Finding bounded strategies in games with imperfect information

Théo Losekoot
ENS Rennes
theo.losekoot@ens-rennes.fr

Supervisor : Tristan Charrier
CNRS, Univ Rennes, IRISA
tristan.charrier@irisa.fr

***Abstract*—We consider turn-based games with imperfect information. The problem is to decide whether a specific player has a bounded uniform winning strategy. We show that this problem is solvable by an algorithm running in polynomial space. To do so, we construct a polynomial reduction to the model checking problem of dynamic epistemic logic.**

***Index Terms*—Game theory, Imperfect Information, Bounded strategies, Dynamic Epistemic Logic, Complexity**

I. Introduction

Game theory is an established domain modeling systems where agents interact with each other and have different objectives [1]. An agent is an abstract entity capable of taking decisions autonomously[1], (e.g. humans, coffee machine, dog, ...). In this paper, we focus on a class of games respecting two constraints. Firstly, we consider *turn-based* games, meaning that at each moment, at most one agent may play. Secondly, agents have *imperfect information* on the system, i.e. might not observe the full state of the system. For instance, a round of poker, a round of tarot, a bargaining process are instances of this class of games. Typically, in a tarot round, there are five agents (human players), each owning fifteen cards. A round of tarot decomposes itself into fifteen sub-rounds in which agents play one card each, turn by turn. There is imperfect information because an agent does not know what cards other agents own.

In these situations, there may be a way for an agent to behave in order to be sure to achieve his goals, called *winning strategy*. Moreover, his strategy must be *uniform*, meaning that it relies on his own knowledge about the game. For instance in tarot, an agent cannot tune his strategy depending on the cards of other agents. Furthermore, we search for k-*bounded* winning strategies, i.e. always winning in less than k turns. It is not a limitation for many of those games, since bounded strategies are sufficient in any game where a resource is consumed each turn (e.g. playing a card).

Currently, the complexity of finding a bounded uniform winning strategy in this class of games is unknown[2]. Our contribution consists of showing that this problem can be polynomially reduced to the *model checking problem* of dynamic epistemic logic (DEL) [4]. Since this problem is in PSPACE[3] [6], we deduce that finding uniform bounded winning strategies yields a PSPACE problem.

The paper is divided as follows. In Section II, we define the class of turn-based games with imperfect information, and also define strategies. In Section III, we define dynamic epistemic logic. In Section IV, we provide the reduction mentioned above. Finally, in Section V, we introduce a symbolic representation of games to avoid the combinatorial explosion on the number of possible states in games, and discuss its influence on the complexity.

II. Turn-based games with imperfect information

In this section, we define an outline for turn-based games with imperfect information, called a *game arena*. Then, we add goals in this arena to define *strategies* for agents. The definition is inspired from the founding paper of Alternating-Time Temporal Logic [1].

A. Game arena

Definition II.1. *A* turn-based game arena with imperfect information $G = \langle Agt, AP, St, Val, Act, Succ, Obs\rangle$ *is a tuple where:*

- $Agt = \{a, b, c, \dots\}$ *is the set of* agents*;*
- $AP = \{p, q, r, \dots\}$ *is the set of* atomic propositions*;*
- $St = \{s_1, \dots, s_n\}$ *is the set of* states *with an* initial state $s_{init} \in St$*;*
- $Val : St \to 2^{AP}$ *is the* valuation *function;*
- Act *is the disjoint union of* Act_a *for each agent* a*.* $Act_a = \{\alpha, \beta, \gamma, \dots\}$ *is the set of* actions *for agent* a*;*
- $Succ : St \times Act \to St$ *is the (partial)* successor *function;*
- Obs *is the tuple of* Obs_a *for each agent* a*.* $Obs_a \subseteq AP$ *is the* observation set *for agent* a*.*

Intuitively, atomic propositions characterize the truth value of facts in states. s_{init} is the state that represents the configuration in which the game begins. Val is the function that assigns to each state its valuation (the set of true atomic propositions). Act_a is the set of available actions for agent a. $Succ$ is the function that gives the successor state from a state and an action (when it exists). Obs_a is the set of atomic propositions that agent a can observe (e.g. his cards).

[1] See page 9 of [2] if you wish for a more detailed definition of an agent.

[2] The closest result is a non-deterministic exponential time complexity for games where we search for strategies for a team of players [3].

[3] That is, solvable with an algorithm using polynomial space [5].

We illustrate the definition on a game inspired from [2].

Example II.1. *Robber in the bank*

A vault containing a large amount of money is protected by a binary code, 0 or 1. If someone enters the right code, the vault opens and gives access to what is stored in. If an incorrect code is entered, the alarm switches on. The code is set anew every morning by the guard agent. Some time later in the day, the robber enters the bank and tries to open the vault. However, he does not know the current code.

We represent this example with a graph on Figure 1. Circles represent states with a color code to specify who is playing. Labeled arrows represent the $Succ$ function, and dashed arrows represent the Obs_a sets.

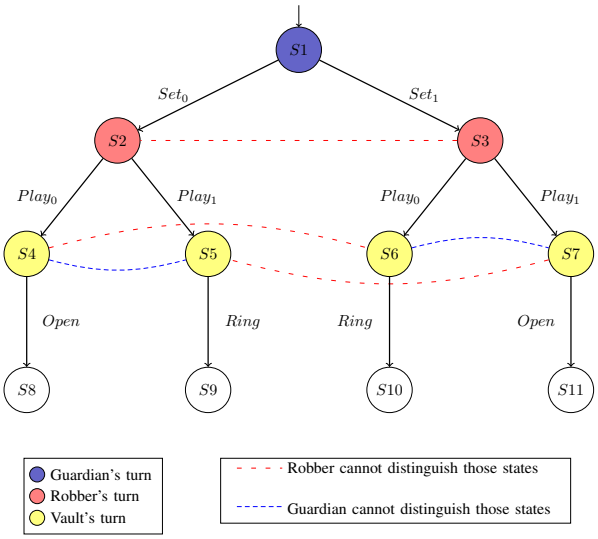

Figure 1: Modeling of the Robber example.

The formalization of this example is in Appendix A.1.

Our definition needs to be refined. Indeed, we capture for instance games where two agents can play at the same turn. That is why we now define some additional constraints.

We define the function $Play_a : St \to 2^{Act}$ that tells which actions agent a can play on a particular state. For every agent a and state s, we have $Play_a(s) = \{\alpha \in Act_a \mid Succ(s, \alpha) \text{ is defined}\}$.

Constraint II.1. *Turn-based constraint*

For every state s, there exists at most one agent a such that $|Play_a(s)| >= 1$ and for all other agents b, $|Play_b(s)| = 0$.

This constraint means only one agent can play at a time, or zero when the game is finished.

We also define the (partial) function $Turn : St \to Agt$ that indicates which player can play. For every state s, $Turn(s) = a$ if and only if $|Play_a(s)| > 0$.

We say that two states s and s' are indistinguishable for agent a, noted $s \sim_a s'$, if their valuation seen by agent a is the same: $Val(s) \cap Obs_a = Val(s') \cap Obs_a$.

Constraint II.2. *Same actions on indistinguishable states*

For all states s, s' and agent a, $s \sim_a s'$ implies $Play_a(s) = Play_a(s')$.

This constraint means that if an agent a does not distinguish two states s and s', then he has the same available actions in those two states.

Note that the previous example respects constraints II.1 and II.2. Now that we have specified the game arena, we focus on victory conditions for agents.

B. Perfect-recall uniform strategies

First, we need to define the notion of history.

Definition II.2. *A* history *$h \in St^+ = \langle h_1, \ldots, h_n \rangle$ is a finite sequence of states corresponding to an execution of the arena, i.e. $h_1 = s_{init}$ and for $i \in \{1, \ldots, n-1\}$, we have $h_{i+1} \in \{Succ(h_i, \alpha) \mid \alpha \in Act\}$. We note $h_{<=i}$ the sub-history composed by the i first elements of h and H the set of all histories for a given arena.*

For example, the history for "The guardian set the code to 1 and then the robber entered 0." is $\langle S1, S3, S6 \rangle$.

Definition II.3. *We say that two histories h and h' are* indistinguishable *for agent a, noted $h \sim_a h'$ if and only if ($|h| = |h'|$ and $h_i \sim_a h'_i$ for $i \in \{1, \ldots, |h|\}$).*

Note that two histories h, h' that end in indistinguishable states for agent a, $h_{|h|} \sim_a h'_{|h'|}$, are not necessarily indistinguishable. For example, in a money game where the agent a can only observe his amount of money, he does not distinguish two states where he has 5 gold coins but he may have obtained them in many different ways (histories) and distinguishes those ways (he *recalls* what happened).

We now consider how agents can behave during the game, and thus define the notion of perfect-recall strategy [1].

Definition II.4. *A* perfect-recall strategy *$\sigma_a : H \to Act_a$ of agent a in a game arena is a partial function that specifies what agent a should play in every possible history: for all history h, if $Turn(h_{|h|}) = a$, then $\sigma_a(h) \in Play_a(h_{|h|})$ and $\sigma_a(h)$ is undefined otherwise.*

In reality, it is pointless to consider strategies that have different moves for two indistinguishable histories. The concept of *uniform strategy* corrects this issue, giving the same output for two histories that the agent cannot distinguish.

Definition II.5. *A strategy is said* uniform *if for all histories h, h', $h \sim_a h'$ implies $\sigma_a(h) = \sigma_a(h')$.*

Definition II.6. *We define the set of all histories $H(\sigma_a)$ generated when agent a sticks to strategy σ_a as follows.*

$H(\sigma_a) = \{h \in H \mid$ for $i \in \{1, \ldots, |h| - 1\}$, if $Turn(h_i) = a$, then $h_{i+1} = Succ(h_i, \sigma_a(h_{<=i}))\}$

Now that we have a way of characterizing how agents behave, we want to lead one of them to victory. Let us say we have, in addition to the game arena, a goal for agent a, $Goal_a \subseteq St$, that is his set of winning states.

Definition II.7. *A strategy σ_a is said* winning *for agent a if all histories $h \in H(\sigma_a)$ contain a winning state. It is said*

k-bounded winning *if we can find a winning state in all histories* $h \in H(\sigma_a)$ *such that* $|h| = k$ *or such that* $|h| < k$ *with no successor for* $h_{|h|}$.

In our example, if we consider $Goal_a = \{S8, S11\}$, there is a winning strategy for agent $Robber$ (play Set_0 in S2 and Set_1 in S3) but no uniform winning strategy.

III. Dynamic Epistemic Logic

Dynamic epistemic logic is a logic to reason about agents' knowledge and their evolution when applying actions. In this section, we first give a definition of epistemic logic, in which we model a given static situation, we then complete it by adding actions.

A. Epistemic Logic

Modern epistemic logic originated from the work of Hintikka [7]. We consider a countable set of atomic propositions AP and a finite set of agents Agt.

Definition III.1. *The* syntax *of epistemic logic (*EL*), whose language is noted* $\mathcal{L}_{EL}$*, is defined by the following grammar:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid K_a(\varphi)$$

with $p \in AP$ *and* $a \in Agt$.

Intuitively, $K_a(\varphi)$ means that agent a *knows* that φ. Boolean connectives $\top, \bot, \wedge, \rightarrow, \leftrightarrow$ are defined as usual and we define the K_a operator's dual $\widehat{K}_a(\varphi) = \neg K_a(\neg\varphi)$.

Example III.1. *Formula* $K_a(K_b(\varphi) \vee K_c(\varphi))$ *means "Agent* a *knows that agent* b *or agent* c *knows* φ*".*

The *semantics* of *EL* is defined on a *pointed Kripke model*.

Definition III.2. *A* Kripke model $\mathcal{M} = (W, (R_a)_{a \in Agt}, V)$ *is a triplet where* W *is the set of possible* worlds*;* R_a *is a binary* epistemic relation *between worlds* ($R_a \subseteq W \times W$)*;* V *is the* valuation *function* ($V : W \to 2^{AP}$).

A pair $(\mathcal{M}, w)$ *is called a* pointed Kripke model *where the world* w *is the* actual world.

Intuitively, a world is the transcription of a state in the epistemic logic domain. We include all worlds agents imagine as possible. The epistemic relations R_a represent the agents' capacities of reasoning, or to be more precise, their inability to distinguish worlds. We have wR_aw' if agent a thinks that w' may be the actual world when in fact the actual world is w.

Definition III.3. *We note* $\mathcal{M}, w \models \varphi$ *for "Formula* φ *is true in the pointed Kripke model* $(\mathcal{M}, w)$*". It is defined by induction on* φ *as follows:*

$\mathcal{M}, w \models p$	iff	$p \in V(w)$;
$\mathcal{M}, w \models \neg\varphi$	iff	$\mathcal{M}, w \not\models \varphi$;
$\mathcal{M}, w \models \varphi \vee \psi$	iff	$\mathcal{M}, w \models \varphi$ or $\mathcal{M}, w \models \psi$;
$\mathcal{M}, w \models K_a(\varphi)$	iff	for all $w' \in W$ such that wR_aw', we have $\mathcal{M}, w' \models \varphi$.

In our case, we limit ourselves to a particular class of Kripke models where epistemic relations are equivalence relations. Called S5, those models are typically used to reason about knowledge [8]. Indeed, since indistinguishable worlds form equivalence classes, the modality $K_a(\varphi)$ stems to checking φ in the whole equivalence class, which is a reasonable semantics for knowledge.

Example III.2. *We cannot represent the full game II.1 with epistemic logic but we can represent the knowledge of agents in one turn. As an example, we can define the pointed Kripke model* $(\mathcal{M}_{S1}, w_1)$ *corresponding to the initial state* $S1$*, and* $(\mathcal{M}_{S3}, w_2)$ *corresponding to the state* $S3$*. The formal representation of these examples are in Appendix B.1.*

We represent these examples with graphs, where nodes represent worlds, arrows represent the epistemic relations, and the valuation is indicated inside the node, next to its name, see Figures 2 and 3.

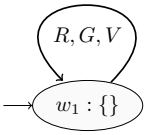

Figure 2: Pointed Kripke model $(\mathcal{M}_{S1}, w_1)$ for state $S1$.

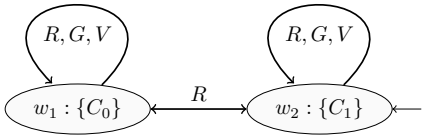

Figure 3: Pointed Kripke model $(\mathcal{M}_{S3}, w_2)$ for state $S3$.

We can see in the example showed in Figure 3 that, as in the $S3$ *state of the previous example, everybody distinguishes the world where the code is 0 from the world where the code is 1, except the* $Robber$ *that cannot.*

B. Dynamic Epistemic Logic

Originally introduced in [4], dynamic epistemic logic (*DEL*) is the subject of many contributions, including the milestone book [9] and more recently [10].

DEL extends epistemic logic by adding a new operator $\langle \mathcal{E}, E_0 \rangle$ for modeling actions, thus allowing us to model a system in evolution, in our case a game.

Definition III.4. *The* syntax *of* DEL*, whose language is noted* $\mathcal{L}_{DEL}$*, is formed by the following grammar:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid K_a(\varphi) \mid \langle \mathcal{E}, E_0 \rangle \varphi$$

with $p \in AP$ *and* $a \in Agt$.

The operator $\langle \mathcal{E}, E_0 \rangle \varphi$ is composed of an *event model* $\mathcal{E}$, a set of *available events* E_0, and a $\mathcal{L}_{\text{DEL}}$ formula φ. $\langle \mathcal{E}, E_0 \rangle \varphi$ reads "After the execution of $\langle \mathcal{E}, E_0 \rangle$, the formula φ is true". His dual is defined as follows: $[\mathcal{E}, E_0]\varphi = \neg\langle \mathcal{E}, E_0 \rangle \neg\varphi$.

We now define *event models* and *(multi-)pointed event models*, their interaction with Kripke models and finally the semantics of this new modal operator.

Definition III.5. *An* event model *is a tuple* $\mathcal{E} = \langle E, (R_a^{\mathcal{E}})_{a\in Agt}, pre, post\rangle$ *with* E *a non-empty set of* events*,* $R_a^{\mathcal{E}} \subseteq E \times E$ *the* epistemic relation *between events,* $pre : E \to \mathcal{L}_{EL}$ *the* precondition *function and* $post : E \times AP \to \mathcal{L}_{EL}$ *the* postcondition *function.*

An event model can be seen as a Kripke model where worlds have been replaced by events. Epistemic relations keep the same meaning as before. For an event e, $pre(e)$ is the necessary condition to execute the event. For every $p \in AP$, $post(e, p)$ is the new value of p after executing event e.

A pair $\langle \mathcal{E}, e\rangle$ is called a *pointed event model*, where $e \in E$ represents the *actual event*. A pair $\langle \mathcal{E}, E_0\rangle$ is called a *multi-pointed event model* where $E_0 \subseteq E$ is the set of events that can be used as actual events.

Example III.3. *We consider the action of setting the code to 1 in the example II.1. This action is modeled by the couple* $(\mathcal{E}_{set1}, e_2)$ *defined in Appendix B.2.*

We represent this example with a graph, where nodes (rectangles for events) represent events, arrows the epistemic relations, and the pre/post conditions are indicated inside the node, next to its name, see Figure 4.

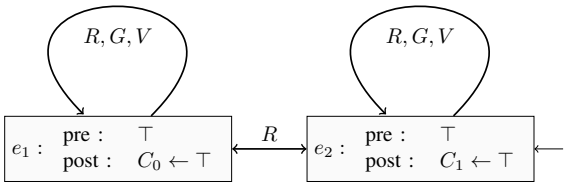

Figure 4: Pointed event model $(\mathcal{E}_{set1}, e_2)$ for setting the code to 1.

We now define the effect that an event model has on a Kripke model, named the *synchronous product* operation.

Definition III.6. *The* synchronous product *is an operation between a Kripke model* $\mathcal{M} = (W, (R_a)_{a\in Agt}, V)$ *and an event model* $\mathcal{E} = \langle E, (R_a^{\mathcal{E}})_{a\in Agt}, pre, post\rangle$ *that results in a new Kripke model* $\mathcal{M} \otimes \mathcal{E} = (W', (R'_a)_{a\in Agt}, V')$*. This operation is defined as follows:*

- $W' = \{(w, e) \in (W \times E) \mid \mathcal{M}, w \models pre(e)\}$;
- $R'_a = \{((w, e), (w', e')) \mid wR_aw' \text{ and } eR_a^{\mathcal{E}}e'\}$;
- $V'((w, e)) = \{p \in AP \mid \mathcal{M}, w \models post(e, p)\}$.

Intuitively, we create a new world (w, e) for any combination of a world w and event e such that w satisfies the precondition of e. The epistemic relation for agent a between (w, e) and (w', e') is kept if the agent could not distinguish w from w' nor e from e'. Proposition p is in the valuation of (w, e) if and only if w satisfies the postcondition of e for p.

The synchronous product operation can be extended to pointed models $(\mathcal{M}, w)$ and $(\mathcal{E}, e)$: the new pointed Kripke model is $((\mathcal{M} \otimes \mathcal{E}), (w, e))$.

Example III.4. *The synchronous product between the Kripke model* $\mathcal{M}_{S1}$ *of Example III.2 and the event model* $\mathcal{E}_{set1}$ *of example III.3 is represented in Figure 5. The formal details are in Appendix, see B.3.*

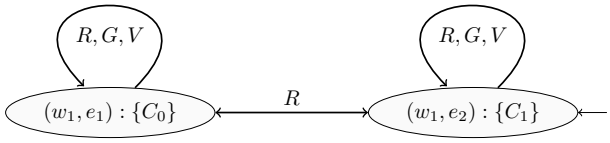

Figure 5: Pointed Kripke model $\big((\mathcal{M}_{S1} \otimes \mathcal{E}_{set1}), (w_1, e_2)\big)$.

Note that, as expected, the resulting pointed Kripke model is equivalent to $(\mathcal{M}_{S3}, w_2)$*, modulo the name of worlds.*

We now define the semantics of the operator $\langle \mathcal{E}, E_0\rangle$.

Definition III.7. *Semantics of the dynamic operator*

We extend the definition $\mathcal{M}, w \models \varphi$ *from* $\mathcal{L}_{EL}$ *to* $\mathcal{L}_{DEL}$ *with the following clause:*

$\mathcal{M}, w \models \langle \mathcal{E}, E_0\rangle\varphi$ *if there exists* $e \in E_0$ *such that* $\mathcal{M}, w \models pre(e)$ *and* $\mathcal{M} \otimes \mathcal{E}, (w, e) \models \varphi$.

When considering pointed event models with only one actual event, we allow the notation $\langle \mathcal{E}, e\rangle$ instead of $\langle \mathcal{E}, \{e\}\rangle$.

Example III.5. *Formulas of* $\mathcal{L}_{DEL}$

For example, using the previously defined models, we have:

$\mathcal{M}_{S1}, w_1 \models \langle \mathcal{E}_{set1}, e_2\rangle K_G(\neg C_0)$*;*

$\mathcal{M}_{S1}, w_1 \not\models \langle \mathcal{E}_{set1}, e_2\rangle K_R(\neg C_0)$.

Since the guardian knows which code he has set but not the robber.

The model checking problem for *DEL* is defined as follows.

Input: A pointed Kripke model $(\mathcal{M}, w)$ and a formula $\varphi \in \mathcal{L}_{\text{DEL}}$;

Output: Yes if and only if $\mathcal{M}, w \models \varphi$.

IV. Problem and reduction from Games to DEL

We consider the following decision problem EasyWin.

Input: A turn-based game with imperfect information G, an integer k and a set $Goal_{a_1}$ of winning states for agent $a_1 \in Agt$.

Output: Yes if and only if there exists a k-bounded uniform winning strategy for agent a_1.

This section objective is to prove that EasyWin is a member of PSPACE. To do so, we polynomially reduce it to the *DEL* model checking problem, which is in PSPACE [6].

To prove that, we need to define the size of a game along with the size of a Kripke model and event models. In the next two definitions, we consider that the size $|f|$ of a function $f : A \to B$ is the product of its definition sets' size, $|A| \times |B|$.

Definition IV.1. *We define the* size *of a game G as follows:*
$G = \langle Agt, AP, St, Val, Act, Succ, Obs\rangle$
$|G|$ is the sum of the size of its components;
$|Val| = |St| \times |AP|$;
$|Succ| = |St| \times |St| \times |Act|$;

Note that $|G| \in \mathcal{O}(|St| \times |Act| \times |Agt| \times |AP|)$.

Definition IV.2. *The* size *of formulas $\varphi \in \mathcal{L}_{DEL}$ and models $\mathcal{M} = (W, (R_a)_{a\in Agt}, V)$, $\mathcal{E} = (E, (R_a^{\mathcal{E}})_{a\in Agt}, pre, post)$ are defined by induction as follows:*
$|p| = 1$;
$|\neg\varphi| = |K_a(\varphi)| = 1 + |\varphi|$;
$|\varphi_1 \vee \varphi_2| = |\varphi_1| + |\varphi_2|$;
$|\langle\mathcal{E}, E_0\rangle\varphi| = 1 + |\mathcal{E}| + |E_0| + |\varphi|$;
$|\mathcal{M}| = |AP| \times |W| + \sum_{a\in Agt} |R_a|$;
$|\mathcal{E}| = |E| + \sum_{e\in E} \big(|pre(e)| + \sum_{p\in AP} |post(e,p)|\big) + \sum_{a\in Agt} |R_a^{\mathcal{E}}|$.

Note that, as usual in decision problems with a bound [11], the bound k uses k memory space, i.e. is written in unary. This is realistic because we need to store a strategy of which size is k and not $\log(k)$. If one does not tolerate this, consider the problem of finding a $\log(k)$-bounded winning strategy. In the following, this convention is accepted.

A. Transformation function

Definition IV.3.

Input: *A turn-based game with imperfect information $G = \langle ...\rangle$, a bound k and the goal set $Goal_{a_1}$ for agent a_1.*

Output: *A pointed Kripke model $(\mathcal{M}, w)$, with $\mathcal{M} = (W, R_{a_1}, V)$, and a formula $\varphi_{a_1}^k$ such that $\mathcal{M}, w \models \varphi_{a_1}^k$ if and only if there exists a k-bounded uniform winning strategy for agent a_1.*

Body:

1) Generating AP and Agt: Add $wait$ to AP and after that, keep untouched. Also keep Agt untouched.

2) Creation of the pointed Kripke model $(\mathcal{M}, w_{s_{init}})$:
$\mathcal{M} = (W, R_{a_1}, V)$;
$W = \{w_s \mid s \in St \text{ and } s \sim_{a_1} s_{init}\}$;
$R_{a_1} = W \times W$;
$V(w) = Val(w)$.

For example, if we try to transform our robber example, the initial Kripke model would be $(\mathcal{M}_{S1}, w_s)$ (with only Robber's epistemic relations).

3) Creation of the formula expressing the bounded uniform winning strategy:

In this section, we create a formula $\varphi_{a_1}^k$ that is true if and only if there exists a k-bounded winning strategy for agent a_1.

We define φ_X the formula corresponding to X. For example, if X is a valuation, then φ_X is the conjunction that details which atomic propositions are true and which are false. If X is an assertion, then the formula φ_X is either $\top$ or $\bot$.

We define $\varphi_{a_1}^0 = \bigvee_{s\in Goal_{a_1}} \varphi_{Val(s)}$, which corresponds to being in a winning state, i.e. having a 0-bounded winning strategy.

We need a formula to represent the course of the game, to express how actions are played. For the next event models, the details of the construction can be found in the Appendix C.

- For every agent a and action $\alpha \in Act_a$, we first define the multi-pointed event model of "agent a plays action α", noted $(\mathcal{E}_a^\alpha, E_0^\alpha)$ (see Appendix C.1). We note $(\mathcal{E}_a, E_0^a)$ the multi-pointed event model corresponding to the disjoint union of the $(\mathcal{E}_a^\alpha, E_0^\alpha)$ for every α in Act_a (see Appendix C.2). Therefore, $(\mathcal{E}_a, E_0^a)$ corresponds to the turn of a. He can play the action he wants (E_0^a is the totality of events), but must be in a compatible state (preconditions) and the resulting action leads to the successor state (postcondition). We also add one more event, e_{wait}, to allow the execution of this multi-pointed event model even if it is not agent a's turn.
- We then define a pointed event model, noted $(\mathcal{E}_{wait}, e_{wait})$ corresponding to "the turn is over, here begins a new one" (see Appendix C.3).
- We finally define pointed event models noted $(\mathcal{E}_p, E_0^p)$ for every $p \in Obs_{a_1}$, each corresponding to "agent a_1 observes the value of the atomic proposition p". We note the first one $(\mathcal{E}_p, E_0^p)$ and the last one $(\mathcal{E}_r, E_0^r)$ (see Appendix C.4).

An example of these three preceding steps applied to the robber example is in Appendix Example C.1.

We now define a turn in the game, noted $\mathcal{T}(\varphi)$, where φ describes the rest of the game. It is defined as follows:
$\mathcal{T}(\varphi) = \varphi_{a_1}^0 \vee \langle\mathcal{E}_{a_1}, E_0^{a_1}\rangle K_{a_1}\big([\mathcal{E}_{a_2}, E_0^{a_2}]...[\mathcal{E}_{a_{|Agt|}}, E_0^{a_{|Agt|}}]$
$\langle\mathcal{E}_{wait}, e_{wait}\rangle\langle\mathcal{E}_p, E_0^p\rangle...\langle\mathcal{E}_r, E_0^r\rangle\varphi\big)$.

The idea of $\mathcal{T}(\varphi)$ is "agent a_1 has won, or else agent a_1 can play an action such that he knows that, regardless of what others play, (and after observing propositions) φ is true". Note that there is only one agent who plays per turn, but there is an event model for every agent, and all of them except one will perform the move e_{wait} because it is not their turn to play.

We note $\varphi_{a_1}^{k+1} = \mathcal{T}(\varphi_{a_1}^k)$. To reword, satisfying $\varphi_{a_1}^k$ means that agent a_1 either wins directly or can play an action such that he knows that, no matter what other agents do, he has a $(k-1)$-bounded uniform winning strategy.

Example IV.1. *We still consider that winning states for agent Robber are S7 and S11. We have:*
$\varphi_R^0 = \varphi_{Val(S7)} \vee \varphi_{Val(S11)}$;
$\mathcal{T}(\varphi) = \varphi_R^0 \vee \langle\mathcal{E}_R, E_0^R\rangle K_R([\mathcal{E}_G, E_0^G][\mathcal{E}_V, E_0^V]\langle\mathcal{E}_{wait}, e_{wait}\rangle$
$\langle\mathcal{E}_{C_0}, E_0^{C_0}\rangle \dots \langle\mathcal{E}_{Op}, E_0^{Op}\rangle\varphi)$;
$\varphi_R^3 = \mathcal{T}(\mathcal{T}(\mathcal{T}(\varphi_R^0)))$;
with $(\mathcal{E}_R, E_0^R)$ composed of the disjoint union of $(\mathcal{E}_R^{play0}, E_0^{play0})$ and $(\mathcal{E}_R^{play1}, E_0^{play1})$, constructed as indicated in the reduction.

This transformation function returns the pointed Kripke model $(\mathcal{M}, w)$ and the formula $\varphi^k_{a_1}$.

B. This reduction was polynomial

We have to prove that the reduction was in polynomial time. Let $(G, k, Goal_{a_1})$ be the input and $(\mathcal{M}, w)$ the resulting pointed Kripke model with $\mathcal{M} = (W, R_{a_1}, Val)$ and $\varphi^k_{a_1}$ the resulting formula of the transformation function.

We note $t(X)$ the creation duration of a certain object X. Roughly we have $t(X) \in \mathcal{O}(|X|)$.

Therefore, we deduce that the execution time for the transformation function is in $\mathcal{O}(k \times |Agt| \times |Act| \times |St| \times |AP|)$, so polynomial.

C. Correction of the transformation function

Let $(G, k, Goal_{a_1})$ be the input and $(\mathcal{M}, w)$ the resulting pointed Kripke model with $\mathcal{M} = (W, R_{a_1}, Val)$ and $\varphi^k_{a_1}$ the resulting formula of the transformation function.

The assertion that we prove here, called $H(k)$, is: "For all game G, bound k and set $Goal_{a_1}$, there is a k-bounded uniform winning strategy for agent a_1 if and only if the constructed formula $\varphi^k_{a_1}$ is satisfied by the constructed pointed Kripke model".

We prove this proposition by induction on k.

- Case $k = 0$:
  These strategies correspond to winning directly. The corresponding formula $\varphi^0_{a_1}$ is simply the disjunction of the formulas $\varphi_{Val(s)}$ for $s \in Goal_{a_1}$. We then have to check if the pointed Kripke model $(\mathcal{M}, w)$ satisfies the formula corresponding to being in a winning state. There is a winning strategy in 0 moves if and only if the game is already on a winning state. The initial world is $w_{s_{init}}$, so the equivalence is true.
- Case $k > 0$:
  We suppose that $H(k-1)$ is true. We distinguish two possible cases:
  - It is agent a_1's turn. Game speaking, agent a_1 has a k-bounded winning strategy if and only if he can choose an action such that he knows that he has a $(k-1)$-bounded uniform strategy. Recall that only person plays per turn, so all other agents are performing the action e_{wait}. DEL speaking, the formula $\varphi^k_{a_1}$ is true if and only if the application of $\langle \mathcal{E}_{a_1}, E_0^{a_1} \rangle$ on the current model leads to a model where agent a_1 knows that the formula $\varphi^{k-1}_{a_1}$ is true. We know that applying the event model $\langle \mathcal{E}_{a_1}, E_0^{a_1} \rangle$ is equivalent to choosing an action for agent a_1. By induction, satisfying a $\varphi^{k-1}_{a_1}$ is equivalent to having a $(k-1)$-bounded uniform strategy. Therefore, the assertion $H(k)$ is true in this subcase.
  - It is not a_1's turn. Game speaking, agent a_1 has a k-bounded winning strategy if and only if he knows that whatever his opponent a_i play, he has a $(k-1)$-bounded uniform strategy. DEL speaking, the formula $\varphi^k_{a_1}$ is true if and only if the agent a_1 knows that, in the resulting model by the application of one event model $[\mathcal{E}_{a_i}, E_0^{a_i}]$ on the current model, the formula $\varphi^{k-1}_{a_1}$ is true. We know that applying the event model $[\mathcal{E}_{a_i}, E_0^{a_i}]$ is equivalent to choosing an action for agent a_i. By induction, satisfying a $\varphi^{k-1}_{a_1}$ is equivalent to having a $(k-1)$-bounded uniform strategy. Therefore, the assertion $H(k)$ is also true in this subcase.

  Consequently, the assertion $H(k)$ is proved for every value of k.

V. Conclusion

We proved a PSPACE membership result for the problem of finding bounded winning uniform strategies in turn-based games with imperfect information. This reduction also paved the road to a more interesting result. Indeed, those games can be represented in a *symbolic* way, thus avoiding the exponential description of the states. These symbolic games can then be reduced to *symbolic DEL*, whose model checking is also in PSPACE [12]. This symbolic reduction is almost the same as the one we did here, and an intuition is provided in the Appendix D.

The problem here was about the *existence* of a bounded winning uniform strategy, but this solution can be used to extract an artificial intelligence. Let us say that we have a game and an agent a_1 for whom there exists a k-bounded winning uniform strategy. When it's agent a_1's turn (at the $n^{th} \leqslant k$ turn of the game), we just have to construct the formula $\varphi^k_{a_1}$, replacing past actions by the actions that have been actually played (or all actions that he cannot distinguish from the real actions), and testing if he has a $(k-n)$-bounded winning uniform strategy for each of its actual choices of actions. When he finds such an action, he plays it. It is necessary that he finds at least one action to play because he had a winning strategy in the first place, and followed it.

Finding *winning* strategies is most of the time doomed to failure, and agents may be more interested in finding strategies with a high *probability* of victory. There exist extensions with probability both in the domain of games [13] and epistemic logic [14]/dynamic epistemic logic [15], so we could use them as well.

References

[1] R. Alur, T. A. Henzinger, and O. Kupferman, "Alternating-time temporal logic," *J. ACM*, 2002.

[2] W. Jamroga, *Logical Methods for Specification and Verification of Multi-Agent Systems*. Institute of Computer Science, Polish Academy of Sciences, 2015.

[3] E. D. Demaine and R. A. Hearn, "Constraint logic: A uniform framework for modeling computation as games," in *Conference on Computational Complexity*, 2008.

[4] A. Baltag, L. S. Moss, and S. Solecki, "The logic of public announcements and common knowledge and private suspicions," in *Theoretical Aspects of Rationality and Knowledge*, 1998.

[5] S. Arora and B. Barak, *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[6] G. Aucher and F. Schwarzentruber, "On the complexity of dynamic epistemic logic," in *Theoretical Aspects of Rationality and Knowledge*, 2013.

[7] J. Hintikka, "Knowledge and belief. an introduction to the logic of the two notions," 1962.

[8] R. Fagin, J. Y. Halpern, Y. Moses, and M. Vardi, *Reasoning about knowledge*. MIT press, 2004.

[9] H. Van Ditmarsch, W. van Der Hoek, and B. Kooi, *Dynamic epistemic logic*. Springer Science & Business Media, 2007.
[10] H. Ditmarsch, J. Y. Halpern, W. van der Hoek, and B. P. Kooi, *Handbook of epistemic logic*. College Publications, 2015.
[11] H. Turner, "Polynomial-length planning spans the polynomial hierarchy," vol. 2424, 08 2002.
[12] T. Charrier and F. Schwarzentruber, "A succinct language for dynamic epistemic logic," in *Autonomous Agents and Multi-Agent Systems*, 2017.
[13] K. R. Apt and E. Grädel, *Lectures in game theory for computer scientists*. Cambridge University Press, 2011.
[14] J. Y. Halpern and M. R. Tuttle, "Knowledge, probability, and adversaries," *J. ACM*, 1993.
[15] B. P. Kooi, "Probabilistic dynamic epistemic logic," *Journal of Logic, Language and Information*, 2003.
[16] T. Charrier and F. Schwarzentruber, "Complexity of dynamic epistemic logic with common knowledge," in *Advances in Modal Logic*, 2018.
[17] S. Reisch, "Hex is pspace-complete," *Acta Informatica*, 1981.

APPENDIX A
APPENDIX OF SECTION II

In Sections A and B, we consider the same agents $Agt = \{R(obber), G(uard), V(ault)\}$ and atomic propositions $AP = \{C_0, C_1, R_0, R_1, Al, Op\}$.

Example A.1.

The formalization of the robber example with the game definition is: $G = \langle Agt, AP, St, Val, Act, Succ, Obs\rangle$ *with:*

$St = \{S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11\}$

$s_{init} = S1$

Val :	S1 $\emptyset$	S2 $\{C_0\}$	S3 $\{C_1\}$
S4 $\{C_0, R_0\}$	S5 $\{C_0, R_1\}$	S6 $\{C_1, R_0\}$	S7 $\{C_1, R_1\}$
S8 $\{C_0, R_0, Op\}$	S9 $\{C_0, R_1, Al\}$	S10 $\{C_1, R_0, Al\}$	S11 $\{C_1, R_1, Op\}$

$Act = Act_R \cup Act_G \cup Act_V$;

$Act_R = \{Play_0, Play_1\}$; $Act_G = \{Set_0, Set_1\}$; $Act_V = \{Ring, Unlock\}$;

$Succ$:	S1	S2	S3	S4	S5	S6	S7	S8/9/10/11
Set_0	S2	/	/	/	/	/	/	/
Set_1	S3	/	/	/	/	/	/	/
$Play_0$	/	S4	S6	/	/	/	/	/
$Play_1$	/	S5	S7	/	/	/	/	/
$Ring$	/	/	/	/	S9	S10	/	/
$Unlock$	/	/	/	S8	/	/	S11	/

$Obs_G = \{C_0, C_1, Al, Op\}$; $Obs_R = \{R_0, R_1, Al, Op\}$; $Obs_V = \{C_0, C_1, R_0, R_1, Al, Op\}$.

APPENDIX B
APPENDIX OF SECTION III

In this section, when defining *post* function, all atomic propositions not mentioned are considered to remain at the same value (i.e. $post(e, p) = p$).

Example B.1. *Kripke models* $(\mathcal{M}_{S1}, w_1)$ *and* $(\mathcal{M}_{S3}, w_2)$.

The state $S1$ *of the previous example corresponds to the following pointed Kripke model* $(\mathcal{M}_{S1}, w_1)$ *where:*

$\mathcal{M}_{S1} = (W, (R_a)_{a\in Agt}, Val)$;

$W = \{w_1\}$;

$R_R = R_G = R_V = \{(w_1, w_1)\}$;

$Val(w_1) = \emptyset$;

The state $S3$ *of the previous example corresponds to the following pointed Kripke model* $(\mathcal{M}_{S3}, w_2)$ *where:*

$\mathcal{M}_{S3} = (W, (R_a)_{a\in Agt}, Val)$;

$W = \{w_1, w_2\}$;

$R_R = W \times W$; $R_G = R_V = \{(w_1, w_1), (w_2, w_2)\}$;

$Val(w_1) = \{C_0\}$; $Val(w_2) = \{C_1\}$.

Example B.2. *Event model* $(\mathcal{E}_{set1}, e_2)$.

$\mathcal{E}_{set1} = \langle E, (R^{\mathcal{E}}_a)_{a\in Agt}, pre, post\rangle$;

$E = \{e_1, e_2\}$;

$R^{\mathcal{E}}_R = E \times E$; $R^{\mathcal{E}}_G = R^{\mathcal{E}}_V = \{(e_1, e_1), (e_2, e_2)\}$;

$pre(e_1) = \top$; $pre(e_2) = \top$;

$post(e_1, C_0) = \top$; $post(e_2, C_1) = \top$.

Example B.3.

$\mathcal{M}_{S1} = (W, (R_a)_{a\in Agt}, V)$ *is as in Example III.2;*

$\mathcal{E}_{set1} = \langle E, (R^{\mathcal{E}}_a)_{a\in Agt}, pre, post\rangle$ *is as in Example III.3;*

We have $\mathcal{M}_{S1} \otimes \mathcal{E}_{set1} = (W', (R'_a)_{a\in Agt}, V')$ *with:*

$W' = \{(w_1, e_1), (w_1, e_2)\}$;

$R'_R = W' \times W'$;

$R'_G = R'_V = \{((w_1, e_1), (w_1, e_1)), ((w_1, e_2), (w_1, e_2))\}$;

$V'((w_1, e_1)) = \{C_0\}$; $V'((w_1, e_2)) = \{C_1\}$.

APPENDIX C
APPENDIX OF SECTION IV

Definition C.1. *Event models* $(\mathcal{E}^\alpha_a, E^\alpha_0)$

In those models, we only represent agent a_1*'s knowledge, because he is the one we search a winning uniform strategy for, and we do not care about what other agents know.*

$\mathcal{E}^\alpha_a = \langle E^\alpha, R^\alpha_{a_1}, pre^\alpha, post^\alpha\rangle$; $E^\alpha_0 = E^\alpha$;

$E^\alpha = \{e^\alpha_s \mid s \in St \text{ and } \alpha \in Play_a(s)\}$;

$R^\alpha_{a_1} = E \times E$;

For every event e^α_s, $pre^\alpha(e^\alpha_s) = \varphi_{Val(s)} \wedge \neg wait$;

For every event e^α_s, $post^\alpha(e^\alpha_s, wait) = \top$ *and for every other atomic proposition* p, $post^\alpha(e^\alpha_s, p) = \varphi_{(Val(Succ(s,\alpha))\models p)}$.

Definition C.2. *Event model* $(\mathcal{E}_a, E^a_0)$

We note $E^\forall_a = \bigsqcup_{\alpha\in Act_a} E^\alpha$.

$\mathcal{E}_a = \langle E, R^{\mathcal{E}}_{a_1}, pre, post\rangle$; $E^{\mathcal{E}}_0 = \big(\bigsqcup_{\alpha\in Act_a} E^\alpha_0\big) \sqcup \{e_{wait}\}$;

$E = E^\forall_a \sqcup \{e_{wait}\}$;

Because agent a_1 *knows which actions he plays but not which action opponents play, if* a *is* a_1*, then*

$R^{\mathcal{E}}_{a_1} = \big(\bigsqcup_{\alpha\in Act_a} R^\alpha_{a_1}\big) \sqcup \{(e_{wait}, e_{wait})\}$;

and else

$R^{\mathcal{E}}_{a_1} = E^\forall \times E^\forall \sqcup \{(e_{wait}, e_{wait})\}$;

For every action $\alpha \in Act_a$ *and event* $e \in E^\forall_a$, $pre(e) = pre^\alpha(e)$*, and for every atomic proposition* p, $post(e, p) = post^\alpha(e, p)$;

$pre(e_{wait}) = wait \vee \big(\bigwedge_{s|Turn(s)=a} \neg\varphi_{Val(s)}\big)$.

Definition C.3. *Event model* $(\mathcal{E}_{wait}, e_{wait})$

The event model $\mathcal{E}_{wait} = \langle E, R^{\mathcal{E}}_{a_1}, pre, post\rangle$ *is defined as follows:*

$E = \{e_{wait}\}$;

$R^{\mathcal{E}}_{a_1} = \{(e_{wait}, e_{wait})\}$;

$pre(e_{wait}) = \top$;

$post(e_{wait}, wait) = \bot$.

Definition C.4. *Event model* $(\mathcal{E}_p, E^p_0)$

$\mathcal{E}_p = \langle E, R^{\mathcal{E}}_{a_1}, pre, post\rangle$; $E^p_0 = E$;

$E = \{e_p, e_{\neg p}\}$;

$R^{\mathcal{E}}_{a_1} = \{(e_p, e_p), (e_{\neg p}, e_{\neg p})\}$;

$pre(e_p) = p$; $pre(e_{\neg p}) = \neg p$.

Example C.1. *Here is an example of the transformation function on the example II.1.*

We can find the multi pointed models corresponding to the 3 steps of the creation of event models. The first step is separated into two substeps. The following models are obtained with the method described in the reduction.

Figure 6 corresponds to the guardian setting the code to 1, Figure 7 to playing the guardian's turn, Figure 8 to stop waiting for the next turn and Figure 9 to observing the proposition p.

We introduce the notation $\varphi_{Val(S1)} = \neg C_0 \wedge \neg C_1 \wedge \neg R_0 \wedge \neg R_1 \wedge \neg Al \wedge \neg Op$.

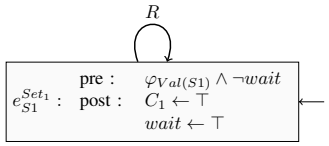

Figure 6: Pointed event model $(\mathcal{E}_G^{Set_1}, e_{S1}^{Set_1})$, corresponding to the guardian setting the code to 1.

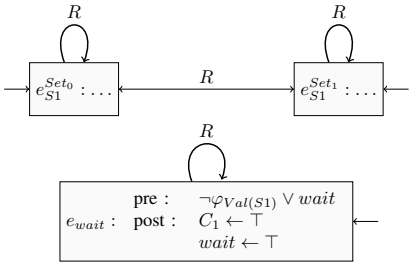

Figure 7: Multi pointed event model $(\mathcal{E}_G, E_0^G)$, corresponding to the turn of the guardian.

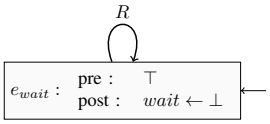

Figure 8: Pointed event model $(\mathcal{E}_{wait}, e_{wait})$ corresponding to stop waiting for next turn.

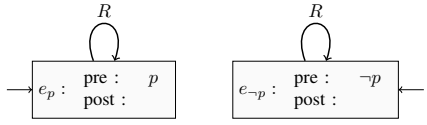

Figure 9: Multi pointed event model $(\mathcal{E}_p, E_0^p)$ corresponding to observing p.

APPENDIX D
SYMBOLIC METHODS

Our definition of game arenas suffers from a combinatorial explosion in the number of states. To solve this problem, we create a symbolic representation of a turn-based game arena with imperfect information. It consists in representing sets by boolean formulas whose elements are the valuations of the formulas. We note $L_0(AP)$ the set of boolean formulas on AP. The elements that are represented symbolically are: St, the states, $Succ$, the successor function and $Play_a$, the playable function. Moreover, Val, the valuation function, disappears because states are already represented by their valuation, so we may use "state" or "valuation" interchangeably.

Definition D.1. *A* symbolic turn-based game arena with imperfect information*, G, is the following tuple:*

$G = \langle Agt, AP, St, Act, Play, Succ, Obs \rangle$

where Agt, AP, Act and Obs are defined as in definition II.1 and:

- $St \in L_0(AP)$ *with an* initial state $s_{init} \in Val(AP)$ *such that* $s_{init} \models St$*;*
- $Succ : Act \to L_0(AP \cup AP')$ *with* $AP' = \{p' \mid p \in AP\}$*. We suppose* $AP \cap AP' = \emptyset$*;*
- $Play$ *is the tuple of* $Play_a$ *for each agent* a*.* $Play_a : Act_a \to L_0(AP)$*;*

Intuitively, St is a formula on AP such that every valuation that satisfies St is a state, s_{init} is the initial valuation, $Play_a$ is the function that returns the formula that represents the states in which the agent a can play the given action. For $Succ$, we need to introduce a new notation: for a valuation s, we note $prime(s)$ the exact same valuation, but on AP' (i.e. every propositional variable is replaced by its equivalent on AP'). There is a transition from state s to state s' using action α if and only if $(s \cup prime(s') \models Succ(\alpha))$.

Note that $Play_a$ is included in the definition this time since computing it may require exponential time. Thus, we impose that it is computed beforehand. It is not a problem in practice since specifying which actions agent a_1 may play is easy. For instance, the action "play card i" is playable in states where agent a_1 has card i.

For example, we can redefine the turn-based constraint.

Constraint D.1. *Turn-based constraint*

For every state s such that $s \models St$, there exists at most an agent a such that $s \models \bigvee_{\alpha \in Act_a} Play_a(\alpha)$ and for all other agents $b \in (Agt \setminus \{a\})$, $s \not\models \bigvee_{\alpha \in Act_b} Play_b(\alpha)$.

The other functions and constraints are defined similarly to what we did in section II.

The definitions of a history and a bounded winning uniform strategy remain the same.

Symbolic dynamic epistemic logic (*SDEL*) is defined with the same ideas. A clear definition can be found in [12] and a simpler definition in [16].

The new decision problem is the following:

Definition D.2. *Symbolic EasyWin decision problem*

Input:	A Symbolic turn-based game with imperfect information G, an integer k and a formula $Goal_{a_1}$ of goals for agent a_1.
Output:	Yes if and only if there exists a k-bounded uniform winning strategy for agent a_1.

The same reduction as in part IV can be done with symbolic methods by transposing the definition of models with their symbolic counterparts. Due to the lack of space, we omit the details here.

Once this reduction is properly written, the result will be that this new decision problem is also a member of PSPACE because the model checking problem in *SDEL* is also in PSPACE. We suspect that this it is also PSPACE-hard since for instance finding strategies for Hex is PSPACE-hard and specifying it as a symbolic game arena should be direct [17].