Modeling Epistemic Actions in Dynamic Epistemic Logic using Coq

Marko Maliković
The Faculty of Humanities and Social Sciences
University of Rijeka
Omladinska 14, 51000 Rijeka, Croatia
marko.malikovic@ffri.hr

Mirko Čubrilo
Faculty of Organization and Informatics
University of Zagreb
Pavlinska 2, 42000 Varaždin, Croatia
mirko.cubrilo@foi.hr

Abstract. *In this paper we reason about knowledge in multiagent systems composed of intelligent agents by using Coq - a formal proof management system. We use the dynamic logic of common knowledge which is an extension of common knowledge logic with a dynamic operator that enables us to express the epistemic consequences of epistemic actions of agents in the form of agents' knowledge about the state of the system, knowledge about other agents' knowledges, higher-order agents' knowledge and so on, up to common knowledge. We define epistemic actions as a special type in Coq, which allows us to add a general form of interchange principle that connects knowledge and time in systems with perfect recall. As an example of multiagent systems we consider knowledge games defined by van Ditmarsch. To the best of our knowledge, there are no papers in which such games are considered using a Coq proof assistant. We use an axiomatization of such games as given by van Ditmarsch but extended with some new axioms required in our approach. Due to a deficit in implementations grounded in theory which enable players to compute their knowledge in any state of the game, our approach provides a good basis for it.*

Keywords. Multiagent systems, Dynamic logic of common knowledge, Epistemic actions, Coq

1 Introduction

In this paper we reason about knowledge in multiagent systems composed of intelligent agents by using *Coq* - a formal proof management system. Coq is available for download at http://coq.inria.fr and his complete theory and possibility of practical applications is given in [6] and [29]. Coq implements a program specification and mathematical higher-level language called *Gallina* that is based on an expressive formal language called the *Calculus of Inductive Constructions*. It combines both higher-order logic and a richly-typed functional programming language. Through a *vernacular* language [29, 39] which is the language of commands of Gallina, Coq allows: to define functions or predicates that can be evaluated efficiently; to state mathematical theorems and software specifications; to interactively develop formal proofs of these theorems; to machine-check these proofs; to extract certified programs to languages like Objective Caml[1], Haskell or Scheme. As a proof development system, Coq provides interactive proof methods and a tactic language [9], [6, 61], [29, 213] for letting the user define its own proof methods.

Due to the many different definitions of multiagent systems that are present in the literature, the following definition will suffice in this paper: "Multiagent systems are those systems that include multiple autonomous entities with either diverging information or diverging interests, or both." [28, xvii]. Under "entities" we mean intelligent agents that are computer systems that have the following characteristics: autonomy, social ability, reactivity, pro-activeness, veracity, benevolence and rationality.[2]

In order to reason about agents' knowledge in a multiagent system, propositional classical logic is insufficient. In propositional logic, a formula can be true or false. But in the area of multiagent systems in which multiple agents are in different mutual interactions, it is necessary to introduce

[1] Coq is written in the OCaml language [19], with a bit of C.

[2] The meaning of these terms can be found in [43, 118].

other modalities for the truth. Some of these modalities are "necessarily true", "known to be true", "believed to be true" and others. Therefore propositional logic extends to the various types of modal logic [41, 267], [40, 335], [28, 396], [26]. Modal logic is interesting for us due to the possibility of epistemic interpretation and the interpretation of the modal operator as *knowledge* operator. Such logic we call *epistemic logic* [7], [11], [14], [32], [26], [42], [41, 278]. To study multiagent systems we use multimodal propositional epistemic logic $S5_n$ [11, 59]. In accordance with [11, 31], language of $S5_n$ logic is the language of classical propositional logic extended by modal operators for knowledge K_1, ..., K_n, where index number indicates the agent. Expression of $K_i\varphi$ we read "agent *i* knows the formula φ". Formally, the language of $S5_n$ logic is defined as follows:

1. Each atomic proposition (atom) is an epistemic formula[3]
2. If φ and ψ are epistemic formulae then $\neg\varphi$, $\varphi\wedge\psi$, $\varphi\vee\psi$, $\varphi \rightarrow \psi$, $\varphi \leftrightarrow \psi$ are epistemic formulae
3. If φ is epistemic formula then $K_i\varphi$ is epistemic formula for i = 1, ..., n

Axioms of $S5_n$ logic that apply to each of the agents are the following:

1. Axioms of classical propositional logic
2. Distribution axiom: $(K_i\varphi \wedge K_i(\varphi \rightarrow \psi)) \rightarrow K_i\psi$
3. Knowledge axiom: $K_i\varphi \rightarrow \varphi$
4. Positive introspection: $K_i\varphi \rightarrow K_iK_i\varphi$
5. Negative introspection: $\neg K_i\varphi \rightarrow K_i\neg K_i\varphi$

The rules used to perform logical consequences are the *Modus ponens* (MP) from classical propositional logic and *Knowledge generalization rule* (RN)[4] for epistemic logic:

MP: "From $\varphi \rightarrow \psi$ and φ infer ψ"
RN: "From φ infer $K_i\varphi$"

Multimodal epistemic logic can be extended with three new operators that are related to the knowledge in entire groups of agents. The first of these operators is the operator E for *shared knowledge*, which read "everybody knows". If all agents in the group of agents A know the formula φ we write $E_A\varphi$.[5] The second term is *distributed knowledge*. Informally, distributed knowledge is knowledge that has an omniscient observer of a group of agents, with the ability to know each agent's knowledge, to "pool" the collective knowledge of the group of agents, and generally to deduce more than any one agent in the group. The concept of distributed knowledge will not be used in the paper, but it's formal description may be found in [11], [41] and [42]. The third operator is the operator of *common knowledge*. The definition of common knowledge in many references is given informally alike as follows: "A fact φ is a common knowledge in group of agents if everyone knows φ, everyone knows that everyone knows φ, and so on", and we write $C_A\varphi$. The formal definition of common knowledge as a fixed point in accordance with [11, 433] and [28, 406] is: $C_A\varphi$ if and only if $E_A^k\varphi$ for k=1,2,... where is $E_A^k\varphi = E_AE_A^{k-1}\varphi$. The multimodal epistemic logic with a modal operator which describes common knowledge condition we call *common knowledge logic*.

2 Epistemic logic in Coq

Before we introduce epistemic logic by using Coq we have to open a new section and to import module *List* because in some points on our system we use lists:[6]

```
Section EaDel.
Require Import List.
```

We introduce agents as enumerated inductive type *Agents* used to describe finite sets [6, 137]. In the example of the multiagent system which we considered in Section 3, we have seven agents, so our definition consists of seven constructors. We define the whole group of agents as list *G*:[7]

[3] Atoms describe some state of affairs in the "actual world".

[4] *RN* acronym comes from the *Rule of Necessitation* in modal logic.

[5] Note that in the case when all agents in a group know some formula that does not mean that any of these agents know anything about the knowledge of other agents.

[6] Coq version 8.0 or higher has to be run with the *-impredicative-set* option because the definition of *proposition* which we introduce below is based on a non predicative notion of *Set* [13, 31].

[7] Most often the agents are in Coq defined as natural numbers. It may seem more natural, because in that case we could say that the definition of agents is more general. Also, the definition we have introduced may seem somewhat "cumbersome". But, in this paper, the set of agents which makes the multiagent system is required to be finite and for this reason, we do not want to use the infinite structures to describe finite types.

```
Inductive Agents : Set := A1 | A2 | A3 | A4 | A5 | A6 | A7.
Definition G := A1 :: A2 :: A3 :: A4 :: A5 :: A6 :: A7 :: nil.
```

As is given in [20], [21], [22] and [8], epistemic logic cannot be represented directly in Coq's logical framework. Reason is knowledge generalization rule for epistemic logic. By using deduction rules from propositional logic in combination with knowledge generalization rule we can infer that "if φ holds then agent A_i knows φ". But, the knowledge generalization rule does not mean this. What it means is that if φ is true in every world that an agent considers to be a possible world, then the agent must know φ at every *possible world*.[8] So, if we want to introduce epistemic logic in Coq we need to present predicate calculus in Hilbert-style as a metatheory and epistemic logic as a theory. To do so we first introduce a type *proposition* as an inductive type with four constructors, namely the implication *Imp*, the quantifier *Forall* and two operators for modalities *K* and *C* for knowledge and common knowledge in a group of agents (then we abreviate connector *Imp* and quantifier *Forall* with "==>", "\-/"):

```
Inductive proposition : Set :=
  | Imp : proposition -> proposition -> proposition
  | Forall : forall A : Set, (A -> proposition) -> proposition
  | K : Agents -> proposition -> proposition
  | C : list Agents -> proposition -> proposition.

Infix "==>" := Imp (right associativity, at level 85).
Notation "\-/ p" := (Forall _ p) (at level 70, right associativity).
```

To define quantifier *Exists* as well as the connectors *AND*, *OR*, *TRUE*, *FALSE* and *NOT* we use the above defined connector ==> and the quantifier \-/ (we abbreviate connectors *AND* and *OR* with "&" and "V"):

```
Definition Exist (A : Set) (P : A -> proposition) := \-/ (fun p : proposition => \-
/ (fun a : A => P a ==> p) ==> p).
Definition FALSE := \-/ (fun p : proposition => p).
Definition TRUE := Exist proposition (fun p : proposition => p).
Definition NOT (p : proposition) := p ==> FALSE.
Definition AND (p q : proposition) := \-/ (fun r : proposition => (p ==> q ==>
r) ==> r).
Definition OR (p q : proposition) := \-/ (fun r : proposition => (p ==> r) ==>
(q ==> r) ==> r).

Infix "&" := AND (left associativity, at level 50).
Infix "V" := OR (left associativity, at level 50).
```

Common knowledge logic requires introducing a modality *E* for shared knowledge and we have done it recursively by using the *Fixpoint* definition [12, 27], [13, 42], [29, 115], [18, 43]:

```
Fixpoint E (G : list Agents) (p : proposition) {struct G} : proposition := match
G with
      | nil => TRUE
      | i :: G' => K i p & E G' p
    end.
```

Finally, we can introduce the predicate *theorem* in the set *proposition*, which tells which propositions are theorems. What we need are three Hilbert-style axioms of propositional logic, Modus Ponens, K-axiom, T-axiom, Knowledge generalization rule as well as one axiom and one rule for common knowledge. We abreviate *theorem* with "|-":

```
Inductive theorem : proposition -> Prop :=
| Hilbert_K : forall p q : proposition, theorem (p ==> q ==> p)
| Hilbert_S : forall p q r : proposition, theorem ((p ==> q ==> r) ==> (p ==>
q) ==> p ==> r)
| Classic_NOTNOT : forall p : proposition, theorem (NOT (NOT p ==> p))
| MP : forall p q : proposition, theorem (p ==> q) -> theorem p -> theorem q
| K_K : forall (a : Agents) (p q : proposition), theorem (K a p ==> K a (p ==>
q) ==> K a q)
| K_T : forall (a : Agents) (p : proposition), theorem (K a p ==> p)
| K_rule : forall (a : Agents) (p : proposition), theorem p -> theorem (K a p)
| Fixpoint_C : forall (G : list Agents) (p : proposition), theorem (C G p ==> p
& E G (C G p))
| Least_Fixpoint_C : forall (G : list Agents) (p q : proposition), theorem (q
==> p & E G q) -> theorem (q ==> C G p).

Notation "|- p" := (theorem p) (at level 80).
```

3 Knowledge games in Coq

The multiagent systems that we consider in this paper are knowledge games as "card games where a number of cards are distributed over a number of players, and where moves consist of information exchange" [35, vii]. In other words, knowledge games are card games where players have limited information about other players' cards and gradually gain information about the states by observing the actions of other players.[9] Under *state* we understand the distribution of cards among players (and maybe in piles on the table), as well as a player's knowledge acquired during the play as well as knowledges about distribution of cards, knowledges about knowledges of other players, higher-order knowledges, ..., right to common knowledges.[10]

An example of knowledge games is the game of Cluedo which was researched from the aspect

[8] More about concept of possible worlds can be found for example in [11], [41, 270], [7, 15], [43, 126], [28, 398] or [42, 14].

[9] In this paper we use the terms "agent" and "player" depending on the context, although under these terms they mean the same thing: members of a multiagent system.

[10] If needed, we can introduce special players with no knowledge and no possibility of taking action. Thus, depending on the game, player may be *table* as a player who possesses a subset of cards [25, 242], [24, 19]. For the other players these cards can be known, partly known or completely unknown. In such a way, without loss of generality, we can assume that all the cards are dealt.

of reasoning about knowledge in several publications using different formalisms. In [4] and [5], it was investigated from the situation calculus point of view, in [10] using temporal epistemic logic, in [24] and [25] using epistemic logic $S5_n$ with a simply introduced special temporal parameter and in [35], [36], [38], [39] using dynamic epistemic logic. This paper is the first attempt to consider the game of Cluedo using a Coq proof assistant.

In the game Cluedo the players gather information about a murder. The players must determine who has done it, where and with what weapon. The suspects, the possible vehicle and the location of the crime are displayed on the cards. There are nine possible rooms in the mansion, six suspects and nine possible murder weapons. The deck of cards contains one card for each of the rooms, suspects and weapons.[11] So, we have twenty four cards. One card representing a room, one a suspect and one a weapon is separated from the deck of cards and put on the table upside down so that none of the players see which cards they are. This triplet of cards represents a real room, murderer and a weapon. The remaining cards are mixed together and are equally dealt out amongst the players. There are seven players so every player gets three cards. The players make suggestions about which crime was committed in what room, by what suspect and with what weapon. If a player left of the player who has voiced his suggestion has one of the suggested cards, then he shows a card to the player who has given the suggestion in a way that the other players don't know which card it is. He privately shows always only one card irrespective of whether he has one, two or all three cards. If the player who is left of the player who gives the suggestion does not have in his hand any of the suggested cards, then he announces this and the following player does the same: privately shows one of the suggested cards or announces that he does not have any. This continues until a player does show a card privately or the round is not finished, and none of the players show any cards. In the following round the player who is to the right of the player who gave the suggestion gives a new suggestion and so on. The players apply their knowledge about the cards and all other knowledge which they get about the knowledge of the other players in order to discover the room, the murderer and the weapon. When a player solves the mystery he proclaims it and that player is the winner. The player who offers the suggestion is not allowed to suggest any of the cards from the triplet which they hold in their hand. Also, the player may not suggest the same cards twice in one game, because by doing so they stop the other players getting any new information. A player can repeat a suggestion only if he deduces which cards are on the table.

The cards in the deck we introduce in Coq as enumerated inductive type *Cards*:

```
Inductive Cards : Set := Hall | Dining_Room | Kitchen | Patio | Observatory
| Theater | Living_Room | Spa | Guest_House | Scarlet | Green | Mustard |
White | Plum | Peacock | Knife | Candlestick | Pistol | Poison | Trophy |
Rope | Bat | Axe | Dumbbell.
```

The fact that an agent holds in his hands some card is an atom as is the fact that cards representing the murder are on the table. To introduce these atoms, first we have to declare two predicate types [6, 74]. First is type *Hold* where term *Hold C A* means that agent A holds card C in his hand. On the other hand, for the cards which represent murder we declare as predicate type *Murder* where term *Murder C* means that card C is one of the card on table:

```
Parameter Hold : Cards -> Agents -> proposition.
Parameter Murder : Cards -> proposition.
```

Now we can set up atoms as instances of predicates *Hold* and *Murder*. Without losing generality we can assume that the crime was committed in *Hall*, by *Kasandra Scarlet* and with *Knife*:

```
Hypothesis M_R : |- Murder Hall.
Hypothesis M_S : |- Murder Scarlet.
Hypothesis M_W : |- Murder Knife.
```

The rest of cards are distributed among the players and we can assume distribution by setting up atoms in this way:[12]

```
Hypothesis H_A1_1 : |- Hold Dining_Room A1.
Hypothesis H_A1_2 : |- Hold Green A1.
Hypothesis H_A1_3 : |- Hold Candlestick A1.
...
```

4 Axioms of knowledge games

We axiomatize the initial state of the Cluedo card game in accordance with axiomatisation of card

[11] There is more version of Cluedo game. In this paper we ignore some aspects and some rules of the game as we are interested in representing and reasoning about knowledge. The original and complete equipment and rules of the game Cluedo is shown in Hasbro web site: http://www.hasbro.com/.

[12] Here we present only the atoms that are related to the three cards that holding a player A_1.

games given in [34] and [35]. In the aforementioned publications it has been proved that the given axiomatization describes a model that in a technical sense all other models are bisimilar to. Axiomatization is given for the initial state of card games for any number of players and cards and with these properties: players see their own (and only their own) cards, all cards are different, each player has a known number of cards and all players know which cards are in the game. These properties match the properties of the game Cluedo with one exception: in the game Cluedo some cards are on the table. As we explained in footnote 10 in Section 3, it is possible to introduce to the table a special player with no knowledge and no possibility of taking action. This is, for example, done in [27], [35] and [37]. But in this paper based on the theory of types it is complicated to introduce restrictions to the *table* as one of the constructor of type *Agents* in accordance with its properties. This particularly applies to the axiomatization of epistemic logic introduced in Section 1. It seems easier to upgrade the axiomatization of card games and this is what we do. We introduce axiomatization in our system as follows (axioms that are added to the axiomatic system from [34] and [35] are marked with an asterisk):

Players see their own cards: Axiom See : forall (A : Agents) (C : Cards), |- Hold C A -> |- K A (Hold C A).

Players only see their own cards (don't see cards of others): Axiom DontSee : forall (A : Agents) (C : Cards), |- NOT (Hold C A) -> |- K A (NOT (Hold C A)).

All cards are different 1 (one card can be held by at most one player): Axiom AtMost_1 : forall (Ai Aj : Agents) (C : Cards), Ai <> Aj -> |- NOT (Hold C Ai & Hold C Aj).

All cards are different 2 (one card can be held by one player or can be on the table):* Axiom AtMost_2 : forall (A : Agents) (C : Cards), |- NOT (Hold C A & Murder C).

Each player has (at least) three cards: Axiom AtLeast_1 : forall A : Agents, exists C1 : Cards, exists C2 : Cards, exists C3 : Cards, C1 <> C2 /\ C1 <> C3 -> |- Hold C1 A & Hold C2 A & Hold C3 A.

On table are (at least) three cards:* Axiom AtLeast_2 : exists C1 : Cards, exists C2 : Cards, exists C3 : Cards, |- Murder C1 & Murder C2 & Murder C3.

Players don't know the cards of others: Axiom DontKnowThat_1 : forall (Ai Aj : Agents) (C : Cards), Ai <> Aj -> |- NOT (K Ai (Hold C Aj)).

Players don't know the cards on table:* Axiom DontKnowThat_2 : forall (A : Agents) (C : Cards), |- NOT (K A (Murder C)).

Players can imagine others to hold other cards: Axiom DontKnowNot_1 : forall (Ai Aj : Agents) (C : Cards), Ai <> Aj -> |- NOT (Hold C Ai) -> |- NOT (K Ai (NOT (Hold C Aj))).

Players can imagine cards on table:* Axiom DontKnowNot_2 : forall (A : Agents) (C : Cards), |- NOT (Hold C A) -> |- NOT (K A (NOT (Murder C))).

5 Dynamic epistemic logic and epistemic actions in Coq

Unlike epistemic logic, which is about the knowledge of the world, dynamic logic is about the changes of the world that is about *actions* which change the world. In this paper under "actions" we mean *epistemic actions* which are performed by agents in order to change their information states [3, 2].[13] Moreover, in [17, 440] are defined *purely epistemic actions* as actions which not produce change in the physical world, but only in the agent's mental state. Dynamic logic in combination with epistemic logic can express epistemic consequences of epistemic actions in the form of new agent's knowledges about the world as well as about each other's knowledges. In order to get dynamic logic of common knowledge, in [39, 72] is common knowledge logic extended with a new operator "[]" where [φ] is a new modality and where formula [φ]ψ stands for "after every announcement of φ, it holds that ψ". So, "announcement of φ" is taken as an epistemic action. In this paper we also introduce operator "[]" but the term [a]φ refers to "after every execution of action *a*, it holds that φ". Generally, in dynamic logic there are as many modalities as there are actions.

As stated in [15] and [16], if we consider systems with perfect recall or no learning then knowledge and time do interact.[14] One of the axioms which connect knowledge and time informally reads as "If a proposition is known to be always true, then it is always known to be true.". This axiom is known as the *interchange principle* [30, 231], [31, 167]) or as KT1 axiom from linear temporal epistemic logic [11, 308], [15, 681], [33, 104]). In our context we can formally write this axiom as $K_A[a]\varphi \rightarrow [a]K_A\varphi$.[15]

In [23], the only existing paper in which dynamic logic of common knowledge using Coq

[13] Unlike in some other publications where is using term "event" (for example in [1] and [2]) we use the term "action".

[14] Informally, a *perfect recall* system assumes that an agent remembers the complete history of state transitions in the past. Formal definition of perfect recall can be found in [15] where is also explained why *no learning* is the dual notion to *perfect recall*.

[15] There are other axioms that connect knowledge and time, but as pointed out in [31], epistemic agents may have different powers of observation and reasoning. According to the same reference, "General dynamic-epistemic logic has no significant interaction axioms for knowledge and action. If such axioms hold, this is due to special features of agents."

is formalised and the first paper that combines the two aforementioned logics using proof assistant, the notion of epistemic actions is not sufficiently elaborated so that it could be used in more general cases of multiagent systems. Only two specifically epistemic actions are considered in relation to a concrete problem (muddy children puzzle) in which in fact only two epistemic actions can happen. They are listed with the assumption that it is quite clear how these actions actually produce new knowledges of agents. Therefore, if we follow this, for each of the possible actions in the system it is necessary to introduce a special instance of the interchange principle as in [23]. Furthermore, from Coq's point of view, the actions are not in any way "grouped" with the aim to define the type of "actions" where each action belonged. In contrast, in this paper we introduce a special type (from the perspective of Coq) which will belong to all epistemic actions which may occur in some multiagent system, and only to them. But epistemic actions in different multiagent systems in general can be very different with respect to the rules of actions established in that system. Because of this, it is necessary to focus on the multiagent system where we know what these rules are. So, in this paper we focus on the game Cluedo described in Section 3. After the agents look at their cards, in the game "Cluedo" three epistemic actions may be conducted:

1. An agent publicly makes a suggestion that some three cards represent the murder;
2. An agent privately shows one of the suggested cards to an another agent;
3. An agent publicly announces that he does not have any of the three suggested cards.

At first glance it seems that among these actions there is too great a difference in terms of their structure and the changes they cause in the system. In fact, all of these actions can be informally reduced to the following structure: *information is given → knowledge of agents is upgraded and some common knowledge is built into the system*. Furthermore, the given information has the following attributes: *agent A_i who gave the information*, *agent A_j who is informed*, *publicity of the information PI* and *information content IC*. *Publicity of the information* means that the information is given as *public* or *private* and we introduce two possible values: *Pri* and *Pub*. Any public information automatically becomes *common knowledge*, and information that is given to an agent in *private* builds on its knowledge of its *contents*.[16] Since actions 1 and 3 can be divided into three independent actions, the *information content* of these actions is "*an agent has not some card*" while *information content* of action 2 is "*an agent has some card*". Generally, the *information content* is always related to *have/do not have some card*.

In Coq, first we have to introduce the type of *publicity* as enumerated inductive type *PI*:

```
Inductive PI : Set := Pri | Pub.
```

Based on the informal description given above, all of the epistemic actions in game Cluedo can be included in a unique record type [6, 145] *Action* with four functions in accordance with the attributes given above. In addition, to indicate that an action is really executed in some state of system, we introduce a predicate *ex_act*:

```
Record Action : Set := act {Ai : Agents; Aj : Agents; pi : PI; c : Cards}.
Parameter ex_act : Action -> proposition.
```

Now we can consider all the actions that agents can execute in the game of Cluedo as a unique type *Action* which can have different instances depending on instances of their attributes. Also, now we have all prerequisites to declare predicate *aft_ex_act* which instance *aft_ex_act a p* stands for: "after every execution of action *a*, it holds that *p*" that is for "*[a] p*":[17]

```
Parameter aft_ex_act : Action -> proposition -> proposition.
```

Now we can add a general form of interchange principle as a new axiom as follows:

```
Axiom IP : forall (A : Agents) (a : Action) (p : proposition), |- (K A
(aft_ex_act a p) ==> aft_ex_act a (K A p)).
```

We introduce axioms which describe how epistemic actions, depending on their attributes, produce new knowledge in the system:

If A_i informs (in public) A_j that he haven't card c then it become a theorem and it become common knowledge:

```
Axiom Not_Hold_Pub :
forall (Ai Aj : Agents) (c : Cards), |- ex_act (act Ai Aj Pub c) -> |- (NOT
(Hold c Ai) ==> (C G (NOT (Hold c Ai))).
```

If A_i informs (in private) A_j that he have card c then A_j knows that:

```
Axiom Hold_Pri :
forall (Ai Aj : Agents) (c : Cards), |- ex_act (act Ai Aj Pri c) -> |- (K Aj (Hold
c Ai)).
```

[16] It is clear that any information given to the private also builds system with some common knowledge. For example, if agent A_i privately show one card to agent A_j, then become common knowledge that A_j know that A_i have "some" card.

[17] In contrast to [23] where each action is separately declared as type *proposition -> proposition*.

6 Goal and implementation

The game ends when at least one of the players knows the room where the murder took place, the murderer and the weapon. The end of the game will be in Coq presented as a *Goal* that must be proved in order for the game to finish:

```
Goal exists A : Agents, |-K A (Murder Hall) & K A (Murder Scarlet) & K A
(Murder Knife).
```

Once the goal is set, it is possible, with regards to the actions carried out during the game, using the appropriate axioms, to calculate what knowledge players have after specific actions. For example, let us assume that the first actions executed in the game are that A_1 makes the suggestion that *Kitchen*, *Plum* and *Pistol* represent the murder cards and that A_2 privately shows *Kitchen* card to A_1. If we code these actions in a sequence of instances of predicates, then we can apply the appropriate sequence of Coq tactics, for example:

```
repeat match goal with [h : |- Hold ?C ?A |- _ ] => apply See in h end.
assert (a : |- ex_act (act A1 A2 Pub Kitchen) /\ |- ex_act (act A1 A2 Pub
Plum) /\ |- ex_act (act A1 A2 Pub Pistol)). Focus 2.
destruct a as [a1 a2]. destruct a2 as [a2 a3].
assert (a : |- ex_act (act A2 A1 Pri Kitchen)).Focus 2.
repeat match goal with [h : |- ex_act (act ?Ai ?Aj ?pi ?C) |- _ ] =>
(apply Not_Hold_Pub in h || apply Hold_Pri in h) end.
repeat match goal with [h : |- (?P ==> ?Q) |- exists A : Agents, |-K A
(Murder Hall) & K A (Murder Scarlet) & K A (Murder Knife)] =>
apply MP with (q:=Q) in h end. Focus 1.
```

By applying such tactics we get knowledge of players and common knowledge among the whole group of players after the first two actions. Then move on to the next action and so on. After each action, it is necessary to check whether it is possible to prove the goal and whether some of the players know which cards are on the table.

7 Conclusion

In this article we have shown how epistemic actions can be modelled in dynamic common knowledge logic using Coq - a formal proof management system. All this is in order to express the epistemic consequences of epistemic actions of agents in the form of agents' knowledge about the state of the system and higher-order agents' knowledge, up to common knowledge. We show how our approach enables us to reason about knowledge games as an example of knowledge-based multiagent system.

The most interesting question for further research is: whether this approach can serve for reasoning about optimal strategies for players in multiagent competitive games, whether this approach can be extended in order to incorporate distributed knowledge among the agents and for more efficient dealing with common knowledge.

References

[1] Aumann, R. J.: Interactive epistemology I: Knowledge, International Journal of Game Theory 28, 1999, pp. 263–300.

[2] Aumann, R. J.: Interactive epistemology II: Probability, International Journal of Game Theory 28, 1999, pp. 301–314.

[3] Baltag, A.: A logic of epistemic actions, (in electronic) Proceedings of the ESSLLI 1999 workshop on Foundations and Applications of Collective Agent-Based Systems, 1999.

[4] Bart, B.: Representations of and strategies for static information, noncooperative games with imperfect information, M.S. thesis, Simon Fraser University, Vancouver, Canada, 2000.

[5] Bart, B., Delgrande, J., Schulte, O.: Knowledge and planning in an action-based multi-agent framework: A case study, Advances in Artificial Intelligence, Springer Lecture Notes in AI 2056, 2001, pp. 121-130.

[6] Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development, Springer-Verlag, Berlin and Heidelberg, Germany, 2004.

[7] Davis, E., Morgenstern, L.: Epistemic Logic and its Applications: Tutorial Notes, International Joint Conferences on Artificial Intelligence, 1993.

[8] de Wind, P.: Modal logic in COQ, M.S. thesis, Vrije Universiteit Amsterdam, 2002.

[9] Delahaye, D.: A Tactic Language for the System Coq, Proceedings of Logic for Programming and Automated Reasoning, 2000, pp. 85-95.

[10] Dixon, C.: Specifying and Verifying the Game Cluedo using Temporal Logics of Knowledge, Technical Report number ULCS-04-003, University of Liverpool, 2004.

[11] Fagin, R., Halpern, J. Y., Moses, Y., Vardi, M. Y.: Reasoning about Knowledge, MIT Press, Cambridge, Massachusetts, 2003.

[12] Giménez, E.: A tutorial on recursive types in coq, Technical report, The French national institute for research in computer science and control (INRIA), 1998.

[13] Gimenez, E., Castéran, P.: A Tutorial on [Co-]Inductive Types in Coq, available at http://www.labri.fr/perso/casteran/RecTutorial.pdf, January, 31st 2007.

[14] Halpern, J. Y.: Reasoning About Knowledge: A Survey, Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 4: Epistemic and Temporal Reasoning, 1995.

[15] Halpern, J. Y., van der Meyden, R., Vardi, M. Y.: Complete Axiomatizations for Reasoning about Knowledge and Time, SIAM Journal on Computing 33:2, 2004, pp. 674-703.
[16] Halpern, J. Y., Vardi, M. Y.: The Complexity of Reasoning about Knowledge and Time: Synchronous Systems, Technical Report RJ 6097, IBM, 1988.
[17] Herzig, A.: Review of "Dynamic Epistemic Logic" by Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi (Springer Verlag, Synthese Library No. 337, 2007), Studia Logica 89, 2008, pp. 439-443.
[18] Huet, G., Kahn, G., Paulin-Mohring, C.: The Coq Proof Assistant - A Tutorial, available at http://coq.inria.fr/V8.2pl1/files/Tutorial.pdf, February, 27st 2009.
[19] Leroy, X., Doligez, D., Garrigue, J., R´emy, D., Vouillon, J.: The Objective Caml system, available at http://caml.inria.fr/distrib/ocaml-3.11/ocaml-3.11-refman.pdf, April, 15st 2010.
[20] Lescanne, P.: Epistemic logic in higher order logic: an experiment with COQ, Technical Report RR2001-12, LIP-ENS de Lyon, 2001.
[21] Lescanne, P.: Mechanizing common knowledge logic using Coq, Annals of Mathematics and Artificial Intelligence, Volume 48, Numbers 1-2, 2006, pp. 15-43.
[22] Lescanne, P.: Mechanizing epistemic logic with Coq, Research Report RR2004-27, LIP-ENS de Lyon, 2004.
[23] Lescanne, P., Puisségur, J.: Dynamic Logic of Common Knowledge in a Proof Assistant, Research Report RR2007-50, LIP-ENS de Lyon, 2007.
[24] Maliković, M.: Reasoning about multiagent systems by using OTTER system for automatic theorem proving on the example of card games, M.S. thesis, Faculty of Organization and Informatics, Varaždin, Croatia, 2006.
[25] Maliković, M.: Reasoning about the Game "Clue" by using OTTER, Journal of Information and Organizational Sciences, Vol. 30, No. 2, 2006, pp. 241-249.
[26] Meyer, J.-J.: Modal Logics for Intelligent Agents, available at http://www.cs.uu.nl/docs/vakken/iag/Handbook.modal.pdf, January, 7th 2006.
[27] Neufeld, E.: Clue as a Testbed for Automated Theorem Proving, Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence, 2002, pp. 69-78.
[28] Shoham, Y., Leyton-Brown, K.: Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations, Cambridge University Press, 2008.
[29] The Coq Development Team: The Coq Proof Assistant Reference Manual Version 8.2, available at http://coq.inria.fr/refman/, February, 27st 2009.
[30] van Benthem, J.: Games in Dynamic Epistemic Logic, Bulletin of Economic Research, 53:4, 2001, pp. 219-248.
[31] van Benthem, J., Fenrong Liu.: Diversity of Logical Agents in Games, Philosophia Scientiæ, 8 (2), 2004, pp. 165–181.
[32] van der Hoek, W., Meyer, J.-J.: A complete epistemic logic for multiple agents: combining distributed and common knowledge. In P. Mongin, M. Bacharach, L. Gerard-Valet, H. Shin (eds.), Epistemic Logic and the Theory of Games and Decisions, pp. 35-68. Kluwer, Dordrecht, 1997.
[33] van der Meyden, R., Wong, K.: Complete Axiomatizations for Reasoning about Knowledge and Branching Time, Studia Logica, Volume 75, Number 1, 2003, pp. 93-123.
[34] van Ditmarsch, H. P.: Axioms for card games, available at http://dare.uva.nl/document/1238, February, 21st 2006.
[35] van Ditmarsch, H. P.: Knowledge games, Ph. D. Thesis, Grafimedia Groningen University, 2000 (ILLC Dissertation Series 2000-06).
[36] van Ditmarsch, H. P.: The description of game actions in Cluedo, Game Theory and Applications, Volume VIII, 2002, pp. 1-28.
[37] van Ditmarsch, H. P.: The logic of knowledge games: showing a card, Proceedings of the Netherlands/Belgium Conference on Artificial Intelligence (BNAIC 99), 1999, pp. 35-42.
[38] van Ditmarsch, H. P., Kooi, B. P.: The Secret of My Success, Synthese, 2006.
[39] van Ditmarsch, H. P., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic, Synthese Library volume 337, Springer, 2007.
[40] Weiss, G. (editor).: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press, Cambridge, 1999.
[41] Wooldridge, M.: An Introduction to Multiagent Systems, John Wiley & Sons, 2002.
[42] Wooldridge, M.: The logical modelling of computational multi-agent systems, Ph. D. Thesis, University of Manchester, 1992.
[43] Wooldridge, M., Jennings, N. R.: Intelligent Agents: Theory and Practice, Knowledge Engineering Review, Vol. 10, 1995, pp. 115-152.