Concurrent Games in Dynamic Epistemic Logic

Bastien Maubert\textsuperscript{1}, Sophie Pinchinat\textsuperscript{2}, François Schwarzentruber\textsuperscript{2} and Silvia Stranieri\textsuperscript{1}

\textsuperscript{1}Università degli Studi di Napoli “Federico II”, Italy
\textsuperscript{2}Université de Rennes, IRISA, CNRS, France

Abstract

Action models of Dynamic Epistemic Logic (DEL) represent precisely how actions are perceived by agents. DEL has recently been used to define infinite multi-player games, and it was shown that they can be solved in some cases. However, the dynamics being defined by the classic DEL update product for individual actions, only turn-based games have been considered so far. In this work we define a concurrent DEL product, propose a mechanism to resolve conflicts between actions, and define concurrent DEL games. As in the turn-based case, the obtained concurrent infinite game arenas can be finitely represented when all actions are public, or all are propositional. Thus we identify cases where the strategic epistemic logic ATL\textsuperscript{K} can be model checked on such games.

1 Introduction

The discipline of \textit{distributed synthesis} seeks solutions for automating the design of artificial agents that interact and make decisions in order to achieve some service. The main challenge is to synthesize strategies for each agent of a team collaborating towards a common goal, against opponents, usually in an imperfect information setting.

As advocated in the Dynamic Epistemic Logic (DEL) approach to \textit{epistemic planning}, we consider multi-agent systems described symbolically by \textit{DEL presentations}, which consist in an initial epistemic state (actual world and worlds considered possible by some agents), and epistemic actions that agents may perform. An epistemic action, besides the standard precondition and effects of the action, also describes the agents’ ability to perceive its execution. Synthesising strategies was studied in the single-agent case \cite{Bolander2011,Aucher2013,Bolander2015,Bolander2017}, and in a multi-player turn-based game setting \cite{Maubert2019,Maubert2020}.

However, concurrent execution of actions has not been considered so far in epistemic planning. Some works consider concurrent execution of abstract actions (as in \textit{concurrent game structures} \cite{deAlfaro2000}), or concurrent execution of purely epistemic actions without effects on the world \cite{vanEijck2011,Ágotnes2014}. But to our knowledge, concurrent execution of arbitrary epistemic actions with explicit effects has never been studied. And yet they are essential for modelling realistic situations.

Consider the two-robot coordination example of Figure 1. Some container can be placed either left or right. Two robots (or agents), \(i\) and \(j\), can either wait or push the container, which incidentally is energy-consuming. Both agents waiting, the container stays in place. Otherwise, if \(i\) pushes while \(j\) waits, the container moves right (if not already there), and, symmetrically, if \(j\) pushes while \(i\) waits, it moves left (if not already there). Agents pushing concurrently results in a conflict. To determine the effect of such conflicting actions, one solution is to consider that they are \textit{blocking}, as done for instance in [Fox and Long, 2003] for in temporal planning, or facts involved in conflicts are just maintained \cite{Lima2014}. Another option consists in selecting a maximal subset of non-conflicting actions, as done for instance in [Eshuis and Wieringa, 2001] in UML workflow modelling.

In this paper, we adopt the latter solution, which is more challenging. A major additional difficulty with respect to this latter UML setting arises from epistemic features of actions of the DEL framework. In our robot example, while robots \(i\) and \(j\) perfectly perceive pushing and waiting actions, the two other agents \(k\) and \(l\) cannot distinguish them. Note that this issue is not tackled in \cite{Lima2014} since actions are public.

Our first contribution is to define a \textit{DEL concurrent update product} that provides the dynamics of concurrent epistemic actions. This new product yields a non-deterministic dynamics controlled by a \textit{scheduler}, whose role is to resolve conflicts. On the basis of this product we show how finite DEL presentations can generate infinite concurrent game arenas.

Our second contribution is a proof that distributed synthesis can be solved for such games for two cases: the case where all actions are public (perfectly observed by everyone), and the case where all actions have propositional pre- and post-
conditions and information among the agents is hierarchical (Theorems 4 and 7). To prove this we extend to the concurrent setting some results from [Maubert et al., 2019; Maubert et al., 2020], namely, that the infinite concurrent game arenas arising from DEL game presentations can be finitely represented when actions are all public or all propositional. We then transfer existing results on the model-checking problem for the epistemic strategic logic $\text{ATL}^*_\text{c}$ [Van der Hoek and Wooldridge, 2003] to obtain, in particular, decidability of distributed synthesis for rich temporal epistemic objectives.

Remark that our work is unrelated to the concurrent dynamic logic from [Van Benthem et al., 2007], where several games are played concurrently. Our work also differs from concurrent dynamic epistemic logic [Van Ditmarsch et al., 2003], where concurrency represents some form of alternation [Chandra and Stockmeyer, 1976].

Outline: In Section 2, we provide all the necessary material on DEL presentations. In Section 3 we present the concurrent update product, and we use it in Section 4 to define DEL games and establish our main results.

2 Background on DEL

We fix a finite set of agents $A_g = \{1, \ldots, N\}$ and a countable set of atomic propositions $AP$. The language of epistemic logic, noted $\mathcal{L}_K$, is defined by the grammar $\varphi ::= \text{true} \mid p \mid \neg \varphi \mid \varphi \land \varphi \mid K_i \varphi$ where $p \in AP$ and $i \in A_g$. Formula $K_i \varphi$ reads as “agent $i$ knows that $\varphi$”. Ruling out modality $K_i$ yields propositional logic.

2.1 Epistemic Models and States

Epistemic models are relational structures that represent agents’ knowledge.

Definition 1 (Epistemic models). An epistemic model is a structure $M = (W, (\approx_i)_{i \in A_g}, \lambda)$ where $W$ is a finite non-empty set of worlds, for each agent $i \in A_g$, the equivalence relation $\approx_i \subseteq W \times W$ is an epistemic relation that reflects agents $i$’s information (or knowledge), and $\lambda : W \rightarrow 2^{AP}$ is a valuation function. An epistemic state is a pair $\langle M, w \rangle$ where $M$ is an epistemic model and $w$ is a world of $M$ representing the actual world.

An epistemic state $\langle M, w \rangle$ will often be written $w$ when the model $M$ is clear from the context.

Example 1. The epistemic state of Figure 2 represents the situation of Figure 1. Proposition $\text{left}$ holds when the container is on the left. In the actual world $w$, $\text{left}$ is false since the container is on the right and we assume that agents $i, j$ and $k$ know this fact, but that $\ell$ does not: he does not distinguish between world $w$ and the world $u$ where the container is on the left.

The semantics of $\mathcal{L}_K$ is based on epistemic states.

Definition 2 (Semantics of $\mathcal{L}_K$). Let $\langle M, w \rangle$ be an epistemic state. We define the sentence “Formula $\varphi$ holds in $\langle M, w \rangle$”, written $\langle M, w \rangle \models \varphi$ or $w \models \varphi$, by induction on $\varphi$: $w \models \text{true}$; $w \models p$ whenever $p \in \lambda(w); w \models \neg \varphi$ whenever $w \not\models \varphi$; $w \models \varphi_1 \land \varphi_2$ whenever $w \models \varphi_1$ and $w \models \varphi_2$; and $w \models K_i \varphi$ whenever $(M, u) \models \psi$, for all worlds $u$ such that $w \approx_i u$.

2.2 Action Models and Epistemic Actions

Action models ([Baltag et al., 1998], [Van Ditmarsch et al., 2007]) gather several “concrete” actions (with preconditions and effects [Van Ditmarsch and Kooi, 2006]) with epistemic relations describing how the concrete actions are perceived.

Definition 3 (Action models). An action model is a structure $\mathcal{A} = (A, (\approx_i)_{i \in A_g}, \pre, \post, \lambda)$ where $A$ is a nonempty finite set of actions, $\approx_i \subseteq A \times A$ is the epistemic relation of agent $i \in A_g$, the precondition function $\pre : A \rightarrow \mathcal{L}_K$ specifies the guard for an action to be executable, and the postcondition is a partial function $\post : A \times AP \rightarrow \mathcal{L}_K$ that describes the effect of actions over atomic propositions; when $\post(a, p)$ is undefined, $p$ is not affected by $a$. An epistemic action is a pair $\langle \mathcal{A}, a \rangle$ where $a \in A$ denotes the action actually executed.

Example 2. Figure 3 describes the action model for the two-robot coordination example. Action $\text{push}_i$ represents agent $i$ pushing the container to the right. Agent $i$ can execute $\text{push}_i$ whenever its energy is high, i.e., when proposition $\text{low}E_i$ is false ($\text{low}E_i \equiv 1$); this action consumes energy if it is effective (i.e., the container was not already on the right): in that case the postcondition makes proposition $\text{low}E_i$ true. Action $\text{push}_j$, for agent $j$ is defined similarly. Finally, the waiting action of robots, represented by $\text{wait}_i$ and $\text{wait}_j$ respectively, have the true precondition and the effect of refilling energy (i.e., turning $\text{low}E_i$ to false). To keep our example simple, we suppose that all agents but $\ell$ distinguish between $\text{push}_i$ and $\text{wait}_i$ even if the box is on the right, and similarly for $\text{push}_j$ and $\text{wait}_j$.

We exhibit two particular kinds of epistemic actions.

Definition 4 (Propositional, public action). An action model $\mathcal{A}$ is propositional if for every concrete action $a$ of $\mathcal{A}$ and each proposition $p$, formulas $\pre(a)$ and $\post(a)(p)$ are propositional. An epistemic action $\langle \mathcal{A}, a \rangle$ is public if for all agents $i, a \approx_i a'$ if and only if $a' = a$.
2.3 Update Product

Execution of epistemic actions may change the world but also agents’ information. It is formalised by the update product between epistemic models and action models.

**Definition 5 (Update product).** The update product of epistemic model $M = (W, (\approx_i)_{i \in Ag}, \lambda)$ with action model $A = (A_i, (\approx_i)_{i \in Ag}, \text{pre}, \text{post})$ is the epistemic model $M \otimes A := (W_\otimes, (\approx^\otimes_i)_{i \in Ag}, \lambda_\otimes)$, where:

- $W_\otimes = \{ \langle w, a \rangle \in W \times A \mid (M, w) \models \text{pre}(a) \}$;
- $\langle w, a \rangle \approx^\otimes_i \langle w, b \rangle$ whenever $w \approx_i u$ and $a \approx_i b$;
- $p \in \lambda_\otimes(w, a)$ whenever $(M, w) \models \text{post}(a, p)$.

We lift the product to epistemic states and epistemic actions: whenever $M, w \models \text{pre}(a)$, we let $(M, w) \otimes (A, a)$ be the epistemic state $(M \otimes A, (w, a))$.

**Example 3.** Figure 4 shows $(M \otimes A, (w, \text{push}_i))$, that describes the epistemic state after execution of action $\text{push}_i$; from the epistemic state of Figure 1 (strictly speaking, it only displays worlds connected to the actual one). The physical situation has not changed: the container is still on the right. However agent $\ell$, who initially considered it possible that the container is on the left, and cannot distinguish between $\text{push}_i$ and $\text{wait}_i$, still considers it possible that the container is on the left. Some information has changed though: initially, agent $\ell$ knew that robots had energy, but now he considers it possible that robot $i$’s energy has become low.

3 Concurrent Actions

Fix an epistemic state $(M, w)$ and an action model $A$. From now on, atomic propositions are partitioned into shared propositions ($AP^s$) that all agents can modify, and private ones: $AP^p_i$ is the set of private propositions of agent $i$. As a result, $AP = AP^s \cup \bigcup_{i \in Ag} AP^p_i$, where $\cup$ is the disjoint union. An agent can play any action that does not modify private propositions of others. We gather in set $A_i$ those actions, namely those whose postconditions are undefined on $\bigcup_{j \neq i} AP^p_j$. A joint action is a tuple $\bar{a} = (a_1, \ldots, a_N) \in \prod_{i \in Ag} A_i$, and we let $\bar{a}_j$ denote action $a_j$. Finally, a joint action $\bar{a}$ is available in $w$ when every individual action $a_j$ can be executed in $w$: $(M, w) \models \text{pre}(\bar{a}_j) \land \cdots \land \text{pre}(\bar{a}_u)$.

3.1 Conflicts

We define a formula $\text{noconflict}(\bar{a})(p)$ expressing that all individual actions of a joint action $\bar{a}$ agree on their effect (if any) on proposition $p$. We first introduce the set $Ag(\bar{a}, p)$ of agents whose individual action in $\bar{a}$ has an effect on $p$. Formally, $Ag(\bar{a}, p)$ is the set $\{ i = 1, \ldots, N \mid \text{post}(\bar{a}_i)(p) \text{ is defined} \}$. Then we let $\text{noconflict}(\bar{a})(p)$ be the formula

$$\bigwedge_{i \in Ag(\bar{a}, p)} \text{post}(\bar{a}_i)(p) \lor \bigwedge_{i \in Ag(\bar{a}, p)} \neg \text{post}(\bar{a}_i)(p).$$

Now, in a situation where all individual actions of the joint action $\bar{a}$ agree on their effect on each proposition, the effect of executing them concurrently can be determined unambiguously. Such a situation is expressed by the formula $\text{noconflict}(\bar{a})(p)$.

**Definition 6.** A joint action $\bar{a}$ is non-conflicting in $w$ if $M, w \models \text{noconflict}(\bar{a})$. Otherwise $\bar{a}$ is conflicting in $w$.

**Example 4.** In our running example, $(\text{push}_i, \text{push}_j)$ is conflicting in both worlds $w$ and $u$.

To resolve conflicts, we resort to a scheduler.

3.2 The Role of the Scheduler

Given a conflicting available joint action, the scheduler selects a maximal subset of consistent individual actions, and inhibits the remaining actions by “cancelling” their effects on shared variables. An inhibited action of agent $i$ may still have effects on $AP^p_i$, which may differ from the original ones. The inhibiting mechanism is implemented by a ghost mapping $\circ : A \rightarrow A$ with the following properties: for every agent $i \in Ag$ and every action $a \in A_i$, the ghost $\circ(a)$ (written $a_\circ$) must be in $A_i$ and $\text{post}(a_\circ)$ is undefined on $AP^p$. This assumption ensures that no conflict arises when inhibiting actions. We also assume that $\circ(a)$ is executable as long as $a$ is, i.e., that $\text{pre}(a_\circ)$ is a logical consequence of $\text{pre}(a)$.

**Example 5.** For the robots, $AP^s = \{ \text{left} \}$, $AP^p_i = \{ \text{lowE}_i \}$, $AP^p_j = \{ \text{lowE}_j \}$, and $AP^p_\ell = AP^p = \emptyset$. Facing the conflicting joint action $(\text{push}_i, \text{push}_j)$, the scheduler can choose between $(\text{push}_\circ i, \text{push}_j)$ and $(\text{push}_i, \text{push}_\circ j)$, and we set $\text{push}_{\circ i} := \text{wait}$, and $\text{push}_{\circ j} := \text{wait}$.

The epistemic relations from a ghost can be set in many different ways, and the choice is a design matter. For example, assume that ghost actions are distinct from those that can be played by agents, and that ghosts are not epistemically related to those playable actions. Then the scheduler’s selection is public: indeed, when for joint action $\bar{a}$ scheduler inhibits the action of agent $i$, all agents are implicitly informed that agent $i$ is discarded.

**Definition 7.** If $A = A_\circ \cup A_\circ$ is split between playable actions $A_\circ$ and ghosts ones $A_\circ$, and $A_\circ$ and $A_\circ$ are not related by any $\approx_i$, we say that the scheduler is public.

In the sequel, we consider that action models $A$ are implicitly equipped with a ghost mapping.

3.3 Concurrent Update Product

To define the concurrent execution of possibly conflicting actions, we introduce a concurrent update product $\otimes$ that generalises the classic update product $\otimes$ (Definition 5).

Before defining this product, we formalise the possible choices for the scheduler. We say that joint action $\bar{b}$ is a sub-action of joint action $\bar{a}$, written $\bar{b} \leq \bar{a}$, whenever $\bar{b}$ is obtained
from \( \bar{a} \) by replacing some individual actions by their ghosts. Formally, \( \bar{b} \preceq \bar{a} \) if \( b_i \in \{ \bar{a}_i, (\bar{a})_i \} \), for every \( 1 \leq i \leq N \). Remark that, by definition of the ghost mapping, if \( \bar{a} \) is available in \( w \), so are its sub-actions.

Given a joint action \( \bar{a} \), we let \( \text{Max}(\bar{a}, w) \) be the set of joint actions composed of the \( \preceq \)-maximal sub-actions of \( \bar{a} \) non-conflicting in \( w \). Elements of \( \text{Max}(\bar{a}, w) \) are therefore joint actions non-conflicting in \( w \) where a minimal number of actions are inhibited. Notice that \( \text{Max}(\bar{a}, w) = \{ \bar{a} \} \) as soon as \( \bar{a} \) is non-conflicting in \( w \).

Example 6. The set \( \text{Max}(\{\text{push}, \text{push}\}, w) \) is \( \{ \text{wait}, \text{push}\} \) (for readability, all reflexive edges are missing).

We now define the concurrent update product, that implements the execution of joint actions.

Definition 8 (Concurrent update product). The concurrent update product of \( M \) and \( A \) is the Kripke model \( M \boxplus A = (W^\boxplus, \approx^\boxplus_A, \lambda^\boxplus) \), where:

- \( W^\boxplus = \{ (w, \bar{b}) \in W \times A^N | \bar{b} \in \text{Max}(\bar{a}, u) \} \) for some joint action \( \bar{a} \) available in \( w \);
- \( (w, \bar{b}) \approx^\boxplus_A (u, \bar{c}) \) if \( w \approx u \) and \( \bar{b}_j \approx \bar{c}_j \) for all \( j \);
- \( p \in \lambda^\boxplus(u, \bar{b}) \) if \( (M, u) \models \bigwedge_{i \in A_g(\bar{b}, p)} \text{Post}(b_i, p) \).

We can now define the set of epistemic states that can result from executing a joint action \( \bar{a} \) in \( (M, w) \):

\[
(M, w) \boxplus A := \{ (M \boxplus A, (w, \bar{b}) | \bar{b} \in \text{Max}(\bar{a}, w) \}.
\]

This set is a singleton when \( \bar{a} \) is non-conflicting in \( w \). Remark that the component-wise definition of concurrent epistemic relations \( \approx^\boxplus_A \) in Definition 8 makes the identity of who performs an action common knowledge.

Example 7. The pointed concurrent update product \( (M \boxplus A, (w, \{\text{push}, \text{wait}\})) \) of Figure 5 belongs to \( (M, w) \boxplus A \) (\( \{\text{push}, \text{push}\} \)). Action \( \text{push}_1 \) was inhibited, but agent \( \ell \) considers it possible that \( \text{lowE}_1 \) is now true.

## 4 Concurrent DEL Games

In this section, we show how DEL can represent infinite concurrent game arenas, in which players act simultaneously. We then show how, as for turn-based games [Maubert et al., 2019; Maubert et al., 2020] these infinite game arenas can in some cases be folded back into finite ones, when actions are all public, or when they are all propositional. Finally, we show that in these cases, one can model check the epistemic strategic logic \( \text{ATL}^*_k \) on DEL-represented concurrent games.

### 4.1 Concurrent Game Arenas

Here we recall (nondeterministic) concurrent game arenas with imperfect information and related notions.

Definition 9. A concurrent game arena is a tuple \( G = (\text{Act}, V, v_{\text{ini}}, (\text{Act}_i)_{i \in A_g}, \Delta, (\approx_i)_{i \in A_g}, \lambda) \), where

- \( \text{Act} \) is a non-empty set of actions,
- \( V \) is a non-empty set of positions,
- \( v_{\text{ini}} \in V \) is an initial position,
- \( \lambda_i : V \to 2^\text{Act} \setminus \{ \emptyset \} \) is a repertoire function for \( i \),
- \( \Delta \subseteq V \times \text{Act}_N \times V \) is a transition relation,
- \( \approx_i \subseteq V \times V \) is an indistinguishability equivalence relation for agent \( i \), and
- \( \lambda : V \to 2^{A^P} \) is a valuation function.

In a position \( v \) each agent \( i \) chooses an available action \( a_i \in \text{Act}(v) \), yielding a joint action \( \bar{a} = (a_1, \ldots, a_N) \). The set of joint actions available in \( v \) is written \( \text{JAct}(v) \). The game then moves nondeterministically to some position \( v' \in \Delta(v, \bar{a}) \). If, for every position \( v \) and joint action \( \bar{a} \), \( \{ v' | (v, \bar{a}, v') \in \Delta \} \) is a singleton, then \( G \) is deterministic and we may represent the transition relation as a function \( \delta : V \times \text{Act}_N \to V \).

A play \( \pi = v_0v_1v_2 \ldots \) is an infinite sequence of positions such that for all \( k \in \mathbb{N} \), there exists \( \bar{a} \in \text{Act}^k \) such that \( v_{k+1} \in \Delta(v_k, \bar{a}) \). We let \( \pi_k = v_k \) and \( \pi_{\leq k} = v_0v_1 \ldots v_k \). A history \( h = v_0v_1 \ldots v_n \) is a finite non-empty prefix of a play, last\((h) = v_n \) is the last position in \( h \), \( |h| = n + 1 \) is its length, and \( \text{Hist}^G \) is the set of histories in \( G \). We write \( \lambda(h) \) for \( \lambda(\text{last}(h)) \).

As usual for agents with synchronous perfect recall (see for instance [Fagin et al., 1995]), the indistinguishability relation of each agent \( i \) is lifted to histories as follows: \( h \approx_i h' \) if \( |h| = |h'| \) and \( h_k \approx_i h'_k \) for every \( k < |h| \).

A (uniform) strategy \( \sigma \) for an agent \( i \) is a function \( \sigma_i \) that associates an action to any history and such that, for every pair of indistinguishable histories \( h \approx_i h' \), \( \sigma(h) = \sigma(h') \). Classically, we assume that \( \text{Act}_i(v) = \text{Act}_i(v') \) whenever \( v \approx v' \). A strategy profile for a coalition of agents \( C \subseteq A_g \) is a tuple \( \sigma_C = (\sigma_i)_{i \in C} \), and \( \Sigma_C \) is the set of strategy profiles for coalition \( C \). Finally, an outcome of a strategy profile \( \sigma_C \) from a position \( v_0 \) is a play starting in position \( v_0 \) and in which agents in \( C \) follow the strategies in \( \Sigma_C \): formally, a play \( \pi = v_0v_1 \ldots \) is an outcome of \( (\sigma_i)_{i \in C} \) from \( v_0 \) if, for every \( k \in \mathbb{N} \), there is a joint action \( \bar{a} \) such that \( v_{k+1} \in \Delta(v_k, \bar{a}) \) and \( \bar{a}_i = \sigma_i(v_0 \ldots v_k) \), for all \( i \in C \). We let \( \text{out}(h, \sigma_C) \) be the set of outcomes of \( \sigma_C \) from position \( v \).

Definition 10. Given a concurrent game arena \( G = (\text{Act}, V, v_{\text{ini}}, (\text{Act}_i)_{i \in A_g}, \Delta, (\approx_i)_{i \in A_g}, \lambda) \), we define the unfolding of \( G \) as the game arena

\[
G^{\text{unf}} = (\text{Act}, V', v'_{\text{ini}}, (\text{Act}'_i)_{i \in A_g}, \Delta', (\approx'_i)_{i \in A_g}, \lambda')
\]

where \( V' = \text{Hist}^G \), \( v'_{\text{ini}} = v_{\text{ini}} \), for every \( h \in \text{Hist}^G \), \( \text{Act}'_i(h) = \text{Act}_i(\text{last}(h)) \), \( \lambda'(h) = \lambda(\text{last}(h)) \), \( \approx'_i \) is the synchronous perfect-recall lifting of \( \approx_i \) to histories, and

\[
\Delta' = \{ (h, \bar{a}, h \cdot v) | (\text{last}(h), \bar{a}, v) \in \Delta \}.
\]

We say that two game structures \( G \) and \( G' \) are equivalent whenever their unfoldings are isomorphic.
4.2 DEL Presentations of Game Arenas

We use DEL models to define infinite concurrent Arenas:

**Definition 11.** A DEL game presentation, or DEL game, is a tuple $\mathfrak{G} = (\mathcal{M}, w, A)$ where $(\mathcal{M}, w)$ is an initial epistemic state and $A$ is an action model.

The game starts in the initial epistemic state $(\mathcal{M}, w)$. In each round, each agent $i$ chooses an action $a_i \in A_i$ available in the current state $(\mathcal{M}', w')$, resulting in a joint action $\vec{a}$ (we assume that each agent always has at least one available action; if needed we may add a dummy action in the model). The next epistemic state is nondeterministically chosen by the scheduler among $(\mathcal{M}', w') \sqsubseteq \vec{a}$ (this choice is nontrivial if $\vec{a}$ is conflicting), and the game goes on.

After $n$ rounds in which the players chose joint actions $\vec{a}_1, \ldots, \vec{a}_n$, the epistemic state is of the form $\mathcal{M}^n = (\mathcal{M}, \vec{b}_0, \ldots, \vec{b}_n)$, where $\mathcal{M}^n$ is defined by letting $\mathcal{M}^0 = \mathcal{M}$ and $\mathcal{M}^{n+1} = \mathcal{M}^n \sqsubseteq A$, and each $\vec{b}_{k+1}$ is a maximal sub-action of $\vec{a}_{k+1}$ consistent in $\mathcal{M}^n$. In the following we may write $w \vec{b}_1 \ldots \vec{b}_n$ instead of $(w, \vec{b}_1, \ldots, \vec{b}_n)$, and call it a history. The length of a history in $\mathcal{M}^n$ is defined as $|w \vec{b}_1 \ldots \vec{b}_n| = n$. Since the length of a history determines the epistemic model to which it belongs, we may omit it and write $w \vec{b}_1 \ldots \vec{b}_n \models \varphi$ instead of $\mathcal{M}^n, w \vec{b}_1 \ldots \vec{b}_n \models \varphi$. We require that players know their available actions: if $w \vec{b}_1 \ldots \vec{b}_n \models u \vec{c}_1 \ldots \vec{c}_n$, then the same actions are available to $i$ in both worlds.

A DEL presentation denotes a concurrent game arena, where positions are epistemic states attainable from the initial one, and moves are obtained by applying the concurrent update product of Definition 8. Because this product yields several epistemic states, this concurrent game arena is nondeterministic.

**Definition 12.** Given a DEL game presentation $\mathfrak{G} = (\mathcal{M}, w, A)$ we define the concurrent game arena $G_{\mathfrak{G}} = (A, V, \text{vinit}, (\text{Act}_i)_{i \in A_G}, \Delta, \text{init}_i)_{i \in A_G}, \lambda)$ where, writing $\mathcal{M}^n = (W^n, (\equiv_i^n)_{i \in A_G}, \lambda^n)$ for every $n$:

- $A$ is the set of actions in $\mathcal{M}$,
- the set of positions is $V = \sqcup_{n \in \mathbb{N}} W^n$,
- the initial position is $\text{vinit} = w$,
- $\text{Act}_i(v) = \{a_i \in A_i \mid v \models \text{pre}(a_i)\}$,
- $\Delta = \{(v, \vec{a}, v') \mid v' \in V \sqsupseteq \vec{a}\}$,
- $v \approx_i v'$ if $|v| = |v'| = n$ and $v \equiv_i^n v'$, and
- $\lambda(v) = \lambda^n(v)$.

We want to reason about strategic abilities and knowledge on infinite concurrent games given as DEL game presentations. To do so we consider the logic $\mathsf{ATL}^*_K$.

**Definition 13.** The syntax of $\mathsf{ATL}^*_K$ is given by the following grammar:

$$
\varphi :: \Box \varphi \mid \varphi_1 \lor \varphi_2 \mid K_i \varphi \mid (C) \psi \\
\psi :: \Box \psi \mid \psi_1 \lor \psi_2 \mid \bigcirc \psi \mid \psi_1 \psi_2
$$

with $p \in AP$, $C \subseteq A_G$ and $i \in A_G$.

Formula $(C) \psi$ reads as “coalition $C$ has a strategy profile to ensure $\psi$”, and the meaning of other operators is as usual.

We let $\equiv_C = \cup_{i \in C} \equiv_i$, and define the semantics of $\mathsf{ATL}^*_K$.

**Definition 14.** Let $G$ be a concurrent game arena, $v$ a position and $\pi$ a play. The semantics of $\mathsf{ATL}^*_K$ is defined as follows, where $h$ is a history in $G$, $\pi$ is a play and $i \in \mathbb{N}$ is a point in time (we omit boolean cases):

$$
G, h \models p \quad \text{if} \quad p \in \lambda(h) \\
G, h \models K_i \varphi \quad \text{if} \quad \forall h' \in \text{His}^G \text{ s.t. } h_i \approx h, G, h' \models \varphi \\
G, h \models (C) \psi \quad \text{if} \quad \exists \sigma_C \in \Sigma_C \text{ s.t. } \forall h' \approx_C h, \forall \pi \in \out(h', \sigma_C), \ G, \pi \models \psi \\
G, \pi \models \varphi \quad \text{if} \quad \pi[0] \models \varphi \\
G, \pi \models C \psi \quad \text{if} \quad G, \pi_{\geq 1} \models \psi_2 \text{ and } \forall 0 \leq j < i, \ G, \pi_{\geq j} \models \psi_1
$$

**Remark 1.** We use the uninformed semantics for the knowledge operators, which does not depend on the strategies used by the agents, and is the usual one used in epistemic planning (see [Puchala, 2010; Maubert, 2014; Maubert et al., 2020] for more details). Also, we use the subjective semantics for strategic operators, in contrast with the objective one that only asks strategies to be winning from the actual current history (see [Bulling and Jamroga, 2014]).

Given a game arena $G$ with initial position $v_{\text{init}}$, we write $G \models \varphi$ if $G, v_{\text{init}} \models \varphi$. We study the following model-checking problem: given a DEL game presentation $\mathfrak{G}$ and a formula $\varphi \in \mathsf{ATL}^*_K$, do we have $G_{\mathfrak{G}} \models \varphi$?

4.3 Decidability for Public Actions

A deterministic game arena with transition function $\delta$ is said to have only public actions if, for all positions $v, v'$ and joint actions $\vec{a}, \vec{a}'$ such that $\delta(v, \vec{a}) \neq \delta(v', \vec{a}')$, we have $\delta(v, \vec{a}) \neq \delta(v', \vec{a}')$. The following is known for $\mathsf{ATL}^*_K$ on such arenas:

**Theorem 1** ([Belardinelli et al., 2017]). Model checking $\mathsf{ATL}^*_K$ on finite deterministic concurrent game structures with only public actions is $2$-EXPTIME-complete.

First we prove that DEL games with public actions can be finitely represented. Our proof yields nondeterministic game arenas, that we will then transform into deterministic ones that have only public actions in the sense of [Belardinelli et al., 2017]. We will need the following result on the concurrent update product with public action models and a public scheduler (see Definition 7):

**Lemma 2.** Let $\mathcal{M}$ be an epistemic model and $A$ an action model. If the scheduler is public, $A$ has only public actions and its ghost mapping $\circ : A_p \rightarrow A_\bullet$ is injective, then for all worlds $w, w' \in \mathcal{M}$, all joint actions $\vec{a}, \vec{a}'$, if $\vec{a} \neq \vec{a}'$ then for all $(w, \vec{b}) \in (\mathcal{M}, w) \sqsubseteq \vec{a}$ and $(w', \vec{b}') \in (\mathcal{M}, w') \sqsubseteq \vec{a}'$, for all agents $i$, $(w, \vec{b}) \neq (w', \vec{b}')$.

The injectivity assumption means that two different played actions cannot become the same after being inhibited, in which case the actions of DEL games could not be public in the sense of [Belardinelli et al., 2017].
The proof that the infinite game arena $G_\emptyset$ induced by a DEL game $\emptyset$ with public actions can be finitely represented is very similar to that in [Maubert et al., 2020]. It relies on the fact that updating an epistemic state with a joint action made of public actions can only decrease the size of the obtained epistemic states (when removing their disconnected components) as long as their ghosts are public too. It follows that only finitely many different epistemic models can be generated from a given initial state, up to isomorphism.

**Proposition 3.** Given a DEL game $\emptyset = (\mathcal{M}, w, \mathcal{A})$ with only public actions, one can build a finite game arena $G$ equivalent to $G_\emptyset$ and of size exponential in $|\emptyset|$.

Because of the scheduler, the resulting arena is nondeterministic. To obtain Theorem 4 below it essentially remains to show that we can transform these nondeterministic arenas into deterministic ones that have only public actions in the sense of [Belardinelli et al., 2017].

**Theorem 4.** Model checking $\text{ATL}_K^n$ on DEL concurrent games with public actions, public scheduler and injective ghost mapping is $2$-EXPTIME-complete.

**Proof sketch.** The lower bound comes from LTL synthesis [Pnueli and Rosner, 1989]. For the upper bounds, let $\emptyset$ be a DEL presentation and $\Phi$ an $\text{ATL}_K^n$ formula. First, by Proposition 3 we build in exponential time a finite nondeterministic game arena $G = (\text{Act}, \mathcal{V}, \mathcal{V}_{\text{uni}}, (\text{Act}_i)_{i \in \mathcal{A}}, \Delta, (\approx_i)_{i \in \mathcal{A}}, \lambda)$ equivalent to $G_\emptyset$. We build a deterministic game arena $G^d$ by adding a special agent, the scheduler $s$, who has perfect information and whose possible actions Act$_s$ allow him to resolve the non-determinism. The agents choose their actions, and then the scheduler resolves nondeterminism. To do so we introduce intermediary positions of the form $(v, \tilde{a})$: in each round, in a position $v$, the agents in $\mathcal{A}g$ choose a joint action $\tilde{a}$ (the scheduler can only play some dummy action), and the game moves deterministically to $(v, \tilde{a})$. Now all “normal” agents can only choose the dummy move, and the scheduler can choose a maximal consistent sub-action $\tilde{b}$ of $\tilde{a}$, which determines the new position $v'$. Indistinguishability relations for all normal agents are as in $G$ for positions $v$, and we let $(v, \tilde{a}) \approx_i (v', \tilde{a}')$ iff $v \approx_i v'$ and $\tilde{a} = \tilde{a}'$. This definition is correct because, by Lemma 2, when the agents choose different joint actions, they distinguish the resulting epistemic states no matter the choices of the scheduler. One can check that $G^d$ has only public actions in the sense of [Belardinelli et al., 2017]. We then duplicate all next operators in $\Phi$ to skip artificial positions of the form $(v, \tilde{a})$. To check $\Phi$ on $G^d$, the procedure from [Belardinelli et al., 2017] takes time doubly exponential in the size of $\Phi$ and exponential in the size of $G^d$, which is itself of size exponential in the size of the original DEL game presentation $\emptyset$. \hfill $\Box$

4.4 Decidability for Propositional Actions

A DEL game presentation presents hierarchical information if the set of agents $\mathcal{A}g$ can be totally ordered $(i_1 < \ldots < i_N)$ so that $\approx_i \subseteq \approx_{i+1}$ for each $1 \leq i < N$, and similarly for concurrent game arenas. We prove that when, in a DEL game presentation, all actions are propositional actions and information is hierarchical, model checking $\text{ATL}_K^n$ is decidable.

We first generalise to the concurrent setting a result from [Maubert et al., 2019] which states that infinite turn-based DEL games induced by propositional models can be finitely represented. As for public actions, the game arena that we obtain is nondeterministic.

**Proposition 5.** Let $\emptyset = (\mathcal{M}, w, \mathcal{A})$ be a DEL game where $\mathcal{A}$ is propositional. One can build a finite nondeterministic game arena $G$ equivalent to $G_\emptyset$, of size exponential in $|\emptyset|$.

We will invoke the following result, that was stated for deterministic game arenas:

**Theorem 6 ([Berthon et al., 2017]).** Model checking $\text{ATL}_K^n$ with uninformed semantics is decidable on deterministic game arenas with hierarchical information.

The result in [Berthon et al., 2017] is for $\text{ATL}^*$ without knowledge operators, and for the objective semantics of strategic operators. But their bottom-up algorithm can be easily extended to deal with knowledge operators and subjective semantics: one first performs a powerset construction on the game arena to include sufficient information to evaluate knowledge operators positionally (for instance using $k$-trees [van der Meyden, 1998]), and then one evaluates knowledge and strategic modalities in a bottom-up fashion.

Thanks to Proposition 5 and a reduction to deterministic arenas similar to the one presented in the proof of Theorem 4, we can use Theorem 6 and obtain that:

**Theorem 7.** Model checking $\text{ATL}_K^n$ on DEL game presentations with propositional actions and hierarchical information is decidable.

However the complexity is nonelementary, as it is already the case for multiplayer reachability games with hierarchical information [Peterson et al., 2001]. The number of exponentials will be the maximum between the number of agents with different observation power and the alternation of knowledge operators in the formula.

5 Conclusion

We proposed a setting that permits the succinct specification of concurrent game arenas with epistemic actions. Conflicts are resolved by a scheduler that nondeterministically selects consistent subsets of actions, and inhibits the discarded ones. We established that on such games one can perform strategic reasoning, or more precisely one can model check the strategic epistemic logic $\text{ATL}_K^n$ in two cases: when actions are public (Theorem 4) and when actions are propositional and information is hierarchical among the agents (Theorem 7).

Note that we could tune the scheduler differently: in our robot example, a third choice for $\langle \text{push}_{\text{t}}, \text{push}_{\text{b}} \rangle$ could be $(\text{wait}_{\text{t}}, \text{wait}_{\text{b}})$, which would capture blocking conflicts. Technically, we could relax the selection to other sub-actions than $\equiv$-maximal ones, and the complexities established in our main theorems would still hold for such different schedulers.

One could also consider fairness requirements to ensure that the scheduler does not favor some agents in resolving conflicts, for instance requiring that if an agent infinitely often enters a conflict, then infinitely often his action is selected by the scheduler. Fairness constraints can often be expressed directly in $\text{ATL}_K^n$, in which case our main results still hold.
References


