

Modelling the Epistemics of Communication with Functional Programming

Jan van Eijck^{1,2} and Simona Orzan³

¹ CWI, Amsterdam (jve@cwi.nl)

² Uil-OTS, Utrecht

³ TUE, Eindhoven (s.m.orzan@tue.nl)

Abstract. Dynamic epistemic logic is the logic of the effects of epistemic actions like making public announcements, passing private messages, revealing secrets, telling lies. This paper takes its starting point from the version of dynamic epistemic logic of [2], and demonstrates a tool that can be used for showing what goes on during a series of epistemic updates: the dynamic epistemic modelling tool DEMO [7, 9]. DEMO allows modelling epistemic updates, graphical display of update results, graphical display of action models, formula evaluation in epistemic models, and translation of dynamic epistemic formulas to PDL [22] formulas. DEMO is written in Haskell. This paper intends to demonstrate its usefulness for visualizing the model transformations that take place during epistemic updating.

Project paper, on the application of functional programming in a new area.

1 Introduction

Analysis of multi-agent communication, in the spirit of [12], consists of representing the knowledge or beliefs of the agents in a semantic model, representing the operations on the knowledge or beliefs of the agents as operations on semantic models, and do model checking to see if given formulas are true in the models that result from given updates. After the advances in dynamic epistemic logic documented in [20, 11, 1, 2], taking a model checking approach to epistemic dynamics is more attractive than ever. In this paper we introduce DEMO, a model checking tool written in Haskell and based on the streamlined version of dynamic epistemic logic taken from [2].

DEMO represents epistemic models as objects of type `EpistM` and update actions (pointed action models) on epistemic models as objects of type `PoAM`. Update operations are specified as

```
upd :: EpistM -> PoAM -> EpistM
upds :: EpistM -> [PoAM] -> EpistM
```

The updates generate new epistemic models. Formula checking is defined as

```
isTrue :: EpistM -> Form -> Bool
```

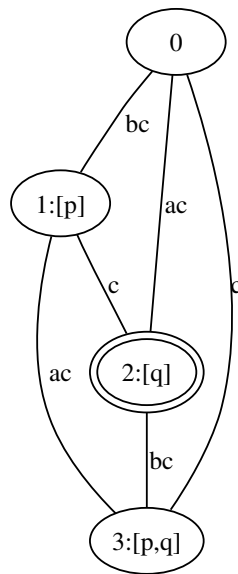
The formula evaluator `isTrue` takes an epistemic model and an epistemic formula as arguments, and returns the truthvalue of the formula in the model.

Here is the lay-out of the pages that follow. Section 2 provides the background on epistemic logic that is needed for understanding what goes on in the rest of the paper. Sections 3 and 4 discuss examples of epistemic modeling with DEMO, Section 5 explains how DEMO can be used for finding reduction axioms in epistemic logic, and the final Section 6 concludes and lists further work.

2 Epistemic Models, Action Models and Updating

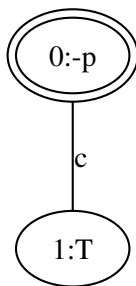
A model for representing the state of knowledge among a group of agents is a labeled transition system (LTS) with labels for the individual agents, and valuations for the states. It is common to call the states *worlds* and to refer to the LTSs as epistemic models or Kripke models [13, 3, 10].

In a situation where one of us (*a*) knows about *p* and the other (*b*) knows about *q*, while you (*c*) know nothing about either *p* or *q*, and while in fact *p* happens to be false and *q* true, the state of knowledge of the agents *a, b, c* is represented by a Kripke model where the worlds are the four different possibilities for the truth of *p* and *q* (\emptyset, p, q, pq), the epistemic accessibility relation \sim_a is the relation that links the two worlds where *p* is true and the two worlds where *p* is false, the epistemic accessibility relation \sim_b is the relation that links the two worlds where *q* is true and the two worlds where *q* is false, and the epistemic accessibility relation \sim_c is the total relation on the set of worlds. This situation can be visualised as follows (this and the following pictures were generated by DEMO with the graphic visualisation tool *dot* [16]):



Epistemic formula $K_a\phi$ evaluates to *true* in a world in such a model if in that world every *a*-accessible world makes ϕ true. In the actual world of the example picture, indicated by the double oval, $K_a\neg p$ is true, for $\neg p$ is true in worlds 0 and 2, and these are the two worlds that are *a*-accessible from the actual world 2 (we assume that every world is self-accessible). On the other hand, K_aq is false in world 2, for *q* is true in world 2 but false in world 0. K_bq is true in the actual world. More subtly, $K_a(K_bq \vee K_b\neg q)$ is true the actual world, for it happens to be the case that K_bq is true in world 2 and $K_b\neg q$ is true on world 0. The examples with embedded knowledge operators illustrate how the Kripke models encode information about what agents know about the knowledge or ignorance of other agents. In the example, *a* does not know about *q*, but *a* knows that *b* knows whether *q*. Also, all agents know that *c* is ignorant about *p* and *q*.

Epistemic updates are themselves also a kind of Kripke models, with the important difference that the worlds do not carry a valuation but a precondition formula [1]. Here is an example of a model of a group message to *a, b* that $\neg p$ is the case:



What this expresses is that in fact $\neg p$ is communicated (indicated by the double oval), but that agent c cannot distinguish this communication from a trivial communication \top .

Technically, 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 with the accessibility relations holding between pairs (w, u) and (w', u') just in case they hold both between w and w' and between u and u' . Further details are in [1, 2].

This product construction causes an exponential blow-up, and in order to model the update process in a feasible way we need to minimize the update results modulo bisimulation [14]. For this, DEMO uses the following algorithm for partition refinement, in the spirit of [18]:

- Start out with a partition of the state set where all states with the same precondition function are in the same class. The equality relation to be used to evaluate the precondition function is given as a parameter to the algorithm.
- Given a partition Π , for each block b in Π , partition b into sub-blocks such that two states s, t of b are in the same sub-block iff for all agents a it holds that s and t have \xrightarrow{a} transitions to states in the same block of Π . Update Π to Π' by replacing each b in Π by the newly found set of sub-blocks for b .
- Halt as soon as $\Pi = \Pi'$.

DEMO implements epistemic formula evaluation in update results, for a wide class of epistemic logics (multimodal epistemic logic, epistemic PDL, epistemic PDL with action modalities).

3 The Riddle of the Caps

Picture a situation of four people a, b, c, d standing in line, with a, b, c looking to the left, and d looking to the right. a can see no-one else; b can see a ; c can see a and b , and d can see no-one else. They are all wearing caps, and they cannot see their own cap. If it is common knowledge that there are two white and two black caps, then in the following situation c knows what colour cap she is wearing.



If c now announces that she knows the colour of her cap (without revealing the colour), b can infer from this that he is wearing a white cap, for b can reason as follows: “ c knows her colour, so she must see two caps of the same colour. The cap I can see is white, so my own cap must be white as well.” In this situation b draws a conclusion from the fact that c knows her colour.

In the following situation b can draw a conclusion from the fact that c does not know her colour.



In this case c announces that she does not know her colour, and b can infer from this that he is wearing a black cap, for b can reason as follows: “ c does not know her colour, so she must see two caps of different colours in front of her. The cap I can see is white, so my own cap must be black.”

To account for this kind of reasoning, we use model checking for epistemic updating, as follows (the Haskell code for this example is given in Figure 1). Proposition p_i expresses the fact that the i -th cap, counting from the left, is white. Thus, the facts of our first example situation are given by $p_1 \wedge p_2 \wedge \neg p_3 \wedge \neg p_4$, and those of our second example by $p_1 \wedge \neg p_2 \wedge \neg p_3 \wedge p_4$.

```

module Caps
where
import List
import DEMO

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
                ]

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

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

mo0 = upd (initE [P 1, P 2, P 3, P 4]) (test capsInfo)
mo1 = upd mo0 (public capsInfo)
mo2 = upds mo1 [awarenessFirstCap, awarenessSecondCap]
mo3a = upd mo2 (public cK)
mo3b = upd mo2 (public (Neg cK))

```

Fig. 1. Haskell code for the caps example.

An initial situation with four agents a, b, c, d and four propositions p_1, p_2, p_3, p_4 , with exactly two of these true, where no-one knows anything about the truth of the propositions, and everyone is aware of the ignorance of the others, is modelled like this:

```

Caps> showM mo0
==> [5,6,7,8,9,10]
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
(0, []) (1, [p1]) (2, [p2]) (3, [p3]) (4, [p4])
(5, [p1,p2]) (6, [p1,p3]) (7, [p1,p4]) (8, [p2,p3]) (9, [p2,p4])
(10, [p3,p4]) (11, [p1,p2,p3]) (12, [p1,p2,p4]) (13, [p1,p3,p4]) (14, [p2,p3,p4])
(15, [p1,p2,p3,p4])
(a, [[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(b, [[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(c, [[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])
(d, [[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]])

```

The first line indicates that worlds 5,6,7,8,9,10 are compatible with the facts of the matter (the facts being that there are two white and two black caps). E.g., 5 is the world where a and b are wearing

the white caps. The second line lists all the possible worlds; there are 2^4 of them, since every world has a different valuation. The third through sixth lines give the valuations of worlds. The last four lines represent the accessibility relations for the agents. All accessibilities are total relations, and they are represented here as the corresponding partitions on the set of worlds. Thus, the ignorance of the agents is reflected in the fact that for all of them all worlds are equivalent: none of the agents can tell any of them apart.

The information that two of the caps are white and two are black is expressed by the formula

$$(p_1 \wedge p_2 \wedge \neg p_3 \wedge \neg p_4) \vee (p_1 \wedge p_3 \wedge \neg p_2 \wedge \neg p_4) \vee (p_1 \wedge p_4 \wedge \neg p_2 \wedge \neg p_3) \\ \vee (p_2 \wedge p_3 \wedge \neg p_1 \wedge \neg p_4) \vee (p_2 \wedge p_4 \wedge \neg p_1 \wedge \neg p_3) \vee (p_3 \wedge p_4 \wedge \neg p_1 \wedge \neg p_2).$$

A public announcement with this information has the following effect:

```
Caps> showM (upd mo0 (public capsInfo))
==> [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])
(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]])
```

Let this model be called `mo1`. The representation above gives the partitions for all the agents, showing that nobody knows anything. A perhaps more familiar representation for this multi-agent Kripke model is given in Figure 2. In this picture, all worlds are connected for all agents, all worlds are compatible with the facts of the matter (indicated by the double ovals).

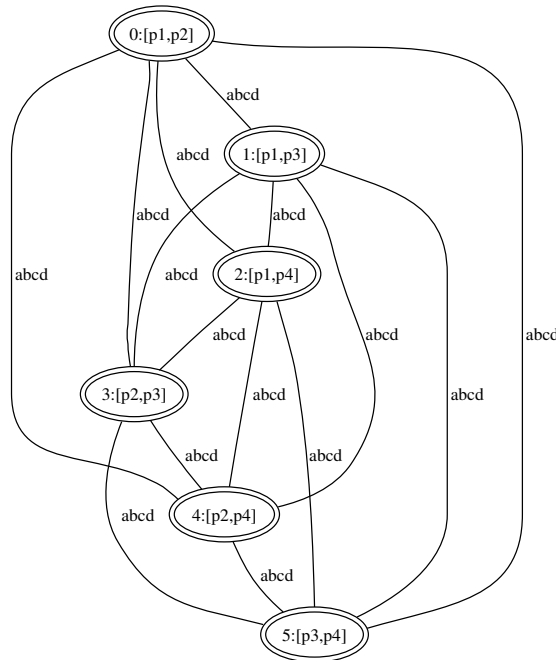


Fig. 2. Caps situation where nobody knows anything about p_1, p_2, p_3, p_4 .

Next, we model the fact that (everyone is aware that) b can see the first cap and that c can see the first and the second cap, as follows:

```
Caps> showM (upds mo1 [info [b,c] p1, info [c] p2])
==> [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])
(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]])
```

Notice that this model reveals that in case a, b wear caps of the same colour (situations 0 and 5), c knows the colour of all the caps, and in case a, b wear caps of different colours, she does not (she confuses the cases 1, 2 and the cases 3, 4). Figure 3 gives a picture representation.

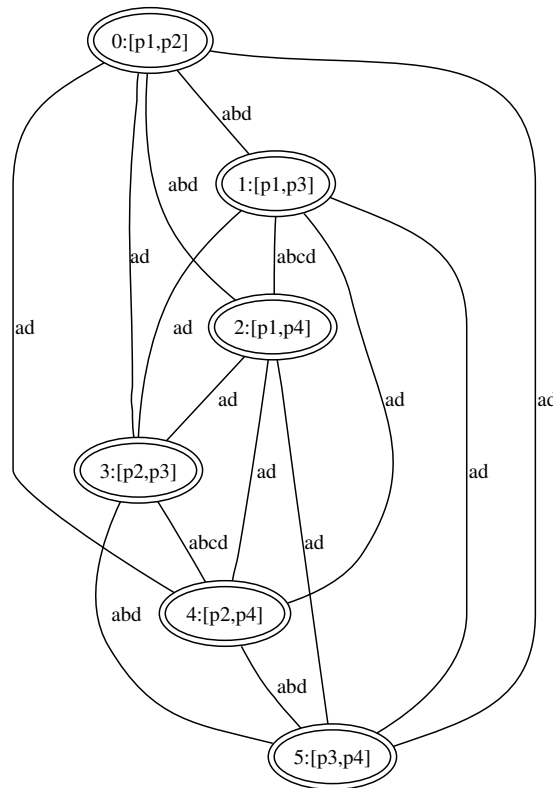


Fig. 3. Caps situation after updating with awareness of what b and c can see.

Let this model be called $mo2$. Knowledge of c about her situation is expressed by the epistemic formula $K_c p_3 \vee K_c \neg p_3$, ignorance of c about her situation by the negation of this formula. Knowledge of b about his situation is expressed by $K_b p_2 \vee K_b \neg p_2$. Let bK , cK express that b, c know about their situation. Then updating with public announcement of cK and with public announcement of the negation of this have different effects:

```
Caps> showM (upd mo2 (public cK))
```

```

==> [0,1]
[0,1]
(0, [p1,p2]) (1, [p3,p4])
(a, [[0,1]])
(b, [[0], [1]])
(c, [[0], [1]])
(d, [[0,1]])

Caps> showM (upd mo2 (public (Neg cK)))
==> [0,1,2,3]
[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]])

```

In both results, b knows about his situation, though:

```

Caps> isTrue (upd mo2 (public cK)) bK
True
Caps> isTrue (upd mo2 (public (Neg cK))) bK
True

```

4 Elements of Secure Communication

An important element of secure communication is the ability to pass information from a to b along an insecure channel in such a way that an eavesdropper c cannot find out what b has learnt. A particular case of that is the Russian Card Problem [4, 6]. See [17, 19] for descriptions of security protocols, and [23] for an attempt at model checking for analysing the security of communication. Applications of epistemic logic, but with rather ad-hoc updates, to the analysis of security protocols can be found in [15].

Here we will merely take an abstract look at the problem. If agent a and b have a link between propositions p and q and a and b are the only ones with this link, then a can inform b in secret about q by means of a public communication about p . Here is how. Assume there are three agents a, b, c . A situation where all agents are ignorant about p and q , and are aware of their common ignorance looks like this:

```

SecCom> showM mo0
==> [0,1,2,3]
[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]])

```

Suppose a has information about p , and a and b either have common knowledge that p and q are equivalent or they have common knowledge that p and $\neg q$ are:

```

SecCom> showM (upds mo0 [info [a] p,link_ab_pq])
==> [0,5,8,9]
[0,1,2,3,4,5,6,7,8,9,10,11]
(0, []) (1, []) (2, []) (3, [p]) (4, [p])
(5, [p]) (6, [q]) (7, [q]) (8, [q]) (9, [p,q])
(10, [p,q]) (11, [p,q])
(a, [[0], [1,6], [2,7], [3,10], [4,11], [5], [8], [9]])
(b, [[0,9], [1,3,6,10], [2,4,7,11], [5,8]])
(c, [[0,1,3,6,9,10], [2,4,5,7,8,11]])

```

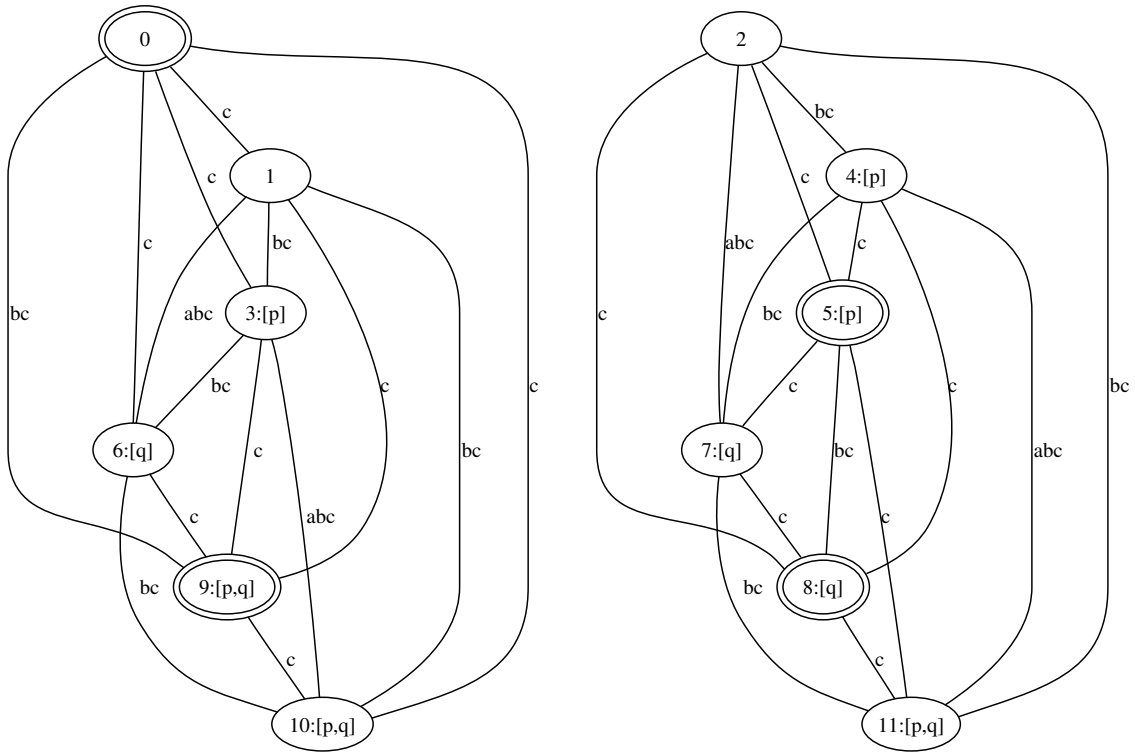


Fig. 4. Situation where a knows whether p and where a, b have common knowledge about a link between p and q .

Call this model `mo1` (see Figure 4 for an alternative representation). In this model, c still knows nothing about q :

```
SecCom> isTrue mo1 (Disj [K c q, K c (Neg q)])
False
```

Also, in this model it is common knowledge among a, b that b knows about the link between p and q . Thus, the result of public announcement of p in this situation is that b knows whether q , while c is still left in the dark about q :

```
SecCom> showM (upd mo1 (public p))
==> [2,3]
[0,1,2,3,4,5]
(0, [p]) (1, [p]) (2, [p]) (3, [p,q]) (4, [p,q])
(5, [p,q])
(a, [[0,4], [1,5], [2], [3]])
(b, [[0,4], [1,5], [2], [3]])
(c, [[0,3,4], [1,2,5]])

SecCom> isTrue (upd mo1 (public p)) (Disj [K b q, K b (Neg q)])
True
SecCom> isTrue (upd mo1 (public p)) (Disj [K c q, K c (Neg q)])
False
```

5 Finding Axiom Schemes for Logics of Communication

An important result of [2] is that epistemic PDL with action modalities has the same expressive power as epistemic PDL [21, 22]. This singles out epistemic PDL as a natural *standard* logic for epistemic updating. This result is based on a constructive program transformation procedure that maps finite action models to program transformers [8]. Since DEMO implements this procedure, it can be used to find epistemic PDL equivalents for arbitrary formulas involving epistemic actions. Here is an equivalent for the formula that expresses that after public announcement that p it is the case that a and b commonly know that $p \leftrightarrow q$:

```
SecCom> tr (Up (public p) (CK [a,b] (p 'equiv' q)))
[(U[?T,C[?p,[a,b]])]*]v[&[p,q],-p]
```

This is the DEMO representation of the epistemic PDL formula

$$[(?T \cup (?p; a \cup b))^*](p \rightarrow q).$$

This formula expresses that at the end of every p path along a or b accessibilities the implication $p \rightarrow q$ holds.

All epistemic actions covered in the examples above map multimodel S5 models to multimodal S5 models, the models that express multiagent knowledge (all accessibility relations are equivalence relations). As soon as epistemic actions involving deception occur, this is no longer the case. A key principle about knowledge is that knowledge implies truth: $K_a\phi \rightarrow \phi$. This no longer holds after a piece of deception. Let k_tr express the formula $K_a(\neg K_b p) \rightarrow \neg K_b p$. Clearly, this is an instance of the principle of truthfulness. After an update with a secret message to b this formula no longer holds:

```
SecCom> isTrue mo0 k_tr
True
SecCom> isTrue (upd mo0 (public p)) k_tr
True
SecCom> isTrue (upd mo0 (secret [b] p)) k_tr
False
```

6 Further Work

As the program listings demonstrate, the DEMO representations of epistemic situations are very concise. A comparison between DEMO and two other model checking tools for epistemic update logic can be found in [6]: “The fastest goal to success was implementing the Russian Cards problem in DEMO.”

At present DEMO is being used in various places around the world for model checking of problems in epistemic update logic [5, 6]. Example code from these analyses are available (or will be made available) as part of the DEMO documentation. Further applications in the area of analysis of security protocols are in preparation.

The current implementation of DEMO does not use monads; a new implementation in terms of state monads is in the making. This will allow dynamic updating, with the current result of all updates so far held in the state.

Acknowledgements Thanks to Hayco de Jong for illuminating comments on an earlier draft, and to Thijs van der Storm for advice on the use of *dot*.

Code availability The Haskell code for DEMO and for the DEMO examples is available from <http://www.cwi.nl/~jve/demo/>.

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/.
- [3] B.F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.
- [4] Hans van Ditmarsch. The Russian card problem. *Studia Logica*, 75:31–62, 2003.
- [5] Hans van Ditmarsch, Ji Ruan, and Rineke Verbrugge. Model checking sum and product. manuscript, University of Otago, New Zealand, 2005.
- [6] Hans van Ditmarsch, Wiebe van der Hoek, Ron van der Meyden, and Ji Ruan. Model checking Russian cards. To appear in Proceedings of MoChArt 05, 2005.
- [7] Jan van Eijck. Dynamic epistemic modelling. Technical Report SEN-E0424, CWI, Amsterdam, December 2004. Available from <http://db.cwi.nl/rapporten/>.
- [8] Jan van Eijck. Reducing dynamic epistemic logic to PDL by program transformation. Technical Report SEN-E0423, CWI, Amsterdam, December 2004. Available from <http://db.cwi.nl/rapporten/>.
- [9] Jan van Eijck. DEMO program and documentation, 2005. Available from <http://www.cwi.nl/~jve/demo/>.
- [10] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.
- [11] J. Gerbrandy. *Bisimulations on planet Kripke*. PhD thesis, ILLC, 1999.
- [12] Joseph Y. Halpern and Moshe Y. Vardi. Model checking vs. theorem proving: A manifesto. In J. Allen, R. E. Fikes, and E. Sandewall, editors, *Proceedings 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning, KR'91*, pages 325–334. Morgan Kaufmann Publishers, San Mateo, CA, 1991.
- [13] J. Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, Ithaca N.Y., 1962.
- [14] M. Hollenberg. *Logic and Bisimulation*. PhD thesis, Utrecht University, 1998.
- [15] A. Hommersom, J.-J. Meyer, and E.P. de Vink. Update semantics of security protocols. *Synthese*, 142:229–267, 2004. Knowledge, Rationality and Action subseries.
- [16] E. Koutsoufios and S. North. Drawing graphs with *dot*. Available from <http://www.research.att.com/~north/graphviz/>.
- [17] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055, pages 147–166. Springer-Verlag, Berlin Germany, 1996.
- [18] Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, 1987.
- [19] O. Pereira. Modelling and security analysis of authenticated group key agreement protocols, 2003.
- [20] 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.
- [21] V. Pratt. Semantical considerations on Floyd–Hoare logic. *Proceedings 17th IEEE Symposium on Foundations of Computer Science*, pages 109–121, 1976.
- [22] V. Pratt. Application of modal logic to programming. *Studia Logica*, 39:257–274, 1980.
- [23] A. W. Roscoe. Modelling and verifying key-exchange protocols using CSP and FDR. In *Proc. 8th IEEE Computer Security Foundations Workshop*, pages 98–107, 1995.