Department of Computer Science, University of Otago

Model Checking Russian Cards

Authors:
Hans van Ditmarsch
Department of Computer Science, University of Otago

Wiebe van der Hoek
Liverpool University

Ron van der Meyden
NICTA/UNSW, Sydney

Ji Ruan
Department of Computer Science, University of Otago; and China

Model Checking Russian Cards[*]

Hans van Ditmarsch[†], Wiebe van der Hoek[‡], Ron van der Meyden[§], Ji Ruan[¶]

Abstract

We implement a specific protocol for bit exchange among card-playing agents in three different state-of-the-art epistemic model checkers and compare the results.

1   Introduction

Protocols based on public/private key encryption assume computationally limited agents. Decryption of messages is tractable for the intended recipient but should be intractable (or at least computationally too expensive) for an eavesdropper, for example, because it is based on factorizing a large product of primes. Instead, we investigate protocols for computationally unlimited agents, also known as 'perfect logicians' or 'ideal agents'. In that case it should not just be intractable but even impossible for an eavesdropper to access the secret content of a message. In a worst-case scenario of that kind, the eavesdropper intercepts all messages entirely. If this scenario is public knowledge, and if all messages are truthful, then, firstly, we can assume a convenient abstract architecture for such systems, namely that of distributed or interpreted systems, where all agents (i.e., sender, receiver, and eavesdropper(s)) at least know their local state; and, secondly, we can assume that all messages are public announcements, and use corresponding logics to specify the properties of protocols for sender and receiver to communicate secret information.

Information-based protocol analysis has proved useful for protocol specification and to uncover protocol flaws [3, 13]. It more particularly applies when protocol requirements are in terms of knowledge and belief, as in zero-knowledge protocols [9], and to the analysis of protocols for ideal agents, such as protocols consisting of public announcements in interpreted systems. Communicative features of interpreted systems have been described in epistemic logics with explicit representation of time (in runs of systems) and/or corresponding temporal epistemic logics [5, 10, 15], and also in dynamic epistemic logics [8, 2, 16]. We focus on the latter, in particular on public announcement logic [2].

In this contribution we assume interpreted systems for card players, as represented by pointed multi-agent Kripke models. A player can communicate secret bits such as card ownership to another player without revealing that to a third player (eavesdropper). This has been investigated in [6, 17, 11, 1, 18]. A typical example is 'Russian Cards': two players each draw three cards from a pack of seven cards, and the remaining player (eavesdropper)

[†]University of Otago, New Zealand, `hans@cs.otago.ac.nz` (**CONTACT PERSON**)

[‡]University of Liverpool, United Kingdom, `wiebe@csc.liv.ac.uk`

[§]NICTA-UNSW, Sydney, Australia, `meyden@cse.unsw.edu.au`

[¶]University of Otago, New Zealand, & China, `neoruan@yahoo.com.cn`

gets the last card. The 'problem' of the Russian Cards problem is to find protocols that allow sender and receiver to learn each other's hand of cards, without the eavesdropper being able to learn that. In [17], protocols of length two are presented that solve it. Protocols of length greater than two are investigated in [18].

In such card protocols, the required postconditions are not always clear or not easy to verify, publicly known protocol features may involve fairly complex nested dynamic epistemic formulas, and enumeration of all possible protocols is an issue as well. For all that, model checkers are promising tools.

Such explorations have been partially carried out for the Russian cards problem in [21]: epistemic properties of the scenario are translated into (linear time) LTL, and then verified using the model checker SPIN. A deal of cards together with a number of announcements corresponds to a time line, not unlike the modeling in MCK presented in this paper. Uncertainty of the agents is represented by exploiting *local propositions* proposed in [4], see also [14].

In this contribution, we verify protocol properties in the epistemic model checkers MCK [7], DEMO [19] and MCMAS [12]. We have selected one specific Russian Cards protocol, the 'five hand protocol', implemented it in these quite different dedicated 'epistemic' model checkers, and verified its relevant properties. This involved reinterpreting dynamic epistemic concepts in temporal epistemic terms; this theoretical exercise was carried out successfully and increased our understanding of dynamic epistemic features. All three implementations were carried out within a reasonable development time and all were successful. Some additional Russian Cards protocol features, in particular for protocols of length greater than two, have been kept outside this comparison. Also, incorrect protocols (such as for non-solutions of the Russian Cards problem) can be easily shown to be so by establishing failure of (commonly known or other) epistemic conditions. This only requires (almost) trivial changes in the scripts presented below for a correct protocol.

In Section 2 we present the Russian Cards problem. Sections 3 to 5 are dedicated to the implementation of the 'five hands' protocol for the Russian Cards problem in the model checkers, respectively, MCK, DEMO, and MCMAS. Section 6 compares the results. The MCK, DEMO, and MCMAS input scripts can be found on `www.cs.otago.ac.nz/staffpriv/hans/aoard/`.[1]

2   Russian Cards

From a pack of seven known cards two players each draw three cards and a third player gets the remaining card. How can the players with three cards openly inform each other about their cards, without the third player learning from any of their cards who holds it?

This 'Russian Cards' problem originated at the Moscow Math Olympiad 2000. Call the players Anne, Bill and Cath, and the cards $0, ..., 6$, and suppose Anne holds $\{0, 1, 2\}$, Bill

$\{3, 4, 5\}$, and Cath card 6. For the hand of cards $\{0, 1, 2\}$, write 012 instead, for the card deal, write 012.345.6, etc. Assume from now on that 012.345.6 is the actual card deal. All announcements must be public and truthful. There are not many things Anne can safely say. Obviously, she cannot say "I have 0 or 6," because then Cath learns that Anne has 0. But Anne can also not say "I have 0 or 3," because Anne does not know if Cath has 3 or another card, and *if* Cath had card 3, she would have learnt that Anne has card 0. But Anne can also not say "I have 0 or 1." Even though Anne holds both 0 and 1, so that she does not appear to risk that Cath eliminates either card and thus gains knowledge about single card ownership (weaker knowledge, about alternatives, is allowed), Cath knows that Anne will not say anything from which Cath may learn her cards. And thus Cath can conclude that Anne will only say "I have 0 or 1" if she actually holds both 0 and 1. And in that way Cath learns two cards at once! The apparent contradiction between Cath not knowing and Cath knowing is not really there, because these observations are about different information states: it is merely the case that announcements may induce further updates that contain yet other information.

Whenever after Anne's announcement it is (at least) not common knowledge to Anne, Bill, and Cath, that Cath remains ignorant of any of Anne's or Bill's cards, this may be informative to Cath after all. A typical example is when *Anne says that she either holds 012 or not any of those cards, after which Bill says that Cath holds card 6.* For details, see [17]. Indeed, a solution requirement is that Cath's ignorance remains public knowledge after any announcement. Such announcements are called *safe*.

A solution to the Russian Cards problem is a sequence of safe announcements after which it is commonly known to Anne and Bill (not necessarily including Cath) that Anne knows Bill's hand and Bill knows Anne's hand. This (instance of a) five hand protocol is a solution:

> Anne says "My hand of cards is one of 012, 034, 056, 135, 246," after which Bill says "Cath has card 6."

Note that Bill's announcement is equivalent to "My hand of cards is one of 345, 125, 024." After this sequence, it is even publicly known that Anne knows Bill's hand and Bill knows Anne's hand. If we extend Anne's announcement with one more hand, namely 245, and if it is public knowledge that the protocols used by Anne and Bill are of finite length (so may consist of more than two announcements), then it is 'merely' common knowledge to Anne and Bill that they know each other's hand, but (disregarding further analysis) Cath considers it possible that they do not know each other's hand of cards. This is a useful security feature for Anne and Bill, as Cath plays the role of the eavesdropper. A further postcondition is that all safe announcements by Anne ensure at least one safe response from Bill, and vice versa. This recursive requirement results in a more complex condition. See [18].

Public announcement logic   The Russian Cards problem can be modelled in public announcement logic with common knowledge. We give a concise overview of the language and its semantics.

Given are a set of agents N and a set of atoms P. An *epistemic model* $M = \langle S, \sim, V \rangle$ consists of a *domain* S of (factual) *states* (or 'worlds'), *accessibility* $\sim : N \to \mathcal{P}(S \times S)$, and a *valuation* $V : P \to \mathcal{P}(S)$. For $s \in S$, (M, s) is an *epistemic state*. For $\sim (n)$ we write $\sim_n$, and for $V(p)$ we write V_p. So, access $\sim$ can be seen as a set of equivalence relations $\sim_n$, and V as a set of valuations V_p. For $(\bigcup_{n \in G} \sim_n)^*$, write $\sim_G$: this is access to interpret common knowledge for group G.

The language of public announcements is inductively defined as

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid K_n\varphi \mid C_G\varphi \mid [\varphi]\psi$$

where $p \in P$, $n \in N$, and $G \subseteq N$ are arbitrary. For $K_n\varphi$, read 'agent n knows formula φ'. For $C_G\varphi$, read 'group of agents G commonly know formula φ'. For $[\varphi]\psi$, read 'after public announcement of φ, formula ψ (is true)'. The effect of the public announcement of φ is the restriction of the epistemic state to all worlds where φ holds. So, 'announce φ' can be seen as an information state transformer, with a corresponding dynamic modal operator $[\varphi]$.

The semantics is as follows. Given is an epistemic model $M = \langle S, \sim, V \rangle$.

$$
\begin{array}{lll}
M, s \models p & \text{iff} & s \in V_p \\
M, s \models \neg\varphi & \text{iff} & M, s \not\models \varphi \\
M, s \models \varphi \wedge \psi & \text{iff} & M, s \models \varphi \text{ and } M, s \models \psi \\
M, s \models K_n\varphi & \text{iff} & \text{for all } t \in S : s \sim_n t \text{ implies } M, t \models \varphi \\
M, s \models C_G\varphi & \text{iff} & \text{for all } t \in S : s \sim_G t \text{ implies } M, t \models \varphi \\
M, s \models [\varphi]\psi & \text{iff} & M, s \models \varphi \text{ implies } M|\varphi, s \models \psi
\end{array}
$$

Model $M|\varphi = \langle S', \sim', V' \rangle$ is defined as

$$
\begin{array}{lll}
S' & := & \{ s' \in S \mid M, s' \models \varphi \} \\
\sim'_n & := & \sim_n \cap (S' \times S') \\
V'_p & := & V_p \cap S'
\end{array}
$$

In other words: the model $M|\varphi$ is the model M restricted to all the states where φ holds, including access between states. Formula φ is valid on model M, notation $M \models \varphi$, if and only if for all states s in the domain of M: $M, s \models \varphi$. Formula φ is valid, notation $\models \varphi$, if and only if for all models M: $M \models \varphi$.

We now model the Russian Cards problem in this logic. Given a stack of known cards and some players, the players blindly draw some cards from the stack. In a state where cards are dealt in that way, but where no game actions of whatever kind have been done, it is commonly known what the cards are, that they are all different, how many cards each player holds, and that players only know their own cards. From the last it follows that two deals are *the same for an agent*, if she holds the same cards in both, and if all players hold the same number of cards in both. This induces an equivalence relation on deals.

An epistemic model $(Rus, 012.345.6)$ for the deal 012.345.6 that we investigate, encodes the knowledge of the players Anne, Bill and Cath (a, b, c) in this card deal. It consists of $\binom{7}{3}\binom{4}{3}\binom{1}{1} = 140$ deals. For each player, access between states is induced by the equivalence above, for example, 012.345.6 $\sim_a$ 012.346.5 says that Anne cannot tell these two card deals apart (as her hand is 012 in both). Facts about card ownership written as q_n, for 'card q is held by player n'. The valuation V_{0_a} of fact 0_a (Anne holds card 0) consists of all 60 deals where 0 occurs in Anne's hand, etc.

After a sequence of announcements that is a solution of the Russian Cards problem, it should hold that Anne knows Bill's cards, that Bill knows Anne's cards, and that Cath doesn't know any of Anne's or Bill's cards:

$$
\begin{array}{lll}
\textsf{a\_knows\_bs} & := & \bigwedge_{q=0..6} (K_a q_b \vee K_a \neg q_b) \\
\textsf{b\_knows\_as} & := & \bigwedge_{q=0..6} (K_b q_a \vee K_b \neg q_a) \\
\textsf{c\_ignorant} & := & \bigwedge_{q=0..6} (\neg K_c q_a \wedge \neg K_c q_b)
\end{array}
$$

4

We suggested in the previous section that these conditions are too weak. This can be exemplified by the observation that, e.g.,

$$Rus, 012.345.6 \models [K_a(012_a \vee (\neg 0_a \wedge \neg 1_a \wedge \neg 2_a))][\mathsf{c\_ignorant}]\neg\mathsf{c\_ignorant}$$

After Anne says that her hand is 012 or that she does not hold any of those cards, $\mathsf{c\_ignorant}$ is true, but a further update with that (in other words: when Cath can assume that this is true) makes Cath learn some of Anne's cards, so that $\mathsf{c\_ignorant}$ is false. The actually required postconditions avoiding such complications are: after every announcement of an executed protocol, it is publicly known that Cath is ignorant, and after the execution of the entire protocol it is commonly known to Anne and Bill that: Anne knows that Bill knows her hand of cards, and Bill knows that Anne knows his hand of cards. Also using that $C_{ab}(K_b\mathsf{a\_knows\_bs} \wedge K_a\mathsf{b\_knows\_as})$ is equivalent to $C_{ab}(\mathsf{a\_knows\_bs} \wedge \mathsf{b\_knows\_as})$ this is formalized as

$$C_{ab}(\mathsf{a\_knows\_bs} \wedge \mathsf{b\_knows\_as})$$
$$C_{abc}\mathsf{c\_ignorant}$$

Concerning protocols: when Anne announces 'φ', this should be interpreted as '$K_a\varphi$' given that she knows what she says, and even as '$K_a\varphi \wedge [K_a\varphi]K_a\mathsf{c\_ignorant}$' given her intention, and beyond that even as '$K_a\varphi \wedge [K_a\varphi]C_{abc}K_a\mathsf{c\_ignorant}$' given that her intention is public. One can then show that

$$Rus, 012.345.6 \models [K_a\varphi \wedge [K_a\varphi]C_{abc}\mathsf{c\_ignorant}]C_{abc}\mathsf{c\_ignorant}$$

So in this case the intention is indeed realized, unlike above: the announcement is safe. We ignore the further complication that safe announcements require safe responses in this submission. The solution given in Section 2 consists of the successive announcements

$$
\begin{aligned}
\mathsf{a\_announce} \quad &:= \quad 012_a \vee 034_a \vee 056_a \vee 135_a \vee 246_a \\
\mathsf{b\_announce} \quad &:= \quad 6_c
\end{aligned}
$$

3   Model Checker MCK

MCK, for 'Model Checking Knowledge', is a prototype model checker for temporal and knowledge specifications, developed by Peter Gammie and Ron van der Meyden [7]. The overall setup supposes a number of agents acting in an environment, by temporal development. This is modelled by an interpreted system where agents perform actions according to a protocol. Actions and the environment may be only partially observable at each instant in time.

Different approaches to the temporal and epistemic interaction and development are implemented. Knowledge may be based on current observations only, on current observations and clock value, and on the history of all observations and clock value. The last corresponds to synchronous perfect recall. We have used that approach. In the temporal dimension, the specification formulas may describe the evolution of the system along a single computation, i.e., using linear time temporal logic (LTL), or they may describe the branching structure of all possible computations, i.e, using branching time or computation tree logic (CTL). We have used LTL. See www.cse.unsw.edu.au/~mck/ for more information.

Russian Cards in MCK   In MCK, we have to reinterpret the dynamic epistemics of Section 2 in temporal epistemic terms. In a program rus.mck we successively introduce environmental

variables and initialize those; we create three agents `A`, `B`, and `C` with corresponding protocols `"anne"`, `"bill"` and `"cath"`; a main part of the program specifies the (temporal) transitions, induced by card dealing and the announcements, that relate different information states for these players; finally `rus.mck` contains a part with various to be verified properties of the timelines created.

A hand of cards of an agent is encoded by a list of seven booleans, for example `a_hand : Bool[7]` specifies for all of the cards 0, ..., 6 whether they are held by Anne or not, such that `anne_cards[0]` is true when Anne holds card 0, etc. Initially, such variables are set to false.

Agent A, for Anne, is created by

```
agent A "anne" (a_hand, a_announce, b_announce, stage)
```

The name of the agent is `A`. It uses protocol `"anne"`. It can interact with, and potentially observe the variables between parentheses. The first of those is, obviously, only observable by Anne, the others will reappear in the other agent definitions, as they are publicly observable. The variable `stage` is the 'clock tick'.

The `transitions` part of `rus.mck` specify what happens in different stages of the execution of the protocol. We distinguish stages (clock ticks) 0, 1, 2, and 3. In stage 0 the cards are dealt to the players, in the order 0, ..., 6. We show it up to the dealing of card 0.

```
 stage == 0  ->
   begin   if
              na < 3  -> begin a_hand[0]:=True; na:= na+1 end []
              nb < 3  -> begin b_hand[0]:=True; nb:= nb+1 end []
              nc == 0 -> begin c_hand[0]:=True; nc:= 1 end
          fi;
```

Variables `na`, `nb`, and `nc` are counters to record how many cards agents have, and `[]` means nondeterministic choice. In this part of the transitions, 140 different deals are created, represented as 140 different timelines.

In stage 1, Anne announces that her hands is one of 012, 034, 056, 135, and 246. This is done indirectly by executing the protocol `"anne"`, that contains a condition corresponding to these five deals, which causes the action `Announce` to be executed. This then results in the atom `a_announce` becoming true.

```
 stage == 1 /\ A.Announce -> a_announce := True
```

In stage 2, Bill announces that Cath holds card 6. Alternatively, one can model that Bill announces Cath's card – whatever it is. Bill's announcement is by way of an action `B.Announce`, and results in the variable `b_announce` to become true. This is the transition to stage 3, the final stage. We can imagine the whole system to consist of 140 different runs. Whether variables `a_announce` and `b_announce` are true in stage 2 and stage 3, respectively, depends on the deal in that run.

The protocol for Anne is

```
protocol "anne" (cards: observable Bool[7],
      a_announce: observable Bool, b_announce: observable Bool,
      stage: observable Counter)
```

```
begin
  skip; if
   ( (cards[0] /\ cards[1] /\ cards[2]) \/  (cards[0] /\ cards[3] /\ cards[4]) \/
     (cards[0] /\ cards[5] /\ cards[6]) \/  (cards[1] /\ cards[3] /\ cards[5]) \/
     (cards[2] /\ cards[4] /\ cards[6]) )
    -> <<Announce>>
  fi
end
```

The 'begin-end' part of this protocol specifies for each of the stages 0, 1, and 2 what happens in that stage. In stage 0 nothing happens: `skip`. In stage 1, the action `Announce` – that is, whatever is found between $<<$ and $>>$ – is executed. Actually, the value or instance of `cards` for Anne is `a_cards`; see above, where Anne is created. Alternatively to five actual hands, a much longer protocol creates five arbitrary hands of cards based on Anne's actual hand. Nothing is specified for stage 2: this is therefore `skip` again by default. Bill has a similar protocol but his protocol starts with `skip ; skip`, as his announcement is in stage 2. And Cath does not act at all, which carries the protocol `skip ; skip ; skip`.

The knowledge of the agents evolves with every stage, via the agents' limited access to the environment. Initially, they only observe their own hand of cards, and Anne's and Bill's public announcement is accessed by all agents. Anne cannot distinguish two states iff her observations are the same in those states. For example, in stage 1 Anne cannot distinguish the timelines for deals 012.345.6 and 012.346.5, because: both have the same `a_hand` values (for all seven variables), `a_announce` is true in both cases, and `b_announce` is false is both cases. But in stage 3, Anne can distinguish these timelines, since `b_announce` is true for the former and false for the latter.

A final part of `rus.mck` lists various temporal epistemic properties to be checked. For example, we want to verify that $Rus, 012.345.6 \models [\text{a\_announce}][\text{b\_announce}]C_{ab}\text{a\_knows\_bs}$. The current version (0.2.0) of MCK does not support common knowledge operators for specification in the perfect recall module. Therefore we verify instead that in stage 3, a_knows_bs is valid in the model. This corresponds to $Rus|\text{a\_announce}|\text{b\_announce} \models \text{a\_knows\_bs}$ which ensures that $Rus|\text{a\_announce}|\text{b\_announce}, 012.345.6 \models C_{abc}\text{a\_knows\_bs}$. And in this specific model $C_{ab}\text{a\_knows\_bs} \leftrightarrow C_{abc}\text{a\_knows\_bs}$ is also true.

```
spec_spr_xn = X 3 ( (a_announce /\ b_announce) =>
  (  (((Knows A b_hand[0]) \/ (Knows A neg b_hand[0]))) /\
            (...)
     (((Knows A b_hand[6]) \/ (Knows A neg b_hand[6]))) ))
```

The part `spec_spr_xn` means that we are using the perfect recall module of MCK, and `X 3` is the triple 'next state' temporal operator, counting from stage 0. Therefore, the formula bound by the operator is checked in stage 3. Similarly, other properties of the five hands protocol are verified.

4   Model Checker DEMO

The tool DEMO is developed by Jan van Eijck [19]. DEMO is short for Dynamic Epistemic MOdelling. It allows modelling epistemic updates, graphical display of Kripke structures involved (i.e., epistemic or state models, and action models that represent epistemic actions),

formula evaluation in epistemic states, etc. Epistemic models are minimized under bisimulation, and action models are minimized under the (more appropriate, weaker) notion of action emulation [20]. DEMO is written in the functional programming language Haskell. See also www.cwi.nl/~jve/papers/04/demo/.

The model checker DEMO implements the dynamic epistemic logic of [2]. In this 'action model logic' the global state of the multi-agent system is represented by an epistemic model (multi-agent Kripke model), and the agents' action is represented by an action model. An action model is also based on a multi-agent Kripke frame, but instead of carrying a valuation it has a precondition function which assigns a precondition to each point in the action model, which stands for an atomic action. The state change in the system is via an operation called update product. This is a restricted modal product. In this submission we restrict our attention to action models for public announcements. Such action models have a singleton domain. We refrain from details and proceed with the recursive definition of formulas in DEMO.

```
Form = Top | Prop Prop | Neg  Form | Conj [Form] | Disj [Form]
       | K Agent Form | CK [Agent] Form
```

Formula `Top` stands for $\top$, `Prop Prop` for atomic propositional letters (the first occurrence of `Prop` means that the datatype is 'propositional atom', whereas the second occurrence of `Prop` is the placeholder for an actual proposition letter, such as P0), `Neg` for negation, `Conj` `[Form]` stands for the conjunction of a list of formulas of type `Form`, similarly for `Disj`, `K` `Agent` stands for the individual knowledge operator for agent `Agent`, and `CK [Agent]` for common knowledge operator for the group of agents listed in `[Agent]`.

A pointed (and singleton) action model for a public announcement is created by a function `public` with a precondition (formula) as argument. The update operation is specified as

```
upd ::  EpistM -> PoAM -> EpistM
```

Here, `EpistM` is an epistemic state and `PoAM` is a pointed action model. The update generates a new epistemic state as specified above. Formula checking is defined as

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

Its arguments are an epistemic state and a formula, and it returns a boolean value.

Russian Cards in DEMO   In DEMO, one is restricted to propositional letters starting with lower case p, q and r, so we cannot write, for example, 0_a for the atomic proposition that Anne holds card 0, as in Section 2. Instead, atoms $\{p, \ldots, p6, q, \ldots, q6, r, \ldots, r6\}$ represent such atomic propositions. The name `p4` – Anne holds card 4 – actually stands for `Prop (P 4)`, etc. Instead of `p0` we write, somewhat arbritrarily, `p`, and similarly for `q` and `r`.

The initial epistemic state `rus` representing the knowledge in card deal 012.345.6 is constructed as follows. A set of integers `[0..139]` represents the 140 different deals. Each integer is associated with seven propositional letters. The first two deals are

```
(0,[P 0,P 1,P 2,Q 3,Q 4,Q 5,R 6]), (1,[P 0,P 1,P 2,Q 3,Q 4,Q 6,R 5])
```

The deal numbered 0 stands for actual deal 012.345.6. A pair of two integers is in the accessibility relation for an agent n, if that agent holds the same cards in both deals. Two such pairs for Anne are (a,0,0),(a,0,1). DEMO assumes arbitrary accessibility relations. So, unfortunately, we have to explicitly list all pairs in the equivalence relation for each agent, as above.

Anne's public announcement a_announce corresponds to the following singleton action model named anne5, which is produced by the function public.

```
public( K a (Disj[Conj[p,p1,p2],Conj[p,p3,p4],Conj[p,p5,p6],
            Conj[p1,p3,p5],Conj[p2,p4,p6]]))
```

Similarly, we have an action model billkc for Bill's announcement b_announce. The postcondition that Anne knows Bill's hand of cards, a_knows_bs, is represented as

```
aknowsbs = Conj[ Disj[K a q,  K a (Neg q) ], Disj[K a q1,  K a (Neg q1) ],
              Disj[K a q2, K a (Neg q2) ], Disj[K a q3,  K a (Neg q3) ],
              Disj[K a q4, K a (Neg q4) ], Disj[K a q5,  K a (Neg q5) ],
              Disj[K a q6,  K a (Neg q6) ]]
```

Similarly for b_knows_as and c_ignorant. The model checker now verifies the postconditions of the constructed models. After Bill's announcement it is common knowledge to Anne and Bill that Anne knows Bill's hand of cards, and it is also common knowledge to Anne and Bill that Bill knows Anne's hand of cards. It is publicly known that Cath remains ignorant:

```
*RUS>isTrue (upd (upd rus anne5) billkc) (CK [a,b] aknowsbs])
True
*RUS>isTrue (upd (upd rus anne5) billkc) (CK [a,b] bknowsas])
True
*RUS>isTrue (upd (upd rus anne5) billkc) (CK [a,b,c] cignorant)
True
```

The epistemic state (upd (upd rus anne5) billkc) is the result of updating epistemic state (upd rus anne5) with singleton pointed action model billkc, and (upd rus anne5) was the result of updating the initial epistemic state rus with the action model representing the announcement of a_announce.

5   Model Checker MCMAS

MCMAS presumably stands for Model Checking Multi-Agent Systems. This model checker has been developed by Franco Raimondi and Alessio Lomuscio [12]. The current version is mcmas 0.6. System descriptions and protocol properties are verified using ordered binary decision diagrams, comparable to the approach used in MCK. It extends existing obdd-based techniques for reactive systems by adding both an epistemic (ATL) and a deontic dimension to the logical language, and allowing input in terms of interpreted systems. MCMAS is implemented in C++. See www.cs.ucl.ac.uk/staff/F.Raimondi/MCMAS/ for more information.

In MCMAS, the global state is represented as a tuple of the local states of the agents. For Russian Cards, agents Anne, Bill, and Cath represent players, and an agent Env (the

environment) represents the card deal. The local state of agent Anne requires five components, that can be seen as variables; three represent her hand of cards, and two the status quo and outcome of the two announcements. Version 0.6 of MCMAS does not support variables in the description of agents' local states. Therefore we encode the variable parts in a single string. For example, one local state for Anne is `a012tf`. This means that Anne holds cards 0,1, and 2, that Anne's announcement `a_announce` has been (truthfully) made in the global state of which this local state is a component, and that Bill's announcement `b_announce` could not be made (was false) in that global state. Similarly, we have five variables for Bill, and three variables for Cath. The local state of the agent `Env` has seven variables, because it represent a card deal. An example is `e0123456`. This stands for the actual deal 012.345.6.

The information changes take the usual steps: (1) the cards are revealed to the agents, (2) Anne announces `a_announce`, and (3) Bill announces `b_announce`. All reachable global states will be included in the next stage. An example initial global state is (`annnnn, bnnnnn, cnnn, e0123456`); an 'n' essentially means that the agent has no information on the value of corresponding variable, modelled by giving the variable that value `n`. So, `bnnnnn` means that Bill's local state is that he does not know his cards yet (the first three n's), that Anne has not made her announcement yet (the fourth n) and that Bill has not made his announcement yet. The above global state (`annnnn, bnnnnn, cnnn, e0123456`) then transits to (`a012nn, b345nn, c6nn, e0123456`), where each agent knows what cards it holds. Anne's `a_announce` is then made, causing the transition to (`a012tn, b345tn,c6tn, e0123456`) and `b_announce` finally results in (`a012tt, b345tt,c6tt, e0123456`) – this time, Bill's announcement is successful. These state transitions are specified in the program. For example, for agent Anne, the transition for step one is as follows; `Lstate` is the local state of (current) agent `Anne`, and `Env.Lstate` is the local state of `Env`.

```
a012nn if (Lstate=annnnn and ( Env.Lstate=e0123456 or Env.Lstate=e0123465  or
     Env.Lstate=e0123564 or Env.Lstate=e0124563 ));
```

The environment `Env` does not change during transitions, but this has to be made explicit as

```
e0123456 if Lstate=e0123456;
```

In the 'valuation' part of an MCMAS program we define what can be seen as (the denotation of) atomic propositions. For example

```
ab_d0123456 if (Anne.Lstate=a012tt and Bill.Lstate=b345tt and
            Cath.Lstate=c6tt and Env.Lstate=e0123456);
```

is the atom that is (uniquely) true in the global state (`a012tt, b345tt,c6tt, e0123456`). Similarly, atoms expressing card ownership such as 0_a for 'Anne holds card 0' are defined by enormous expressions starting as (and consisting of 60 alternative card deals)

```
a0 if (Env.Lstate=e0123456 or Env.Lstate=e0123465 or ...
```

Groups of agents can be named too. This is useful when checking common knowledge. For example `ABC={Anne, Bill, Cath};` gives the group consisting of Anne, Bill, and Cath the label `ABC`. A common knowledge formula such as $C_{abc}(0_a \rightarrow K_a 0_a)$ is then represented as `CK(ABC,a0->K(Anne,a0))`. We conclude this short exposition with the postcondition C_{abc}`c_ignorant` that verifies that Cath remains ignorant after both announcements have been made – '!' stands for negation.

```
ab_d0123456 -> GCK(ABC,(
  ( !K(Cath,a0) and !K(Cath,b0) ) and ( !K(Cath,a1) and !K(Cath,b1) ) and
  ( !K(Cath,a2) and !K(Cath,b2) ) and ( !K(Cath,a3) and !K(Cath,b3) ) and
  ( !K(Cath,a4) and !K(Cath,b4) ) and ( !K(Cath,a5) and !K(Cath,b5) ) and
  ( !K(Cath,a6) and !K(Cath,b6) ) ));
```

6 Comparison

We did not compare the model checkers on their computational powers. Currently, none of
our checks took more than a few minutes. Mostly, we were interested in how versatile the
tools appeared to be, to implement a problem that was originally formulated in local, and
dynamic epistemic, terms, into temporal epistemic terms and/or as an interpreted system. In
other words, we were more than anything else interested in development time and supported
functionality. Conclusions based on our experiences are extremely tentative. The fastest goal
to success was implementing the Russian Cards problem in DEMO. This took about half a
day, for Ji Ruan, who is an expert in DEMO. But it took Ron van der Meyden, expert in MCK,
also half a day to write the initial version of the MCK implementation – though that was
finished much later by Hans and Ji, non-experts in MCK. Currently, MCK does not support
common knowledge (in the used module), nor epistemic preconditions, nor preconditions to
temporal formulas. The last makes it impossible to have knowledge preconditions to players'
announcements. Such preconditions are *always* epistemic, as agents only announce what they
know to be true. Also, unsuccessful updates – formulas that become false because they are
announced – cannot be made visible in the way they have to be checked in MCK: the analogue
is a conditional formula where the antecedent is also a subformula of the temporal consequent.
On the other hand, MCK allows a very natural formalization of protocols – this is not, or
less, possible in DEMO or MCMAS. The 'fully interpreted system' approach of MCMAS is
very transparent, but the models that need to be built are 'very' large: (automated input of)
thousands of lines of code, as opposed to (manual input of) about a hundred lines of code
in MCK. More than anything, this case-study increased our insight into the state of the art
in epistemic model checking, and our understanding of the theoretical issues involved in card
cryptography, emerging from the need to reformulate these issues in different logics.

7 Conclusions

We have implemented the five hand protocol to solve the Russian Cards problem in the
model checkers MCK, DEMO, and MCMAS. Dynamic epistemic requirements can be easily
reformalized in temporal epistemic terms, a necessary requirement for formalization in MCK
and MCMAS. The model checkers vary in how easy, or difficult, it is to build the initial
epistemic state, in how difficult it is to formalize announcements and execute them in that
initial state, and in how to verify protocol properties. We intend to pursue this investigation
by implementing more complex protocols and verifying more complex properties for such 'card
cryptography', and generalize it to the level of interpreted systems with agent dependencies,
where groups of agents aim to share their local state value while keeping it a secret from the
remaining agents.

References

[1] M.H. Albert, R.E.L. Aldred, M.D. Atkinson, H.P. van Ditmarsch, and C.C. Handley. Safe communication for card players by combinatorial designs for two-step protocols. *Australasian Journal of Combinatorics*, 2005. To appear.

[2] A. Baltag and L.S. Moss. Logics for epistemic programs. *Synthese*, 139:165–224, 2004. Knowledge, Rationality & Action 1–60.

[3] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8:18–36, 1990.

[4] K. Engelhardt, R. van der Meyden, and Y. Moses. Knowledge and the logic of local propositions. In I. Gilboa, editor, *Proceedings of TARK VII*, pages 29–41. Morgan Kaufmann, 1998.

[5] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge MA, 1995.

[6] M.J. Fischer and R.N. Wright. Bounds on secret key exchange using a random deal of cards. *Journal of Cryptology*, 9(2):71–99, 1996.

[7] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In R. Alur and D. Peled, editors, *Proceedings of CAV 2004*, pages 479–483. Springer, 2004.

[8] J.D. Gerbrandy and W. Groeneveld. Reasoning about information change. *Journal of Logic, Language, and Information*, 6:147–169, 1997.

[9] O. Goldreich. Zero-knowledge protocols twenty years after its invention. Manuscript, 2004.

[10] J.Y. Halpern, R. van der Meyden, and M.Y. Vardi. Complete axiomatizations for reasoning about knowledge and time. *SIAM Journal on Computing*, 33(3):674–703, 2004.

[11] K. Koizumi, T. Mizuki, and T. Nishizeki. Necessary and sufficient numbers of cards for the transformation protocol. In K.-Y. Chwa and J. Ian Munro, editors, *Proceedings of COCOON 2004*, LNCS 3106, pages 92–101. Springer, 2004.

[12] Franco Raimondi and Alessio Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: An algorithm and its implementation. In *Proceedings of AAMAS 2004*, pages 630–637. IEEE Computer Society, 2004.

[13] R. Ramanujam and S. P. Suresh. Information based reasoning about security protocols. *Electr. Notes Theor. Comput. Sci.*, 55(1), 2001.

[14] W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In D. Bošnački and S. Leue, editors, *Proceedings of SPIN 2002 (LNCS 2318)*, pages 95–111. Springer, 2002.

[15] R. van der Meyden. Common knowledge and update in finite environments. *Information and Computation*, 140(2):115–157, 1998.

[16] H.P. van Ditmarsch. Descriptions of game actions. *Journal of Logic, Language and Information*, 11:349–365, 2002.

[17] H.P. van Ditmarsch. The russian cards problem. *Studia Logica*, 75:31–62, 2003.

[18] H.P. van Ditmarsch. The case of the hidden hand. In *Liber Amicorum Dick de Jongh*, 2004.

[19] J. van Eijck. Dynamic epistemic modelling. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam, 2004. CWI Report SEN-E0424.

[20] J. van Eijck and J. Ruan. Action emulation. manuscript, 2005.

[21] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Model checking a knowledge exchange scenario. *Applied Artificial Intelligence*, 18(9-10):937–952, 2004.