5, Chapter 10.3], and in [2] how generic updating with finite epistemic actions can be axiomatized in PDL. Below we give a reduction of generic updating with finite guarded action models to PDL.

2 PDL and Guarded Action Models

Let p range over a set of basic propositions P and let a range over a set of agents Ag. Then the language $\mathcal{L}_{\mathcal{P},Ag}$ of PDL over P, Ag is given by:

$$\begin{array}{rcl} \varphi & ::= & \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\pi]\varphi \\ \pi & ::= & a \mid ?\varphi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \end{array}$$

Employ the usual abbreviations: $\bot$ is shorthand for $\neg\top$, $\varphi_1 \vee \varphi_2$ is shorthand for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \to \varphi_2$ is shorthand for $\neg(\varphi_1 \wedge \varphi_2)$, $\varphi_1 \leftrightarrow \varphi_2$ is shorthand for $(\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1)$, and $\langle\pi\rangle\varphi$ is shorthand for $\neg[\pi]\neg\varphi$. Also, if $B \subseteq Ag$ and B is finite, use B as shorthand for $b_1 \cup b_2 \cup \cdots$. Under this convention, the general knowledge operator $E_B\varphi$ takes the shape $[B]\varphi$, while the common knowledge operator $C_B\varphi$ appears as $[B^*]\varphi$, i.e., $[B]\varphi$ expresses that it is general knowledge among agents B that φ, and $[B^*]\varphi$ expresses that it is common knowledge among agents B that φ. In the special case where $B = \emptyset$, B turns out equivalent to $?\bot$, the program that always fails.

The semantics of $\mathcal{L}_{\mathcal{P},Ag}$ is given relative to labelled transition systems $\mathbf{M} = (W, V, R)$, where

- W is a set of worlds (or states),
- $V : W \to \mathcal{P}(P)$ is a valuation function,
- $R = \{\xrightarrow{a} \subseteq W \times W \mid a \in Ag\}$ is a set of labelled transitions, i.e., binary relations on W, one for each label a.

Substitutions $\mathcal{L}_{\mathcal{P},Ag}$ substitutions are functions of type $\mathcal{L}_{\mathcal{P},Ag} \to \mathcal{L}_{\mathcal{P},Ag}$ that distribute over all language constructs, and that map all but a finite number of basic propositions to themselves.

$\mathcal{L}_{\mathcal{P},Ag}$ substitutions can be represented as sets of bindings

$$\{p_1 \mapsto \varphi_1, \ldots, p_n \mapsto \varphi_n\}$$

where all the p_i are different, and where no φ_i is equal to p_i. If σ is a $\mathcal{L}_{\mathcal{P},Ag}$ substitution, then the set $\{p \in P \mid \sigma(p) \neq p\}$ is called its *domain*, notation $\mathrm{dom}(\sigma)$. Use ϵ for the identity substitution. Let $\Sigma_{P,Ag}$ be the set of all $\mathcal{L}_{\mathcal{P},Ag}$ substitutions.

If $\sigma = \{p_1 \mapsto \varphi_1, \ldots, p_n \mapsto \varphi_n\}$ is a $\mathcal{L}_{\mathcal{P},Ag}$ substitution, we use φ^σ for $\sigma(\varphi)$ and π^σ for $\sigma(\pi)$. We can spell out φ^σ and π^σ, as follows:

$$\begin{array}{rcl}
\top^\sigma & = & \top \\
p^\sigma & = & \begin{cases} \sigma(p) & \text{if } p \in \mathrm{dom}(\sigma), \\ p & \text{otherwise} \end{cases} \\
(\neg\varphi)^\sigma & = & \neg\varphi^\sigma \\
(\varphi_1 \wedge \varphi_2)^\sigma & = & \varphi_1^\sigma \wedge \varphi_2^\sigma \\
([\pi]\varphi)^\sigma & = & [\pi^\sigma]\varphi^\sigma \\
\\
a^\sigma & = & a \\
B^\sigma & = & B \\
(?\varphi)^\sigma & = & ?\varphi^\sigma \\
(\pi_1;\pi_2)^\sigma & = & \pi_1^\sigma;\pi_2^\sigma \\
(\pi_1 \cup \pi_2)^\sigma & = & \pi_1^\sigma \cup \pi_2^\sigma \\
(\pi^*)^\sigma & = & (\pi^\sigma)^*.
\end{array}$$

If $M = (W, V, R)$ is a $\mathcal{L}_{\mathcal{P},Ag}$ model and σ is a $\mathcal{L}_{\mathcal{P},Ag}$ substitution, then V_M^σ is the valuation given by $\lambda w \lambda p \cdot w \in [\![p^\sigma]\!]^M$. In other words, V_M^σ assigns to w the set of basic propositions p such that p^σ is true in world w in model M. For $M = (W, V, R)$, call M^σ the model given by (W, V_M^σ, R).

We can prove the following:

Lemma 1 (Substitution) *For all $\mathcal{L}_{\mathcal{P},\mathrm{Ag}}$ models M, all $\mathcal{L}_{\mathcal{P},\mathrm{Ag}}$ formulas σ, all $\mathcal{L}_{\mathcal{P},\mathrm{Ag}}$ programs π, all $\mathcal{L}_{\mathcal{P},\mathrm{Ag}}$ substitutions σ:*

$$M \models_w \varphi^\sigma \text{ iff } M^\sigma \models_w \varphi.$$

$$(w, w') \in [\![\pi^\sigma]\!]^M \text{ iff } (w, w') \in [\![\pi]\!]^{M^\sigma}.$$

Proof. Simultaneous induction on the structure of φ and π. □

[1] proposes to model epistemic actions as epistemic models, with valuations replaced by preconditions. Here, we extend this to guarded action models.

Guarded action models for $\mathcal{L}_{\mathcal{P},Ag}$ A guarded action model for $\mathcal{L}_{\mathcal{P},Ag}$ is a quadruple $A = ([s_0, \dots, s_{n-1}], \mathrm{pre}, \mathrm{sub}, T)$ where

- $[s_0, \dots, s_{n-1}]$ is a finite list of action states,
- $\mathrm{pre} : \{s_0, \dots, s_{n-1}\} \to \mathcal{L}_{\mathcal{P},Ag}$ assigns a precondition to each action state,
- $\mathrm{sub} : \{s_0, \dots, s_{n-1}\} \to \Sigma_{P,Ag}$ assigs a $\mathcal{L}_{\mathcal{P},Ag}$ substitution to each action state,
- $T : Ag \to \mathcal{P}(\{s_0, \dots, s_{n-1}\}^2)$ assigns an accessibility relation $\xrightarrow{a}$ to each agent $a \in Ag$.

A pair $\mathbf{A} = (A, s)$ with A a guarded action model and $s \in \{s_0, \dots, s_{n-1}\}$ is a pointed guarded action model. In a pointed guarded action model (A, s), s points to the action that actually takes place.

Note that an action model in the sense of [1] can be viewed as the special case of a guarded action model where sub assigns the empty substitution ϵ to every state.

Guarded actions can be executed in $\mathcal{L}_{\mathcal{P},Ag}$ models by means of the following product construction:

Guarded Action Update Let a $\mathcal{L}_{\mathcal{P},Ag}$ model $\mathbf{M} = (W, V, R)$, a world $w \in W$, and a pointed guarded action model (A, s), with $A = ([s_0, \dots, s_{n-1}], \mathrm{pre}, \mathrm{sub}, T)$, be given. Then the result of executing (A, s) in $(\mathbf{M}, w)$ is the model $(\mathbf{M} \otimes A, (w, s))$, with $\mathbf{M} \otimes A = (W', V', R')$, where

$$\begin{array}{rcl} W' & = & \{(w, s) \mid s \in \{s_0, \dots, s_{n-1}\}, w \in [\![\mathrm{pre}(s)]\!]^{\mathbf{M}}\} \\ V'(w, s) & = & V_M^{\mathrm{sub}(s)}(w) \\ R'(a) & = & \{((w, s), (w', s')) \mid (w, w') \in R(a), (s, s') \in T(a)\}. \end{array}$$

The language of $\mathcal{L}_{\mathcal{P},Ag}(\mathrm{GA})$ ($\mathcal{L}_{\mathcal{P},Ag}$ with guarded actions) is given by extending the $\mathcal{L}_{\mathcal{P},Ag}$ language with update constructions $[A, s]\varphi$, where (A, s) is a pointed guarded action model. The interpretation of $[A, s]\varphi$ in $\mathbf{M}$ is given by:

$$[\![[A, s]\varphi]\!]^{\mathbf{M}} = \{w \in W_{\mathbf{M}} \mid \text{if } \mathbf{M} \models_w \mathrm{pre}(s) \text{ then } (w, s) \in [\![\varphi]\!]^{\mathbf{M} \otimes A}\}.$$

Updating with multiple pointed guarded update actions is also possible. A multiple pointed guarded action is a pair (A, S), with A a guarded action model, and S a subset of the state set of A. Extend the language with updates $[A, S]\varphi$, and interpret this as follows:

$$[\![[A, S]\varphi]\!]^{\mathbf{M}} \quad = \quad \{w \in W_{\mathbf{M}} \mid \forall s \in S(\text{ if } \mathbf{M} \models_w \text{pre}(s) \text{ then } \mathbf{M} \otimes A \models_{(w,s)} \varphi)\}.$$

We have to check that the definition of updating with guarded action models is well behaved. The following theorems state that it is, in the sense that it preserves epistemic model bisimulation and guarded action model bisimulation.

Theorem 2 *For all pointed $\mathcal{L}_{\mathcal{P},\mathrm{Ag}}$ models $(M, w), (N, v)$, all pointed action models (A, s):*

$$\textit{If } M, w \leftrightarrow N, v \textit{ then } M \otimes A, (w, s) \leftrightarrow N \otimes A, (v, s).$$

Proof. Let R be a bisimulation witnessing $M, w \leftrightarrow N, v$. Then the relation C between $W_M \times W_A$ and $W_N \times W_A$ given by $(w, s)C(v, t)$ iff wRv and $s = t$ is a bisimulation.

Suppose $(w, s)C(v, t)$. Then wRv and $s = t$. The only non-trivial check is the check for sameness of valuation. By wRv, w and v satisfy $V_M(w) = V_N(s)$. By $s = t$, s and t have the same substitution σ. By the fact that w and v are bisimilar, we have $w \in [\![\varphi]\!]^M$ iff $v \in [\![\varphi]\!]^N$. Thus, by $V_M(w) = V_N(s)$ and the definition of V_M^σ and V_N^σ, we get $V_M^\sigma(w) = V_N^\sigma(v)$. □

A guarded action bisimulation is like an ordinary bisimulation, except for the fact that the requirement of 'same valuations' is replaced by a requirement of 'equivalent preconditions and equivalent substitutions'.

Theorem 3 *For all pointed $\mathcal{L}_{\mathcal{P},\mathrm{Ag}}$ models (M, w), all pointed action models (A, s) and (B, t):*

$$\textit{If } A, s \leftrightarrow B, t \textit{ then } M \otimes A, (w, s) \leftrightarrow M \otimes B, (w, t).$$

Proof. Let R be a bisimulation witnessing $A, s \leftrightarrow B, t$. Then the relation C between $W_M \times W_A$ and $W_M \times W_B$ given by $(w, s)C(v, t)$ iff $w = v$ and sRt is a bisimulation.

Suppose $(w, s)C(v, t)$. Then $w = v$ and sRt. Again, the only non-trivial check is the check for sameness of valuation. By sRt, the substitutions σ of s and τ of t are equivalent. By $w = v$, $V_M(w) = V_M(v)$. It follows that $V_M^\sigma(w) = V_M^\sigma(v)$, i.e., (w, s) and (v, t) have the same valuation. □

3 Program Transformation

We will now show how $\mathcal{L}_{\mathcal{P},Ag}$(GA) formulas can be reduced to $\mathcal{L}_{\mathcal{P},Ag}$ formulas. The procedure is identical to that for reducing update PDL to PDL from [2].

For every action model A with states $s_0, \dots, s_{n-1}$ we define a set of n^2 program transformers

$T^A_{i,j}$ $(0 \leq i < n, 0 \leq j < n)$, as follows:

$$\begin{array}{rcl}
T^A_{ij}(a) & = & \begin{cases} ?\mathrm{pre}(s_i); a & \text{if } s_i \xrightarrow{a} s_j, \\ ?\bot & \text{otherwise} \end{cases} \\
T^A_{ij}(?\varphi) & = & \begin{cases} ?(\mathrm{pre}(s_i) \wedge [A, s_i]\varphi) & \text{if } i = j, \\ ?\bot & \text{otherwise} \end{cases} \\
T^A_{ij}(\pi_1; \pi_2) & = & \bigcup_{k=0}^{n-1} (T^A_{ik}(\pi_1); T^A_{kj}(\pi_2)) \\
T^A_{ij}(\pi_1 \cup \pi_2) & = & T^A_{ij}(\pi_1) \cup T^A_{ij}(\pi_2) \\
T^A_{ij}(\pi^*) & = & K^A_{ijn}(\pi)
\end{array}$$

where $K^A_{ijk}(\pi)$ 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.

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

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

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

Lemma 4 (Kleene Path) *Suppose* $(w, w') \in [\![T^A_{ij}(\pi)]\!]^{\mathbf{M}}$ *iff there is a* π *path from* (w, s_i) *to* (w', s_j) *in* $\mathbf{M} \otimes A$. *Then* $(w, w') \in [\![K^A_{ijn}(\pi)]\!]^{\mathbf{M}}$ *iff there is a* π^* *path from* (w, s_i) *to* (w', s_j) *in* $\mathbf{M} \otimes A$.

Proof. As in [2]. □

The Kleene path lemma allows us to prove the program transformation lemma:

Lemma 5 (Program Transformation) *Assume* A *has* n *states* $s_0, \ldots, s_{n-1}$. *Then:*

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

Proof. As in [2]. □

4 Reduction Axioms for PDL with Guarded Actions

The program transformations can be used to translate $\mathcal{L}_{\mathcal{P},Ag}(\text{GA})$ to $\mathcal{L}_{\mathcal{P},Ag}$, as follows:

$$\begin{array}{rcl}
t(\top) & = & \top \\
t(p) & = & p \\
t(\neg\varphi) & = & \neg t(\varphi) \\
t(\varphi_1 \wedge \varphi_2) & = & t(\varphi_1) \wedge t(\varphi_2) \\
t([\pi]\varphi) & = & [r(\pi)]t(\varphi) \\
t([A,s]\top) & = & \top \\
t([A,s]p) & = & t(\text{pre}(s)) \rightarrow p^{\text{sub}_A(s)} \\
t([A,s]\neg\varphi) & = & t(\text{pre}(s)) \rightarrow \neg t([A,s]\varphi) \\
t([A,s](\varphi_1 \wedge \varphi_2) & = & t([A,s]\varphi_1) \wedge t([A,s]\varphi_2) \\
t([A,s_i][\pi]\varphi & = & \bigwedge_{j=0}^{n-1} [T^A_{ij}(r(\pi))]t([A,s_j]\varphi) \\
t([A,s][A',s']\varphi & = & t([A,s]t([A',s']\varphi))
\end{array}$$

$$\begin{array}{rcl}
r(a) & = & a \\
r(?\varphi) & = & ?t(\varphi) \\
r(\pi_1;\pi_2) & = & r(\pi_1); r(\pi_2) \\
r(\pi_1 \cup \pi_2) & = & r(\pi_1) \cup r(\pi_2) \\
r(\pi^*) & = & (r(\pi))^*.
\end{array}$$

Note that the *only* difference between this translation and the translation from [2] is in the clause for basic propositions, where the substitution comes into play.

The correctness of this translation follows from direct semantic inspection, using the program transformation lemma for the translation of $[A,s_i][\pi]\varphi$ formulas. The translation points the way to appropriate reduction axioms, as follows.

Take all axioms and rules of PDL [8, 4, 7], plus the following reduction axioms:

$$\begin{array}{rcl}
[A,s]\top & \leftrightarrow & \top \\
[A,s]p & \leftrightarrow & (\text{pre}(s) \rightarrow p^{\text{sub}_A(s)}) \\
[A,s]\neg\varphi & \leftrightarrow & (\text{pre}(s) \rightarrow \neg[A,s]\varphi) \\
[A,s](\varphi_1 \wedge \varphi_2) & \leftrightarrow & ([A,s]\varphi_1 \wedge [A,s]\varphi_2) \\
[A,s_i][\pi]\varphi & \leftrightarrow & \bigwedge_{j=0}^{n-1} [T^A_{ij}(\pi)][A,s_j]\varphi.
\end{array}$$

and necessitation for action model modalities. The reduction axiom for $[A,s]p$ is new: it takes the effect of the substitution into account. The reduction axioms for $[A,s]\neg\varphi$ and $[A,s](\varphi_1 \wedge \varphi_2)$ are as in [6]. The final reduction axiom, based on program transformation, is as in [2].

If updates with multiple pointed action models are also in the language, we need the following additional reduction axiom:

$$[A, S]\varphi \quad \leftrightarrow \quad \bigwedge_{s \in S}[A, s]\varphi$$

Theorem 6 (Completeness of $\mathcal{L}_{\mathcal{P},\boldsymbol{Ag}}$(GA)) *If* $\models \varphi$ *then* $\vdash \varphi$.

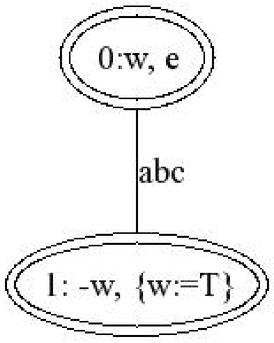
Proof. The proof system for $\mathcal{L}_{\mathcal{P},Ag}$ is complete, and every formula in the language of $\mathcal{L}_{\mathcal{P},Ag}$(GA) is provably equivalent to a $\mathcal{L}_{\mathcal{P},Ag}$ formula. □

5 Examples

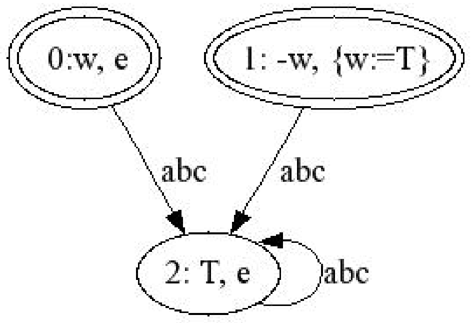
Opening a window Precondition for opening a window is that the window is closed. To make this into an action that can be performed no matter what, modify it as follows:

- Check whether the window is closed or not.
- If it is open, then do nothing.
- If it is closed, then open it.

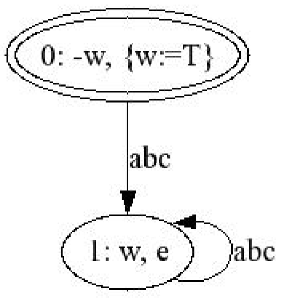
Assuming the action is visible for all (Alice, Bob and Carol), it is modelled as follows:



If the window is opened in secret, its action model looks as follows:



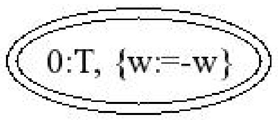
Many subtle variations on this action are possible. E.g., the window is in fact opened, while everyone gets told that it was already open. Here is the corresponding action:



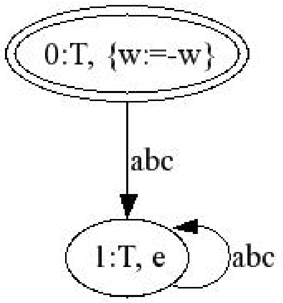
Fiddling with a window Fiddling with a window is the following composite action:

- Check whether the window is closed or not.
- If it is open, then close it.
- If it is closed, then open it.

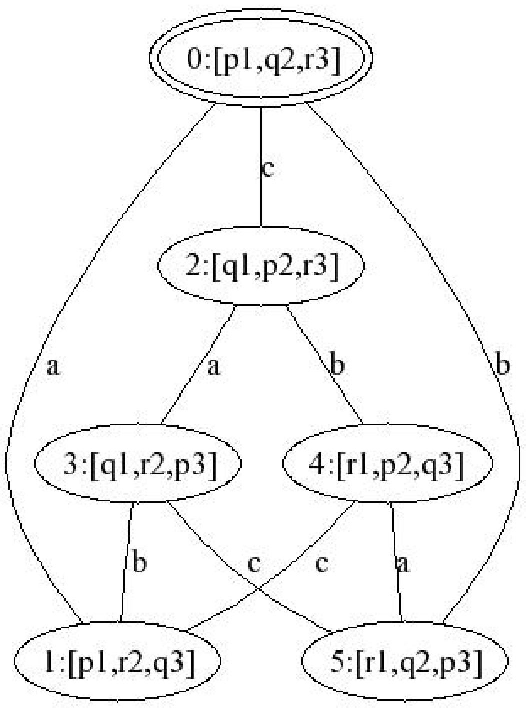
If the fiddling is done in a way that is visible to all, then here is its action model:



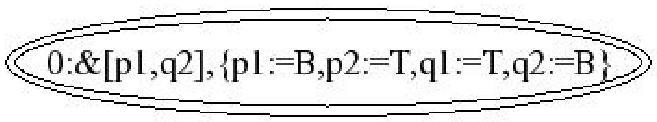
If the fiddling is done in secret, its action model looks as follows:



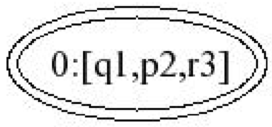
Card exchanges in games Alice, Bob and Carol each hold one of cards Purple, Qaki (Khaki), Red. The actual deal is: Alice holds Purple, Bob holds Qaki, Carol holds Red:



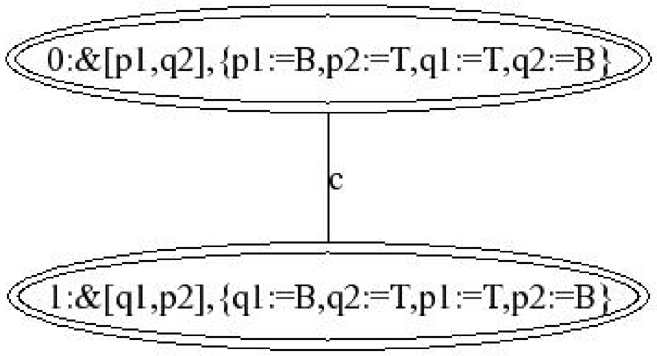
The exchange action is: Alice exchanges her Purple against Bob's Qaki, showing their cards to all.



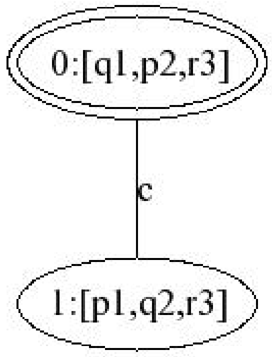
The result of this update:



Now suppose the update is the exchange of the cards that Alice and Bob are holding, but without showing the cards to Carol.


0:&[p1,q2],{p1:=B,p2:=T,q1:=T,q2:=B}
c
1:&[q1,p2],{q1:=B,q2:=T,p1:=T,p2:=B}

The result of this update in the initial situation:



6 Further Work

In [3], *action emulation* was identified as the appropriate structural equivalence on action models for capturing the relation of having the same update effect. What is the appropriate structural equivalence on guarded action models for capturing the relation of having the same update effect?

References

[1] Baltag, A., Moss, L., and Solecki, S. The logic of public announcements, common knowledge, and private suspicions. Tech. rep., Dept of Cognitive Science, Indiana University and Dept of Computing, Oxford University, 2003.

[2] Eijck, J. v. Reducing dynamic epistemic logic to PDL by program transformation. CWI, Amsterdam, `www.cwi.nl:~/papers/04/delpdl/`, 2004.

[3] Eijck, J. v., and Ruan, J. Action emulation. CWI, Amsterdam, `www.cwi.nl:~/papers/04/ae`, 2004.

[4] Fischer, M. J., and Ladner, R. E. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences 18*, 2 (1979), 194–211.

[5] Harel, D., Kozen, D., and Tiuryn, J. *Dynamic Logic. Foundations of Computing.* MIT Press, Cambridge, Massachusetts, 2000.

[6] Kooi, B., and van Benthem, J. Reduction axioms for epistemic actions. In *AiML-2004: Advances in Modal Logic* (2004), R. Schmidt, I. Pratt-Hartmann, M. Reynolds, and H. Wansing, Eds., no. UMCS-04-9-1 in Technical Report Series, University of Manchester, pp. 197–211.

[7] Parikh, R. The completeness of propositional dynamic logic. In *Mathematical Foundations of Computer Science 1978*. Springer, 1978, pp. 403–415.

[8] Segerberg, K. A completeness theorem in the modal logic of programs. In *Universal Algebra and Applications*, T. Traczyck, Ed. Polish Science Publications, 1982, pp. 36–46.