Reasoning about Epistemic Actions and Knowledge in Multi-agent Systems using Coq

Marko Maliković[1] and Mirko Čubrilo[2]

1. Faculty of Humanities and Social Sciences in Rijeka, University of Rijeka, Croatia

2. Faculty of Organization and Informatics in Varaždin, University of Zagreb, Croatia

Abstract: In this paper we outline a formal system for reasoning about agents' knowledge in knowledge games - a special type of multi-agent system. Knowledge games are card games where the agents' actions involve an exchange of information with other agents in the game. Our system is modeled using Coq - a formal proof management system. To the best of our knowledge, there are no papers in which knowledge games are considered using a Coq proof assistant. We use the dynamic logic of common knowledge, where we particularly focus on the epistemic consequences of epistemic actions carried out by agents. We observe the changes in the system that result from such actions. Those changes that can occur in such a system that are of interest to us take the form of agents' knowledge about the state of the system, knowledge about other agents' knowledge, higher-order agents' knowledge and so on, up to common knowledge. Besides an axiomatic of epistemic logic, we use a known axiomatization of card games that is extended with some new axioms that are required for our approach. Due to a deficit in implementations grounded in theory that enable players to compute their knowledge in any state of the game, we show how our approach can be used for these purposes.

Key words: Multi-agent systems, Knowledge games, Dynamic logic of common knowledge, Epistemic actions, Coq.

1. Introduction

"Multi-agent systems are those systems that include multiple autonomous entities with either diverging information or diverging interests, or both." [1, pp. xvii]. The multi-agent 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" [2, pp. 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.[1] Under *state* of the system 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 knowledge about distribution of cards, knowledge about knowledge of other players, higher-order knowledge, ..., right to common knowledge.[2]

In this paper we reason about knowledge in multi-agent systems 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 [3] and

Corresponding author: Marko Maliković, Ph.D., research fields: Formal systems, Multiagent Systems, Reasoning about knowledge, Modal logic, Automated reasoning, Inteligent systems. E-mail: marko.malikovic@ffri.hr

Mirko Čubrilo, Ph.D., research fields: Symbolic logic, Logic programming, Automated theorem proving, Formal methods. E-mail: mirko.cubrilo@foi.hr

[1] 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 multi-agent system.

[2] The concept of common knowledge will be defined in Section 4.

[4]. 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 [4, pp. 40] 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, Haskell or Scheme. Coq is written in the OCaml language [5], with a bit of C. As a proof development system, Coq provides interactive proof methods and a tactic language [6], [3, pp. 61], [4, pp. 221] for letting the user define its own proof methods.

The paper is organized as follows: Section 2 informally outlines an example of a knowledge game (as a special type of knowledge based multi-agent system). In Section 3, in order to study knowledge games, language and axiomatization of multi-modal propositional epistemic logic $S5_n$ is given while in Section 4 this logic is extended to common knowledge logic. In Section 5 common knowledge logic is formalized using Coq. In Section 6 knowledge games are formalized using the example of the game of Cluedo, in a way that new formalization extends an already known formalization. In Section 7 the possibility of formalization of epistemic actions using Coq within dynamic epistemic logic is discussed while in Section 8 a possible implementation of the system (in the form of using Coq's tactics) is shown. In Section 9 conclusions are drawn.

2. Example of knowledge game

An example of knowledge games is the game of Cluedo which was researched from the aspect of reasoning about knowledge in several publications using different formalisms. In [7] and [8], it was investigated from the situation calculus point of view, in [9] using temporal epistemic logic, in [10] and [11] using epistemic logic $S5_n$ with a simply introduced special temporal parameter and in [2], [12], [13], [14] using dynamic epistemic logic. This paper has been improved with respect to [15] and these two papers are a 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.[3]

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

[3] 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/.

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.

3. Multi-modal propositional epistemic logic

In order to reason about agents' knowledge in a multi-agent system, propositional classical logic is insufficient. In propositional logic, a formula can be true or false. But in the area of multi-agent systems in which multiple agents are in different mutual interactions, it is necessary to introduce 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 [16, pp. 267], [17, pp. 335], [1, pp. 409], [18]. 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* [19], [20], [21], [22], [18], [23], [16, pp. 278]. To study multi-agent systems we use multi-modal propositional epistemic logic $S5_n$ [20, pp. 59]. In accordance with [20, pp. 31], language of $S5_n$ logic is the language of classical propositional logic extended by modal operators for knowledge $K_1,\ldots,K_n$, 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 (atoms describe some state of affairs in the "actual world")
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-axiom):

   $(K_i\varphi \wedge K_i(\varphi \rightarrow \psi)) \rightarrow K_i\psi$

3. Knowledge axiom (T-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) for epistemic logic. RN acronym comes from the *Rule of Necessitation* in modal logic [24, pp. 25]:

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

RN: "From φ infer $K_i\varphi$ "

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*. More about concept of possible worlds can be found for example in [20], [16, pp. 270], [19, pp. 15], [25, pp. 126], [1, pp. 398] or [23, pp. 14]. Due to the knowledge generalization rule, epistemic logic cannot be directly represented in Coq's logical framework, as we will see in Section 5.

4. Common knowledge logic

Multi-modal 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 $E_A\varphi$. 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.

The second operator is the operator of *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. For example, if agent A_1 knows that agent A_3 has either card C_1 or C_2 and agent A_2 knows that agent A_3 does not have card C_1, then together agents A_1 and A_2 have distributed knowledge of the fact that agent A_3 has card C_2, although neither A_1 nor A_2 individually has this knowledge. The concept of distributed knowledge will not be used in the paper, but it's formal description may be found in [20], [16] and [23].

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, everyone knows that 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 [20, pp. 433] and [1, pp. 406] is:

Definition (*Common knowledge*): $C_A\varphi$ iff $E_A^{\,k}\varphi$ for $k = 1, 2, \ldots$ where is $E_A^{\,k}\varphi = E_A E_A^{\,k-1}\varphi$.

The multi-modal epistemic logic with a modal operator which describes common knowledge condition we call *common knowledge logic* [22], [26].

5. Common knowledge 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:

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

Coq version 8.0 or higher has to be run with the *-impredicative-set* option because the definition of *prop* which we introduce below is based on a non predicative notion of *Set*. For details see [4, pp. 123].

We introduce agents as enumerated inductive type *Agents* used to describe finite sets [3, pp. 137]. In the example of the multi-agent system which we considered in Section 2, we have seven agents, so our definition consists of seven constructors:

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

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 multi-agent system is required to be finite and for this reason, we do not want to use the infinite structures to describe finite types.

Also, we will need the concept of a whole group of agents and we define it as list *G*:

```
Definition G := A1 :: A2 :: A3 :: A4 :: A5 :: A6 :: A7 :: nil.
```

As is already mentioned in Section 3, and given in [27], [28], [29] and [30], epistemic logic cannot be directly represented in Coq's logical framework. The reason for this is the knowledge generalization rule for epistemic logic. 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, mostly in accordance with [28], we first introduce a type *prop* (which should not be confused with Coq's type of proposition *Prop*) as an inductive type. This type has 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. Just as the types *prop* and *Prop* should not be confused, also our implication *imp* should not be confused with Coq's implication "->", our quantifier *Forall* should not be confused with Coq's quantifier *forall*, as well as any further quantifiers and connectives, which we introduce below:

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

We will abbreviate connector *Imp* and quantifier *Forall* with "==>", "\-/":

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

Also, in the metatheory that presents the Hilbert-style predicate calculus, we need an existential quantifier, the remaining logical connectives (conjunction, disjunction and negation) and the propositional constants (*True* and *False*). To define them, we use the above defined connector ==> and the quantifier \-/:

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

We abbreviate connectors *AND* and *OR* with "&" and "V"):

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

Of course, these concepts should not be confused with Coq's quantifier *exists*, with connectives *and*, *or* and *not* (abbreviated /\, \/ and ~) and with the constants *True* and *False*.

Common knowledge logic requires introducing a modality *E* for shared knowledge and we have done it recursively by using the *Fixpoint* definition [31, pp. 27], [32, pp. 42], [4, pp. 119], [33, pp. 43]:

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

Finally, we can introduce the predicate *theorem* in the set *prop*, which tells which propositions are theorems. What we need are 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 as outlined below.

```
Inductive theorem : prop -> Prop :=
| Lukas_1 : forall p q : prop, theorem (p ==> q ==> p)
| Lukas_2 : forall p q r : prop, theorem ((p ==> q ==> r) ==> (p ==> q) ==> p ==> r)
| Lukas_3 : forall p q : prop, theorem ((NOT p ==> NOT q) ==> q ==> p)
| MP : forall p q : prop, theorem (p ==> q) -> theorem p -> theorem q
| K_K : forall (a : Agents) (p q : prop), theorem (K a p ==> K a (p ==> q) ==> K a q)
| K_T : forall (a : Agents) (p : prop), theorem (K a p ==> p)
| K_rule : forall (a : Agents) (p : prop), theorem p -> theorem (K a p)
| Fixed_point_C_axiom : forall p : prop, theorem (C G p ==> E G (p & C G p))
| Induction_rule_C : forall p q : prop, theorem (p ==> E G (p & q)) -> theorem (p ==> C G q).
```

We abbreviate *theorem* with "|-":

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

For the Hilbert-style axioms of propositional logic we took three which are given by Łukasiewicz (constructors *Lukas_1*, *Lucas_2* and *Lucas_3*), together with the *Modus ponens* (*MP*) rule. From the axioms of epistemic logic, we exclude the positive and negative introspection because the knowledge of agents about their own knowledge is not important for us in this paper. However, it is clear that the K-axiom and T-axiom are important.

On the basis of the example of the Knowledge generalization rule (*K_rule*) which is introduced in the definition of *theorem*, we can finally explain more clearly why we (in the Coq system) had to raise propositional logic to the level of metatheory. Namely, if we look at the definition of the *theorem*, we see that the Knowledge generalization rule may be applied to a proposition *p* only if it is a theorem. As a result of this application, we gather that $K_a p$ is a theorem for each agent *a* from group *G*. This, however, would not be feasible when the proposition *p* is valid, but is not a theorem. The matter becomes even clearer if we compare the Knowledge generalization rule and the Knowledge axiom (*T-axiom*). Specifically, the Knowledge axiom claims that it is a theorem that "if an agent knows a statement *p* then statement *p* is valid" and it is correct.

The last two constructors (*Fixed_point_C_axiom* and *Induction_rule_C*) in the definition of *theorem* are related to common knowledge. These are actually properties of common knowledge introduced in [34] and studied further in many publications, for example in [20]. In [34], they are given as:

C1. The fixed point axiom: $C_G\varphi \equiv E_G(\varphi \wedge C_G\varphi)$

C2. The induction rule: From $\varphi \rightarrow E_G(\varphi \wedge \psi)$ infer

$$\varphi \rightarrow C_G\psi$$

The fixed point axiom essentially characterizes $C_G\varphi$ as the greatest solution of the following fixed point equation $X \equiv E_G(\varphi \wedge X)$. [34, pp. 37] illustrates how common knowledge can be formally defined as the greatest fixed point. A second property of common knowledge (induction rule) says that if φ is "public" and implies ψ, so whenever φ holds then everybody knows $\varphi \wedge \psi$, then whenever φ holds, ψ is common knowledge.

6. Knowledge games in Coq

6.1. Definitions, declarations and hypotheses

We introduce the formal bases of knowledge games using Coq with the example of the game Cluedo, as described in Section 2. We introduce the cards in the deck 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 [3, pp. 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 -> prop.

Parameter Murder : Cards -> prop.
```

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:

```
Hypothesis A1_1 : |- Hold Dining_Room A1.
Hypothesis A1_2 : |- Hold Green A1.
Hypothesis A1_3 : |- Hold Candlestick A1.
Hypothesis A2_1 : |- Hold Kitchen A2.
Hypothesis A2_2 : |- Hold Mustard A2.
Hypothesis A2_3 : |- Hold Pistol A2.
Hypothesis A3_1 : |- Hold Patio A3.
Hypothesis A3_2 : |- Hold White A3.
Hypothesis A3_3 : |- Hold Poison A3.
Hypothesis A4_1 : |- Hold Observatory A4.
Hypothesis A4_2 : |- Hold Plum A4.
Hypothesis A4_3 : |- Hold Trophy A4.
Hypothesis A5_1 : |- Hold Theater A5.
Hypothesis A5_2 : |- Hold Peacock A5.
Hypothesis A5_3 : |- Hold Rope A5.
Hypothesis A6_1 : |- Hold Living_Room A6.
Hypothesis A6_2 : |- Hold Bat A6.
Hypothesis A6_3 : |- Hold Spa A6.
Hypothesis A7_1 : |- Hold Guest_House A7.
Hypothesis A7_2 : |- Hold Axe A7.
Hypothesis A7_3 : |- Hold Dumbbell A7.
```

6.2. Axioms of knowledge games

We axiomatize the initial state of the Cluedo card game in accordance with axiomatization of card games given in [35] and [2]. 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. It is possible to introduce the table as a special player, which has no knowledge and no possibility of taking action. This is, for example, done in [36], [2] 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. 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 [35] and [2] 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))).
```

7. 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 some other publications (for example in [38] and [39]) the term *event* is used instead of *action*. In this paper under *actions* we mean *epistemic actions* which are performed by agents in order to change their information states [40, pp. 2]. Moreover, in [41, pp. 440] are defined *purely epistemic actions* as actions which not produce change in the physical world, but only in the agent's mental state. In this paper, epistemic actions and purely epistemic actions mean the same but in general this does not have to be the case.

Dynamic logic in combination with epistemic logic can express epistemic consequences of epistemic actions in the form of a new agent's knowledge about the world, as well as about each other's knowledge. In order to get dynamic logic of common knowledge, in [14, pp. 72] is common knowledge logic extended with a new operator "[]" where $[\varphi]$ is a new modality and where formula $[\varphi]\psi$ 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 $[a]\varphi$ 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 [42] and [43], if we consider systems with *perfect recall* or *no learning* then knowledge and time do interact.[4] One of the axioms which connect

[4] 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

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* [44, pp. 231], [45, pp. 167]) or as *KT1* axiom from linear temporal epistemic logic [20, pp. 308], [42, pp. 681], [46, pp. 104]). In our context we can formally write this axiom as $K_A[a]\varphi \rightarrow [a]K_A\varphi$.[5]

[47] is the only existing paper in which the dynamic logic of common knowledge using Coq is formalized, and is the first paper that combines the two aforementioned logics using a proof assistant. Two specific epistemic actions are considered in relation to a concrete problem (*Muddy children puzzle*) where in fact only two epistemic actions can happen. They are listed with the assumption that it is quite clear how these actions actually produce a new knowledge 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 [47]. 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 multi-agent system, and only to them. But epistemic actions in different multi-agent 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 multi-agent system where we know what these rules are. So, in this paper we focus on the game Cluedo described in Section 2. 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*.[6] 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.
```

found in [42] where is also explained why *no learning* is the dual notion to *perfect recall*.

[5] There are other axioms that connect knowledge and time, but as pointed out in [45], 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."

[6] 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.

Based on the informal description given above, all of the epistemic actions in game Cluedo can be included in a unique record type [3, pp. 145] *Action* with four functions in accordance with the attributes given above:

```
Record Action : Set := act {Ai : Agents; Aj : Agents; pi : PI; c : Cards}.
```

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.

In addition, to indicate that an action is really executed in some state of system, we introduce a predicate *ex_act*:

```
Parameter ex_act : Action -> prop.
```

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*":

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

We can add a new axiom which describes the general form of the interchange principle:

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

Finally, 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)).
```

As we mentioned in footnote 6 in this section, we could extend the axiom *Hold_Pri* with the fact that "knowledge which an agent A_i got from an agent A_j in private" implies the common knowledge that "agent A_i knows that agent A_j has some card". However, the consequences of such knowledge are not considered in this paper. Therefore, it is sufficient enough to use a weakened axiom.

8. Implementation

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. Now it is possible, with regards to these actions, to set additional hypotheses in the context:

```
Hypothesis Act1_1 : |- ex_act (act A1 A2 Pub Kitchen).
Hypothesis Act1_2 : |- ex_act (act A1 A2 Pub Plum).
Hypothesis Act1_3 : |- ex_act (act A1 A2 Pub Pistol).
Hypothesis Act2 : |- ex_act (act A2 A1 Pri Kitchen).
```

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).
```

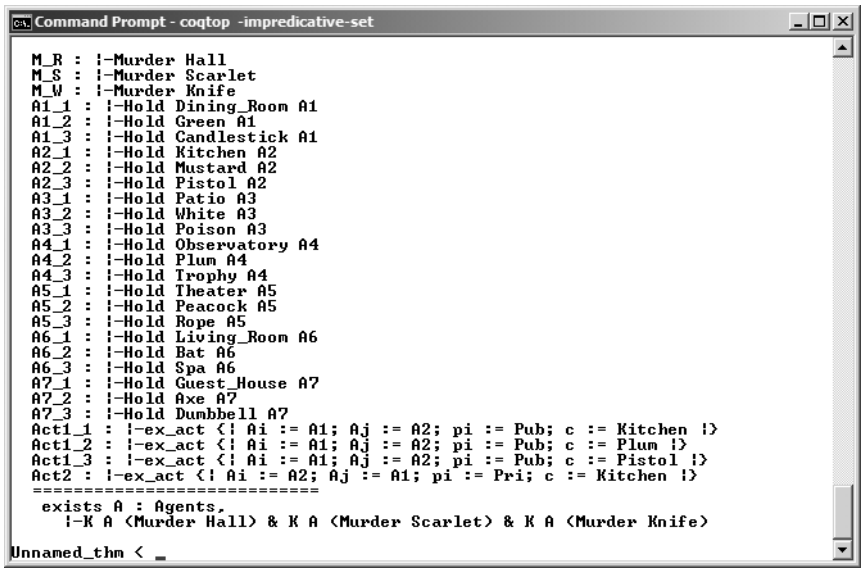

Fig. 1 Coq's interface after a set goal.

Once the goal is set, the Coq's interface is as shown in Figure 1. Coq shows only the hypotheses of distribution of cards among agents, hypotheses about the cards that are on the table, hypotheses about the executed actions and goal.

Now we can, using the appropriate axioms and tactics, calculate what knowledge players have after specific actions:

```
Time

repeat
match goal with
[h : |- Hold ?C ?A |- _ ] =>
apply See in h
end;

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 |- _ ] =>
destruct h; clear h
end;

repeat
match goal with
[h : |- NOT (Hold ?C ?A) |- _ ] =>
apply DontSee in h
end.
```

By applying such tactics we get knowledge of the players' knowledge and common knowledge among the whole group of players after the first two actions. Coq's interface now looks as shown in Figure 2.

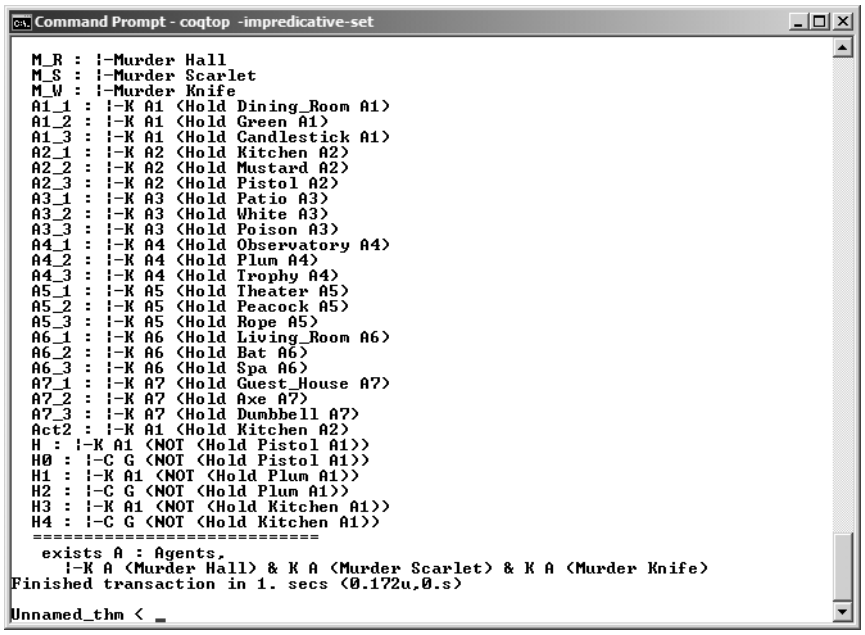

Fig. 2 Coq's interface after the first two agents' actions.

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.

9. Conclusions

In this paper, we have shown how the epistemic consequences of the epistemic actions of agents can be derived in knowledge games - a special type of multi-agent system based on the agents' knowledge. We model epistemic actions in dynamic common knowledge logic using Coq - a formal proof management system. The epistemic consequences of the epistemic actions of agents are presented in the form of agents' knowledge about the state of the system, knowledge about other agents' knowledge, higher-order agents' knowledge, up to common knowledge. We show how our approach can be practically implemented in order to reason about knowledge games.

The most interesting questions for further research are: whether this approach can serve for reasoning about optimal strategies for players in multi-agent competitive games, whether this approach can be extended in order to incorporate distributed knowledge among the agents and to deal with common knowledge more effectively.

References

[1] Y. Shoham, K. Leyton-Brown, Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, 2008.

[2] H.P. van Ditmarsch, Knowledge games. PhD thesis, Grafimedia Groningen University, Groningen, Holland, 2000.

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

[4] The Coq Development Team, The Coq Proof Assistant Reference Manual Version 8.3pl1, http://coq.inria.fr/distrib/current/files/Reference-Manual.pdf, downloaded: March, 25st 2011.

[5] X. Leroy, D. Doligez, J. Garrigue, D. R´emy, J. Vouillon, The Objective Caml system - release 3.12 - Documentation and user's manual, http://caml.inria.fr/distrib/ocaml-3.12/ocaml-3.12-refman.pdf, downloaded: March, 25st 2011.

[6] D. Delahaye, A Tactic Language for the System Coq. In Proceedings of Logic for Programming and Automated Reasoning, pages 85-95, Reunion Island, 2000.

[7] B. Bart, Representations of and strategies for static information, noncooperative games with imperfect

information. M.S. thesis, Simon Fraser University, Vancouver, Canada, 2000.

[8] B. Bart, J. Delgrande, O. Schulte, Knowledge and planning in an action-based multi-agent framework: A case study. In Proceedings of Canadian Conference on AI'2001, pages 121-130, 2001.

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

[10] M. Maliković, 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.

[11] M. Maliković, Reasoning about the Game "Clue" by using OTTER. Journal of Information and Organizational Sciences, 30(2):241-249, 2006.

[12] H.P. van Ditmarsch, The description of game actions in Cluedo. In Petrosjan, L.A; Mazalov, V.V, editors. Game Theory and Applications, Vol. 8. Nova Science Publishers, Hauppauge, NY, 2002.

[13] H.P. van Ditmarsch, B.P. Kooi, The Secret of My Success. Synthese, 151:201-232, 2006.

[14] H.P. van Ditmarsch, W. van der Hoek, B. Kooi. Dynamic Epistemic Logic. Synthese Library volume 337, Springer, 2007.

[15] M. Maliković, M. Čubrilo, Modeling Epistemic Actions in Dynamic Epistemic Logic using Coq. In Proceedings of 21st Central European Conference on Information and Intelligent Systems, pages 3-10, Varaždin, Croatia, 2010.

[16] M. Wooldridge, An Introduction to Multiagent Systems. John Wiley & Sons, 2002.

[17] G. Weiss, editor, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge, 1999.

[18] J.-J. Meyer, Modal Logics for Intelligent Agents, http://www.cs.uu.nl/docs/vakken/iag/Handbook.modal.pdf , downloaded: January, 7th 2011.

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

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

[21] J.Y. Halpern, Reasoning about knowledge: a survey. In Gabbay, D; Hogger, C.J; Robinson, J.A, editors. Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 4. Oxford University Press, 1995.

[22] W. van der Hoek, J.-J. Meyer, A complete epistemic logic for multiple agents: combining distributed and common knowledge. In Mongin, P; Bacharach, M; Gerard-Valet, L; Shin, H, editors. Epistemic Logic and the Theory of Games and Decisions. Kluwer, Dordrecht, 1997.

[23] M. Wooldridge, The logical modelling of computational multi-agent systems. PhD thesis, University of Manchester, Manchester, England, 1992.

[24] M.J. Cresswell, G.E. Hughes, A New Introduction to Modal Logic. Routledge, London and New York, 1996.

[25] M. Wooldridge, N.R. Jennings, Intelligent Agents: Theory and Practice. Knowledge Engineering Review, 10(2):115-152, 1995.

[26] M. Kaneko, Common Knowledge Logic and Game Logic. The Journal of Symbolic Logic, 64(2):685-700, 1999.

[27] P. Lescanne, Epistemic logic in higher order logic: an experiment with COQ. Technical Report RR2001-12, LIP-ENS de Lyon, 2001.

[28] P. Lescanne, Mechanizing common knowledge logic using Coq. Annals of Mathematics and Artificial Intelligence, 48(1-2):15-43, 2006.

[29] P. Lescanne, Mechanizing epistemic logic with Coq. Research Report RR2004-27, LIP-ENS de Lyon, 2004.

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

[31] E. Giménez, A tutorial on recursive types in coq. Technical report 0221, The French national institute for research in computer science and control (INRIA), 1998.

[32] E. Gimenez, P. Castéran, A Tutorial on [Co-]Inductive Types in Coq, http://www.labri.fr/perso/casteran/RecTutorial.pdf, downloaded: March, 25st 2011.

[33] G. Huet, G. Kahn, C. Paulin-Mohring, The Coq Proof Assistant - A Tutorial Version 8.3pl1, http://coq.inria.fr/distrib/current/files/Tutorial.pdf, downloaded: March, 25st 2011.

[34] J.Y. Halpern, Y. Moses, Knowledge and common knowledge in a distributed environment. Journal of the ACM, 37(3):549-587, 1990.

[35] H.P. van Ditmarsch, Axioms for card games. Technical Notes X-2000-02, Institute for Logic, Language and Computation, University of Amsterdam, 2000.

[36] E. Neufeld, Clue as a Testbed for Automated Theorem Proving. In Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence, pages 69-78, Calgary, Canada, 2002.

[37] H.P. van Ditmarsch, The logic of knowledge games: showing a card. In Proceedings of the Netherlands/Belgium Conference on Artificial Intelligence, pages 35-42, Maastricht University, 1999.

[38] R.J. Aumann, Interactive epistemology I: Knowledge. International Journal of Game Theory, 28(3):263-300, 1999.

[39] R.J. Aumann, Interactive epistemology II: Probability. International Journal of Game Theory, 28(3):301-314, 1999.

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

[41] A. Herzig, 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(3):439-443, 2008.

[42] J.Y. Halpern, R. van der Meyden, M.Y. Vardi, Complete Axiomatizations for Reasoning about Knowledge and Time. SIAM Journal on Computing, 33(3):674-703, 2004.

[43] J.Y. Halpern, M.Y. Vardi, The Complexity of Reasoning about Knowledge and Time: Synchronous Systems. Technical Report RJ 6097, IBM, 1988.

[44] J. van Benthem, Games in Dynamic Epistemic Logic. Bulletin of Economic Research, 53(4):219-248, 2001.

[45] J. van Benthem, F. Liu, Diversity of Logical Agents in Games. Philosophia Scientiæ, 8(2):165-181, 2004.

[46] R. van der Meyden, K. Wong, Complete Axiomatizations for Reasoning about Knowledge and Branching Time. Studia Logica, 75(1):93-123, 2003.

[47] P. Lescanne, J. Puisségur, Dynamic Logic of Common Knowledge in a Proof Assistant. Research Report RR2007-50, LIP-ENS de Lyon, 2007.